Hardware Locality (hwloc)

0.9.1rc1

Generated by Doxygen 1.5.9

Fri Oct 9 15:27:29 2009

Contents

Chapter 1

hwloc

Portable abstraction of hierarchical architectures for high-performance computing

1.1 Introduction

hwloc provides a portable abstraction (across OS, versions, architectures, ...) of the hierarchical topology of modern architectures. It primarily aims at helping high-performance computing applications with gathering information about the hardware so as to exploit it accordingly and efficiently.

hwloc provides a hierarchical view of the machine, NUMA memory nodes, sockets, shared caches, cores and simultaneous multithreading. It also gathers various attributes such as cache and memory information.

hwloc supports the following operating systems:

- Linux (including old kernels not having sysfs topology information, with knowledge of cpusets, offline cpus, and Kerrighed support)
- Solaris
- AIX
- Darwin
- OSF/1 (aka. Tru64)
- HP-UX
- Windows

- For other OSes, only the number of processors is available for now.

For development and debugging purposes, hwloc also offers the ability to work on fake topologies:

- Symmetrical tree of resources generated from a list of level arities
- Remote machine simulation through the gathering of Linux sysfs topology files

hwloc may also display the topology in a convenient format, either in graphical mode, or by exporting in PDF, PNG, FIG, ... format, or in text mode (see Examples below).

hwloc offers a programming interface for manipulating topologies and objects. It also brings a powerful cpu bitmap API that is used to describe topology objects location on physical/logical processors. See the Programming interface below. It may also be used to binding applications onto certain cores or memory nodes. Several utility programs are also provided to ease command-line manipulation of topology objects, binding of processes, ...

1.2 Installation

hwloc (`http://www.open-mpi.org/projects/hwloc/`) is available under the BSD license. It is hosted by Open MPI (`http://www.open-mpi.org/`). The current SVN snapshot can be fetched with:

- svn checkout `http://svn.open-mpi.org/svn/hwloc/trunk` hwloc-trunk
- cd hwloc-trunk
- ./autogen.sh

Note that autoconf >=2.60, automake >=1.10 and libtool >=2.2.6 are required in that case.

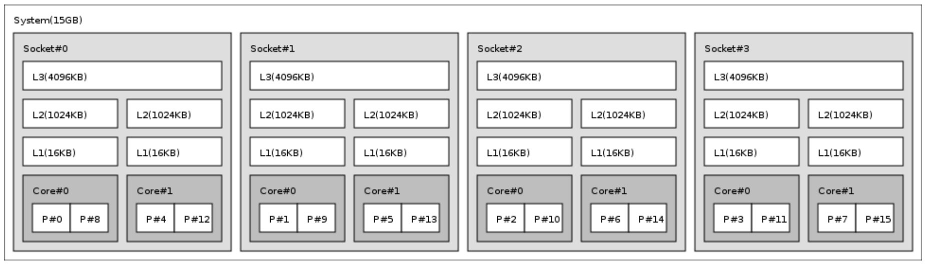
Installation by itself is as usual:

- ./configure –prefix=...
- make
- make install

Lstopo's fig support is always available. To get support for pdf, ps and png support, cairo is needed. To get support for xml, libxml2 is needed.

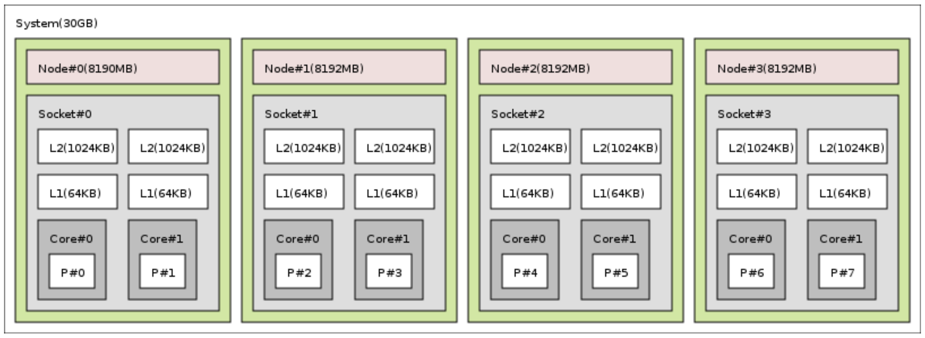
1.3 Examples

On a 4-socket 2-core machine with hyperthreading, the `lstopo` tool may show the following outputs:



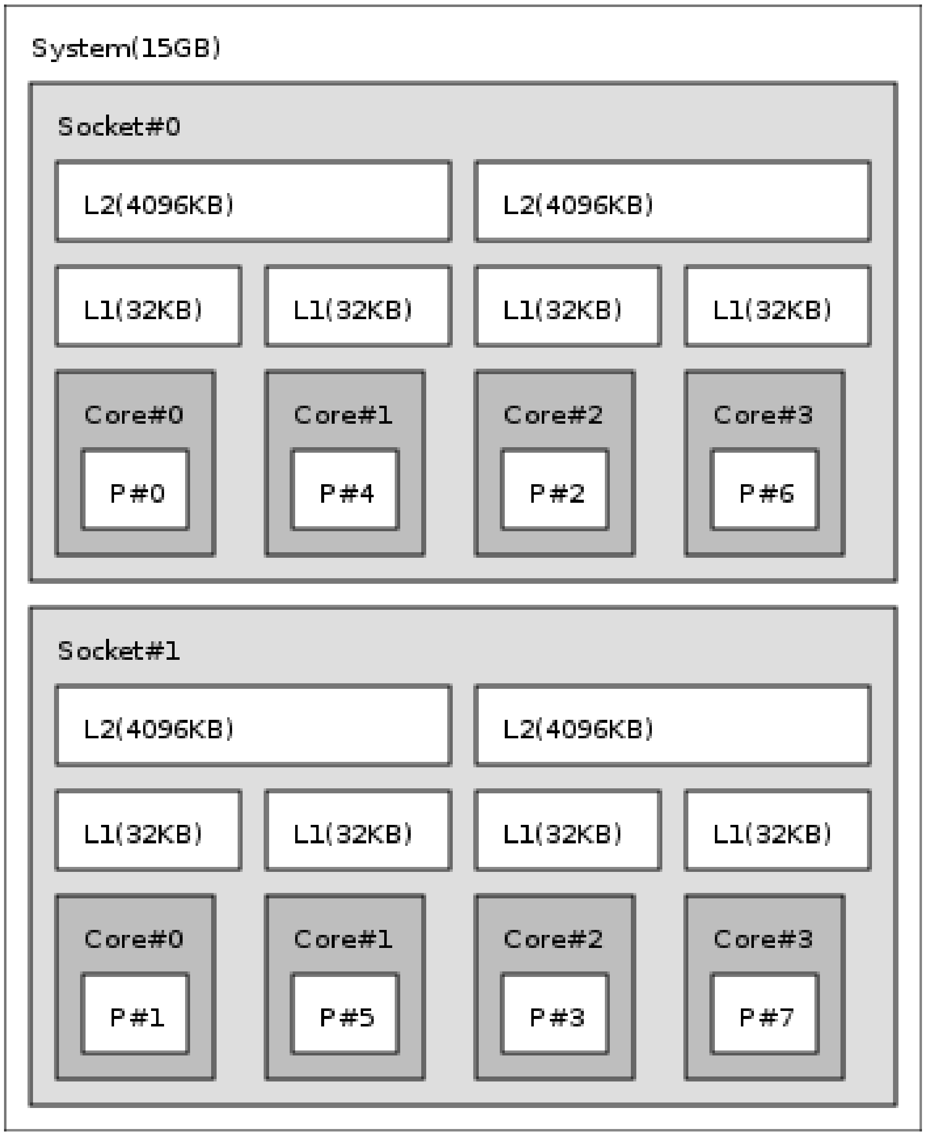
```
System(15GB)
  Socket#0 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#0
      P#8
    L2(1024KB) + L1(16KB) + Core#1
      P#4
      P#12
  Socket#1 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#1
      P#9
    L2(1024KB) + L1(16KB) + Core#1
      P#5
      P#13
  Socket#2 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#2
      P#10
    L2(1024KB) + L1(16KB) + Core#1
      P#6
      P#14
  Socket#3 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#3
      P#11
    L2(1024KB) + L1(16KB) + Core#1
      P#7
      P#15
```

On a 4-socket 2-core Opteron NUMA machine, the `lstopo` tool may show the following outputs:



```
System(62GB)
  Node#0(8190MB) + Socket#0
    L2(1024KB) + L1(64KB) + Core#0 + P#0
    L2(1024KB) + L1(64KB) + Core#1 + P#1
  Node#1(8192MB) + Socket#1
    L2(1024KB) + L1(64KB) + Core#0 + P#2
    L2(1024KB) + L1(64KB) + Core#1 + P#3
  Node#2(8192MB) + Socket#2
    L2(1024KB) + L1(64KB) + Core#0 + P#4
    L2(1024KB) + L1(64KB) + Core#1 + P#5
  Node#3(8192MB) + Socket#3
    L2(1024KB) + L1(64KB) + Core#0 + P#6
    L2(1024KB) + L1(64KB) + Core#1 + P#7
  Node#4(8192MB) + Socket#4
    L2(1024KB) + L1(64KB) + Core#0 + P#8
    L2(1024KB) + L1(64KB) + Core#1 + P#9
  Node#5(8192MB) + Socket#5
    L2(1024KB) + L1(64KB) + Core#0 + P#10
    L2(1024KB) + L1(64KB) + Core#1 + P#11
  Node#6(8192MB) + Socket#6
    L2(1024KB) + L1(64KB) + Core#0 + P#12
    L2(1024KB) + L1(64KB) + Core#1 + P#13
  Node#7(8192MB) + Socket#7
    L2(1024KB) + L1(64KB) + Core#0 + P#14
    L2(1024KB) + L1(64KB) + Core#1 + P#15
```

On a 2-socket quad-core Xeon (pre-Nehalem ones assembling 2 dual-core dies into each socket):



```
System(15GB)
  Socket#0
    L2(4096KB)
      L1(32KB) + Core#0 + P#0
      L1(32KB) + Core#1 + P#4
    L2(4096KB)
      L1(32KB) + Core#2 + P#2
      L1(32KB) + Core#3 + P#6
  Socket#1
    L2(4096KB)
```

```
      L1(32KB) + Core#0 + P#1
      L1(32KB) + Core#1 + P#5
    L2(4096KB)
      L1(32KB) + Core#2 + P#3
      L1(32KB) + Core#3 + P#7
```

1.4 Programming interface

The basic interface is available in hwloc.h . It mostly offers low-level routines for advanced programmers that want to manually manipulate objects and follow links between them. Most users should look at hwloc/helper.h which provides a lot of interesting higher-level traversal examples.

Each object contains a cpuset which describes the list of processors that it contains. These cpusets may be used for Binding. hwloc offers an extensive cpuset manipulation interface in hwloc/cpuset.h .

Moreover, hwloc also comes with additional helpers for interoperability with several commonly used environments. For Linux, some specific helpers are available in hwloc/linux.h , and hwloc/linux-libnuma.h if using libnuma. On glibc-based systems, additional helpers are available in hwloc/glibc-sched.h . For systems with the Infiniband Verbs library, some dedicated helpers are provided in hwloc/ibverbs.h .

To precisely define the vocabulary used by hwloc, a Glossary is available and should probably be read first.

Further documentation is available in html, manual pages, and pdf format in the source tarball in doc/doxygen-doc/ (after doxygen compilation for svn checkouts) and are installed in $prefix/share/doc/hwloc/ and the usual manual repository.

The following section presents an example of API usage.

1.5 Interface example

This section shows how to use hwloc with an small example `hwloc-hello.c` that just prints the topology and binds itself to the first processor of the second core of the machine.

Hardware Location provides a pkg-config object, so compiling the example boils down to

```
CFLAGS+=$(pkg-config --cflags hwloc)
LDLIBS+=$(pkg-config --libs hwloc)
cc hwloc-hello.c $(CFLAGS) -o hwloc-hello $(LDLIBS)
```

```
/* topo-hello.c */
#include <hwloc.h>
```

```
static void print_children(hwloc_topology_t topology, hwloc_obj_t obj, int depth)

{
        char string[128];
        int i;

        hwloc_obj_snprintf(string, sizeof(string), topology, obj, "#", 0);
        printf("%*s%s\n", 2*depth, "", string);
        for (i = 0; i < obj->arity; i++)
                print_children(topology, obj->children[i], depth + 1);
}

int main(void)
{
        /* Topology object */
        hwloc_topology_t topology;

        /* Allocate and initialize topology object.  */
        hwloc_topology_init(&topology);

        /* ... Optionally, put detection configuration here to e.g. ignore some
           objects types, define a synthetic topology, etc....  The default is
           to detect all the objects of the machine that the caller is allowed
           to access.
           See Configure Topology Detection.  */

        /* Perform the topology detection.  */
        hwloc_topology_load(topology);


        /* Optionally, get some additional topology information
         * in case we need the topology depth later.
         */
        unsigned topodepth = hwloc_topology_get_depth(topology);


        /* Walk the topology with an array style, from level 0 (always the
         * system level) to the lowest level (always the proc level). */
        int depth, i;
        char string[128];
        for (depth = 0; depth < topodepth; depth++) {
                for (i = 0; i < hwloc_get_nbobjs_by_depth(topology, depth); i++)
      {
                        hwloc_obj_snprintf(string, sizeof(string), topology,
                                        hwloc_get_obj_by_depth(topology, depth, i
      ), "#", 0);
                        printf("%s\n", string);
                }
        }

        /* Walk the topology with a tree style.  */
        print_children(topology, hwloc_get_system_obj(topology), 0);


        /* Print the number of sockets.  */
        depth = hwloc_get_type_depth(topology, HWLOC_OBJ_SOCKET);
        if (depth == HWLOC_TYPE_DEPTH_UNKNOWN)
                printf("The number of sockets is unknown\n");
        else
                printf("%u socket(s)\n", hwloc_get_nbobjs_by_depth(topology, dept
      h));


        /* Find out where cores are, or else smaller sets of CPUs if the OS
         * doesn't have the notion of core. */
        depth = hwloc_get_type_or_below_depth(topology, HWLOC_OBJ_CORE);

        /* Get last one.  */
        hwloc_obj_t obj = hwloc_get_obj_by_depth(topology, depth,
      hwloc_get_nbobjs_by_depth(topology, depth) - 1);
        if (!obj)
                return 0;

        /* Get a copy of its cpuset that we may modify.  */
        hwloc_cpuset_t cpuset = hwloc_cpuset_dup(obj->cpuset);

        /* Get only one logical processor (in case the core is SMT/hyperthreaded)
      .  */
        hwloc_cpuset_singlify(cpuset);

        /* And try to bind ourself there.  */
        if (hwloc_set_cpubind(topology, cpuset, 0)) {
                char *str = NULL;
                hwloc_cpuset_asprintf(&str, obj->cpuset);
                printf("Couldn't bind to cpuset %s\n", str);
                free(str);
        }

        /* Free our cpuset copy */
        hwloc_cpuset_free(cpuset);

        /* Destroy topology object.  */
        hwloc_topology_destroy(topology);

        return 0;
}
```

1.6 Questions and bugs

Questions should be sent to the devel mailing list (`http://www.open-mpi.org/community/lists/hwloc.php`).
Bug reports should be reported in the tracker (`https://svn.open-mpi.org/trac/hwloc/`).

1.7 History / credits

hwloc is the evolution and merger of the libtopology (`http://runtime.bordeaux.inria.fr/libtopology/`) project and the Portable Linux Processor Affinity (PLPA) (`http://www.open-mpi.org/projects/plpa/`) project. Because of functional and ideological overlap, these two code bases and ideas were merged and released under the name "hwloc" as an Open MPI sub-project.

libtopology was initially developed by the INRIA Runtime Team-Project (`http://runtime.bordeaux.inria.fr/`) (headed by Raymond Namyst (`http://dept-info.labri.fr/~namyst/`)). PLPA was initially developed by the Open MPI development team as a sub-project. Both are now deprecated in favor of hwloc, which is distributed as an Open MPI sub-project.

Chapter 2

Glossary

Object Interesting kind of part of the system, such as a Core, a Cache, a Memory node, etc. The different types detected by hwloc are detailed in the hwloc_obj_-type_e enumeration.

They are topologically sorted by CPU set into a tree whose root is the System object which always exists.

CPU set The set of logical processors logically included in an object, if any

Father object The object logically containing the current object, for instance because its CPU set includes the CPU set of the current object.

Children objects The object contained in the current object because their CPU set is included in the CPU set of the current object.

Arity The number of children of an object

Sibling objects Objects of the same type which have the same father

Sibling rank Index to uniquely identify objecst of the same type which have the same father, numbered from 0 to the arity of the father minus one.

Cousin objects Objects of the same type as the current object

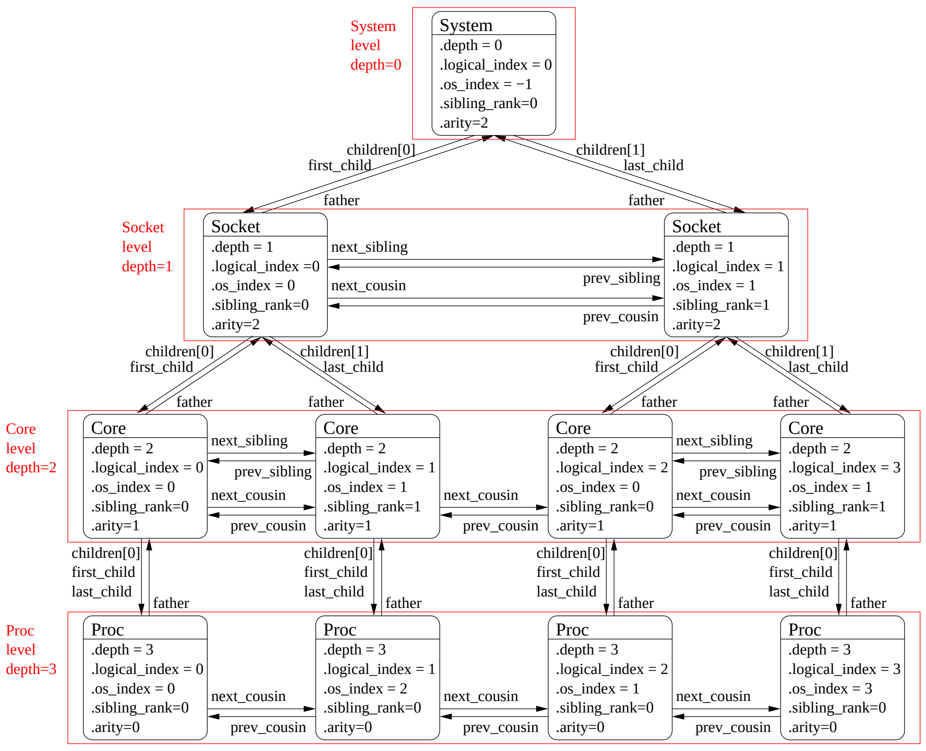
Level Set of objects of the same type

OS index The index that the OS uses to identify the object. This may sometimes be completely arbitrary or depend on the BIOS configuration.

Depth Nesting level in the object tree, starting from the System object.

Logical index Index to uniquely identify objects of the same type. This index is always linear from 0 to the number of objects of the level for that type, to express proximity. It could also be called cousin rank.

The following diagram can help to understand the vocabulary of the relationships by showing the example of a machine with two dual core non-SMT sockets, thus a topology with 4 levels.



It can be noticed that for Processor objects, the logical index, computed linearly by hwloc, is not the same as the OS index.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

Module Documentation

5.1 Topology context

Typedefs

- typedef struct hwloc_topology ∗ hwloc_topology_t

  Topology context.

5.1.1 Typedef Documentation

5.1.1.1 typedef struct hwloc_topology∗ hwloc_topology_t

Topology context.

To be initialized with hwloc_topology_init() and built with hwloc_topology_load().

5.2 Topology Object Types

Defines

- #define HWLOC_TYPE_UNORDERED INT_MAX

  Value returned by hwloc_compare_types when types can not be compared.

Enumerations

- enum hwloc_obj_type_t {

  HWLOC_OBJ_SYSTEM, HWLOC_OBJ_MACHINE, HWLOC_OBJ_NODE, HWLOC_OBJ_SOCKET,

  HWLOC_OBJ_CACHE, HWLOC_OBJ_CORE, HWLOC_OBJ_PROC, HWLOC_OBJ_MISC }

  Type of topology object.

Functions

- int hwloc_compare_types (hwloc_obj_type_t type1, hwloc_obj_type_t type2)

  Compare the depth of two object types.

5.2.1 Define Documentation

5.2.1.1 #define HWLOC_TYPE_UNORDERED INT_MAX

Value returned by hwloc_compare_types when types can not be compared.

5.2.2 Enumeration Type Documentation

5.2.2.1 enum hwloc_obj_type_t

Type of topology object.

Note:

Do not rely on the ordering or completeness of the values as new ones may be defined in the future! If you need to compare types, use hwloc_compare_types() instead.

Enumerator:

HWLOC_OBJ_SYSTEM Whole system (may be a cluster of machines). The whole system that is accessible to hwloc. That may comprise several machines in SSI systems like Kerrighed.

HWLOC_OBJ_MACHINE Machine. A set of processors and memory with cache coherency.

HWLOC_OBJ_NODE NUMA node. A set of processors around memory which the processors can directly access.

HWLOC_OBJ_SOCKET Socket, physical package, or chip. In the physical meaning, i.e. that you can add or remove physically.

HWLOC_OBJ_CACHE Data cache. Can be L1, L2, L3, ...

HWLOC_OBJ_CORE Core. A computation unit (may be shared by several logical processors).

HWLOC_OBJ_PROC (Logical) Processor. An execution unit (may share a core with some other logical processors, e.g. in the case of an SMT core).

Objects of this kind are always reported and can thus be used as fallback when others are not.

HWLOC_OBJ_MISC Miscellaneous objects. Objects which do not fit in the above but are detected by hwloc and are useful to take into account for affinity. For instance, some OSes expose their arbitrary processors aggregation this way.

5.2.3 Function Documentation

5.2.3.1 int hwloc_compare_types (hwloc_obj_type_t *type1*, hwloc_obj_type_t *type2*)

Compare the depth of two object types.

Types shouldn't be compared as they are, since newer ones may be added in the future. This function returns less than, equal to, or greater than zero if `type1` is considered to be respectively higher than, equal to, or deeper than `type2` in the hierarchy. If the types can not be compared (because it does not make sense), HWLOC_TYPE_-UNORDERED is returned. Object types containing CPUs can always be compared.

Note:

HWLOC_OBJ_SYSTEM will always be the highest, and HWLOC_OBJ_PROC will always be the deepest.

5.3 Topology Objects

Data Structures

- struct hwloc_obj

  Structure of a topology object.

- union hwloc_obj_attr_u

  Object type-specific Attributes.

Typedefs

- typedef struct hwloc_obj ∗ hwloc_obj_t

5.3.1 Typedef Documentation

5.3.1.1 typedef struct hwloc_obj∗ hwloc_obj_t

5.4 Create and Destroy Topologies

Functions

- int hwloc_topology_init (hwloc_topology_t ∗topologyp)

  Allocate a topology context.

- int hwloc_topology_load (hwloc_topology_t topology)

  Build the actual topology.

- void hwloc_topology_destroy (hwloc_topology_t topology)

  Terminate and free a topology context.

- void hwloc_topology_check (hwloc_topology_t topology)

  Run internal checks on a topology structure.

5.4.1 Function Documentation

5.4.1.1 void hwloc_topology_check (hwloc_topology_t *topology*)

Run internal checks on a topology structure.

Parameters:

topology is the topology to be checked

5.4.1.2 void hwloc_topology_destroy (hwloc_topology_t *topology*)

Terminate and free a topology context.

Parameters:

topology is the topology to be freed

5.4.1.3 int hwloc_topology_init (hwloc_topology_t ∗ *topologyp*)

Allocate a topology context.

Parameters:

→ ***topologyp*** is assigned a pointer to the new allocated context.

Returns:

0 on success, -1 on error.

5.4.1.4 int hwloc_topology_load (hwloc_topology_t *topology*)

Build the actual topology.

Build the actual topology once initialized with hwloc_topology_init() and tuned with hwlocality_configuration routine. No other routine may be called earlier using this topology context.

Parameters:

topology is the topology to be loaded with objects.

Returns:

0 on success, -1 on error.

See also:

Configure Topology Detection

5.5 Configure Topology Detection

Enumerations

- enum hwloc_topology_flags_e { HWLOC_TOPOLOGY_FLAG_WHOLE_-SYSTEM = (1<<0), HWLOC_TOPOLOGY_FLAG_IS_THISSYSTEM = (1<<1) }

  Flags to be set onto a topology context before load.

Functions

- int hwloc_topology_ignore_type (hwloc_topology_t topology, hwloc_obj_-type_t type)

  Ignore an object type.

- int hwloc_topology_ignore_type_keep_structure (hwloc_topology_t topology, hwloc_obj_type_t type)

  Ignore an object type if it does not bring any structure.

- int hwloc_topology_ignore_all_keep_structure (hwloc_topology_t topology)

  Ignore all objects that do not bring any structure.

- int hwloc_topology_set_flags (hwloc_topology_t topology, unsigned long flags)

  Set OR'ed flags to non-yet-loaded topology.

- int hwloc_topology_set_fsroot (hwloc_topology_t restrict topology, const char ∗restrict fsroot_path)

  Change the file-system root path when building the topology from sysfs/procfs.

- int hwloc_topology_set_synthetic (hwloc_topology_t restrict topology, const char ∗restrict description)

  Enable synthetic topology.

- int hwloc_topology_set_xml (hwloc_topology_t restrict topology, const char ∗restrict xmlpath)

  Enable XML-file based topology.

5.5.1 Detailed Description

These functions can optionally be called between hwloc_topology_init() and hwloc_-topology_load() to configure how the detection should be performed, e.g. to ignore some objects types, define a synthetic topology, etc.

If none of them is called, the default is to detect all the objects of the machine that the caller is allowed to access.

5.5.2 Enumeration Type Documentation

5.5.2.1 enum hwloc_topology_flags_e

Flags to be set onto a topology context before load.

Flags should be given to hwloc_topology_set_flags().

Enumerator:

HWLOC_TOPOLOGY_FLAG_WHOLE_SYSTEM

HWLOC_TOPOLOGY_FLAG_IS_THISSYSTEM

5.5.3 Function Documentation

5.5.3.1 int hwloc_topology_ignore_all_keep_structure (hwloc_topology_t *topology*)

Ignore all objects that do not bring any structure.

Ignore all objects that do not bring any structure: Each ignored object should have a single children or be the only child of its father.

5.5.3.2 int hwloc_topology_ignore_type (hwloc_topology_t *topology*, hwloc_obj_type_t *type*)

Ignore an object type.

Ignore all objects from the given type. The top-level type HWLOC_OBJ_SYSTEM and bottom-level type HWLOC_OBJ_PROC may not be ignored.

5.5.3.3 int hwloc_topology_ignore_type_keep_structure (hwloc_topology_t *topology*, hwloc_obj_type_t *type*)

Ignore an object type if it does not bring any structure.

Ignore all objects from the given type as long as they do not bring any structure: Each ignored object should have a single children or be the only child of its father. The top-level type HWLOC_OBJ_SYSTEM and bottom-level type HWLOC_OBJ_PROC may not be ignored.

5.5.3.4 int hwloc_topology_set_flags (hwloc_topology_t *topology*, unsigned long *flags*)

Set OR'ed flags to non-yet-loaded topology.

Set a OR'ed set of hwloc_topology_flags_e onto a topology that was not yet loaded.

5.5.3.5 int hwloc_topology_set_fsroot (hwloc_topology_t restrict *topology*, const char *restrict *fsroot_path*)

Change the file-system root path when building the topology from sysfs/procfs.

On Linux system, use sysfs and procfs files as if they were mounted on the given `fsroot_path` instead of the main file-system root. Not using the main file-system root causes hwloc_topology_is_thissystem field to return 0.

Note:

For conveniency, this backend provides empty binding hooks which just return success. To have hwloc still actually call OS-specific hooks, the HWLOC_-TOPOLOGY_FLAG_IS_THISSYSTEM has to be set to assert that the loaded file is really the underlying system.

5.5.3.6 int hwloc_topology_set_synthetic (hwloc_topology_t restrict *topology*, const char *restrict *description*)

Enable synthetic topology.

Gather topology information from the given `description` which should be a comma separated string of numbers describing the arity of each level. Each number may be prefixed with a type and a colon to enforce the type of a level.

Note:

For conveniency, this backend provides empty binding hooks which just return success.

5.5.3.7 int hwloc_topology_set_xml (hwloc_topology_t restrict *topology*, const char ∗restrict *xmlpath*)

Enable XML-file based topology.

Gather topology information the XML file given at `xmlpath`. This file may have been generated earlier with lstopo file.xml.

Note:

For conveniency, this backend provides empty binding hooks which just return success. To have hwloc still actually call OS-specific hooks, the HWLOC_-TOPOLOGY_FLAG_IS_THISSYSTEM has to be set to assert that the loaded file is really the underlying system.

5.6 Get some Topology Information

Defines

- #define HWLOC_TYPE_DEPTH_UNKNOWN -1

  No object of given type exists in the topology.

- #define HWLOC_TYPE_DEPTH_MULTIPLE -2

  Objects of given type exist at different depth in the topology.

Functions

- unsigned hwloc_topology_get_depth (hwloc_topology_t restrict topology)

  Get the depth of the hierachical tree of objects.

- int hwloc_get_type_depth (hwloc_topology_t topology, hwloc_obj_type_t type)

  Returns the depth of objects of type `type`.

- hwloc_obj_type_t hwloc_get_depth_type (hwloc_topology_t topology, unsigned depth)

  Returns the type of objects at depth `depth`.

- unsigned hwloc_get_nbobjs_by_depth (hwloc_topology_t topology, unsigned depth)

  Returns the width of level at depth `depth`.

- static inline int hwloc_get_nbobjs_by_type (hwloc_topology_t topology, hwloc_obj_type_t type)

  Returns the width of level type `type`.

- int hwloc_topology_is_thissystem (hwloc_topology_t restrict topology)

  Does the topology context come from this system?

5.6.1 Define Documentation

5.6.1.1 #define HWLOC_TYPE_DEPTH_MULTIPLE -2

Objects of given type exist at different depth in the topology.

5.6.1.2 #define HWLOC_TYPE_DEPTH_UNKNOWN -1

No object of given type exists in the topology.

5.6.2 Function Documentation

5.6.2.1 hwloc_obj_type_t hwloc_get_depth_type (hwloc_topology_t *topology*, unsigned *depth*)

Returns the type of objects at depth `depth`.

5.6.2.2 unsigned hwloc_get_nbobjs_by_depth (hwloc_topology_t *topology*, unsigned *depth*)

Returns the width of level at depth `depth`.

5.6.2.3 static inline int hwloc_get_nbobjs_by_type (hwloc_topology_t *topology*, hwloc_obj_type_t *type*) `[static]`

Returns the width of level type `type`.

If no object for that type exists, 0 is returned. If there are several levels with objects of that type, -1 is returned.

5.6.2.4 int hwloc_get_type_depth (hwloc_topology_t *topology*, hwloc_obj_type_t *type*)

Returns the depth of objects of type `type`.

If no object of this type is present on the underlying architecture, or if the OS doesn't provide this kind of information, the function returns HWLOC_TYPE_DEPTH_-UNKNOWN.

If type is absent but a similar type is acceptable, see also hwloc_get_type_or_below_-depth() and hwloc_get_type_or_above_depth().

5.6.2.5 unsigned hwloc_topology_get_depth (hwloc_topology_t restrict *topology*)

Get the depth of the hierachical tree of objects.

This is the depth of HWLOC_OBJ_PROC objects plus one.

5.6.2.6 int hwloc_topology_is_thissystem (hwloc_topology_t restrict *topology*)

Does the topology context come from this system?

Returns:

1 if this topology context was built using the system running this program.
0 instead (for instance if using another file-system root, a XML topology file, or a synthetic topology).

5.7 Retrieve Objects

Functions

- hwloc_obj_t hwloc_get_obj_by_depth (hwloc_topology_t topology, unsigned depth, unsigned index)

  Returns the topology object at index `index` *from depth* `depth`.

- static inline hwloc_obj_t hwloc_get_obj_by_type (hwloc_topology_t topology, hwloc_obj_type_t type, unsigned index)

  Returns the topology object at index `index` *with type* `type`.

5.7.1 Function Documentation

5.7.1.1 hwloc_obj_t hwloc_get_obj_by_depth (hwloc_topology_t *topology*, unsigned *depth*, unsigned *index*)

Returns the topology object at index `index` from depth `depth`.

5.7.1.2 static inline hwloc_obj_t hwloc_get_obj_by_type (hwloc_topology_t *topology*, hwloc_obj_type_t *type*, unsigned *index*) `[static]`

Returns the topology object at index `index` with type `type`.

If no object for that type exists, `NULL` is returned. If there are several levels with objects of that type, `NULL` is returned and ther caller may fallback to hwloc_get_obj_by_depth().

5.8 Object/String Conversion

Functions

- const char ∗ hwloc_obj_type_string (hwloc_obj_type_t type)

  Return a stringified topology object type.

- hwloc_obj_type_t hwloc_obj_type_of_string (const char ∗string)

  Return an object type from the string.

- int hwloc_obj_snprintf (char ∗restrict string, size_t size, hwloc_topology_t topology, hwloc_obj_t obj, const char ∗restrict indexprefix, int verbose)

  Stringify a given topology object into a human-readable form.

- int hwloc_obj_cpuset_snprintf (char ∗restrict str, size_t size, size_t nobj, const hwloc_obj_t ∗restrict objs)

  Stringify the cpuset containing a set of objects.

5.8.1 Function Documentation

5.8.1.1 int hwloc_obj_cpuset_snprintf (char ∗restrict *str*, size_t *size*, size_t *nobj*, const hwloc_obj_t ∗restrict *objs*)

Stringify the cpuset containing a set of objects.

Returns:

how many characters were actually written (not including the ending \0).

5.8.1.2 int hwloc_obj_snprintf (char ∗restrict *string*, size_t *size*, hwloc_topology_t *topology*, hwloc_obj_t *obj*, const char ∗restrict *indexprefix*, int *verbose*)

Stringify a given topology object into a human-readable form.

Returns:

how many characters were actually written (not including the ending \0).

5.8.1.3 hwloc_obj_type_t hwloc_obj_type_of_string (const char ∗ *string*)

Return an object type from the string.

5.8.1.4 const char∗ hwloc_obj_type_string (hwloc_obj_type_t *type*)

Return a stringified topology object type.

5.9 Binding

Enumerations

- enum hwloc_cpubind_policy_t { HWLOC_CPUBIND_PROCESS = (1<<0), HWLOC_CPUBIND_THREAD = (1<<1), HWLOC_CPUBIND_STRICT = (1<<2) }

  Process/Thread binding policy.

Functions

- int hwloc_set_cpubind (hwloc_topology_t topology, const hwloc_cpuset_t set, int policy)

  Bind current process or thread on cpus given in cpuset `set`*.*

- int hwloc_set_proc_cpubind (hwloc_topology_t topology, hwloc_pid_t pid, const hwloc_cpuset_t set, int policy)

  Bind a process `pid` *on cpus given in cpuset* `set`*.*

- int hwloc_set_thread_cpubind (hwloc_topology_t topology, hwloc_thread_t tid, const hwloc_cpuset_t set, int policy)

  Bind a thread `tid` *on cpus given in cpuset* `set`*.*

5.9.1 Detailed Description

It is often useful to call hwloc_cpuset_singlify() first so that a single CPU remains in the set. This way, the process will not even migrate between different CPUs. Some OSes also only support that kind of binding.

Note:

Some OSes do not provide all ways to bind processes, threads, etc and the corresponding binding functions may fail. ENOSYS is returned when it is not possible to bind the requested kind of object processes/threads). EXDEV is returned when the requested cpuset can not be enforced (e.g. some systems only allow one CPU, and some other systems only allow one NUMA node)

The most portable version that should be preferred over the others, whenever possible, is

```
hwloc_set_cpubind(topology, set, 0),
```

as it just binds the current program, assuming it is monothread, or

```
hwloc_set_cpubind(topology, set, HWLOC_CPUBIND_THREAD),
```

which binds the current thread of the current program (which may be multithreaded).

Note:

To unbind, just call the binding function with either a full cpuset or a cpuset equal to the system cpuset.

5.9.2 Enumeration Type Documentation

5.9.2.1 enum hwloc_cpubind_policy_t

Process/Thread binding policy.

These flags can be used to refine the binding policy.

The default (0) is to bind the current process, assumed to be mono-thread, in a non-strict way. This is the most portable way to bind as all OSes usually provide it.

Note:

Depending on OSes and implementations, strict binding (i.e. the thread/process will really never be scheduled outside of the cpuset) may not be possible, not be allowed, only used as a hint when no load balancing is needed, etc. If strict binding is required, the strict flag should be set, and the function will fail if strict binding is not possible or allowed.

Enumerator:

HWLOC_CPUBIND_PROCESS Bind all threads of the current multithreaded process. This may not be supported by some OSes (e.g. Linux).

HWLOC_CPUBIND_THREAD Bind current thread of current process.

HWLOC_CPUBIND_STRICT Request for strict binding from the OS Note that strict binding may not be allowed for administrative reasons, and the binding function will fail in that case.

5.9.3 Function Documentation

5.9.3.1 int hwloc_set_cpubind (hwloc_topology_t *topology*, const hwloc_cpuset_t *set*, int *policy*)

Bind current process or thread on cpus given in cpuset `set`.

5.9.3.2 int hwloc_set_proc_cpubind (hwloc_topology_t *topology*, hwloc_pid_t *pid*, const hwloc_cpuset_t *set*, int *policy*)

Bind a process `pid` on cpus given in cpuset `set`.

Note:

hwloc_pid_t is pid_t on unix platforms, and HANDLE on native Windows platforms
HWLOC_CPUBIND_THREAD can not be used in `policy`.

5.9.3.3 int hwloc_set_thread_cpubind (hwloc_topology_t *topology*, hwloc_thread_t *tid*, const hwloc_cpuset_t *set*, int *policy*)

Bind a thread `tid` on cpus given in cpuset `set`.

Note:

hwloc_thread_t is pthread_t on unix platforms, and HANDLE on native Windows platforms
HWLOC_CPUBIND_PROCESS can not be used in `policy`.

5.10 Object Type Helpers

Functions

- static inline unsigned hwloc_get_type_or_below_depth (hwloc_topology_-t topology, hwloc_obj_type_t type)

  Returns the depth of objects of type `type` *or below.*

- static inline unsigned hwloc_get_type_or_above_depth (hwloc_topology_-t topology, hwloc_obj_type_t type)

  Returns the depth of objects of type `type` *or above.*

5.10.1 Function Documentation

5.10.1.1 static inline unsigned hwloc_get_type_or_above_depth (hwloc_topology_t *topology*, hwloc_obj_type_t *type*) `[static]`

Returns the depth of objects of type `type` or above.

If no object of this type is present on the underlying architecture, the function returns the depth of the first "present" object typically containing `type`.

5.10.1.2 static inline unsigned hwloc_get_type_or_below_depth (hwloc_topology_t *topology*, hwloc_obj_type_t *type*) `[static]`

Returns the depth of objects of type `type` or below.

If no object of this type is present on the underlying architecture, the function returns the depth of the first "present" object typically found inside `type`.

5.11 Basic Traversal Helpers

Functions

- static inline hwloc_obj_t hwloc_get_system_obj (hwloc_topology_t topology)

  Returns the top-object of the topology-tree. Its type is HWLOC_OBJ_SYSTEM.

- static inline hwloc_obj_t hwloc_get_next_obj_by_depth (hwloc_topology_t topology, unsigned depth, hwloc_obj_t prev)

  Returns the next object at depth `depth`.

- static inline hwloc_obj_t hwloc_get_next_obj_by_type (hwloc_topology_t topology, hwloc_obj_type_t type, hwloc_obj_t prev)

  Returns the next object of type `type`.

- static inline hwloc_obj_t hwloc_get_next_child (hwloc_topology_t topology, hwloc_obj_t father, hwloc_obj_t prev)

  Return the next child.

- static inline hwloc_obj_t hwloc_get_common_ancestor_obj (hwloc_topology_t topology, hwloc_obj_t obj1, hwloc_obj_t obj2)

  Returns the common father object to objects lvl1 and lvl2.

- static inline int hwloc_obj_is_in_subtree (hwloc_topology_t topology, hwloc_obj_t obj, hwloc_obj_t subtree_root)

  Returns true if _obj_ is inside the subtree beginning with `subtree_root`.

5.11.1 Function Documentation

5.11.1.1 static inline hwloc_obj_t hwloc_get_common_ancestor_obj (hwloc_topology_t *topology*, hwloc_obj_t *obj1*, hwloc_obj_t *obj2*) `[static]`

Returns the common father object to objects lvl1 and lvl2.

5.11.1.2 static inline hwloc_obj_t hwloc_get_next_child (hwloc_topology_t *topology*, hwloc_obj_t *father*, hwloc_obj_t *prev*) `[static]`

Return the next child.

If `prev` is `NULL`, return the first child.

5.11.1.3 static inline hwloc_obj_t hwloc_get_next_obj_by_depth (hwloc_topology_t *topology*, unsigned *depth*, hwloc_obj_t *prev*) `[static]`

Returns the next object at depth `depth`.

If `prev` is `NULL`, return the first object at depth `depth`.

5.11.1.4 static inline hwloc_obj_t hwloc_get_next_obj_by_type (hwloc_topology_t *topology*, hwloc_obj_type_t *type*, hwloc_obj_t *prev*) `[static]`

Returns the next object of type `type`.

If `prev` is `NULL`, return the first object at type `type`. If there are multiple or no depth for given type, return `NULL` and let the caller fallback to hwloc_get_next_obj_by_-depth().

5.11.1.5 static inline hwloc_obj_t hwloc_get_system_obj (hwloc_topology_t *topology*) `[static]`

Returns the top-object of the topology-tree. Its type is HWLOC_OBJ_SYSTEM.

5.11.1.6 static inline int hwloc_obj_is_in_subtree (hwloc_topology_t *topology*, hwloc_obj_t *obj*, hwloc_obj_t *subtree_root*) `[static]`

Returns true if _obj_ is inside the subtree beginning with `subtree_root`.

5.12 Finding Objects Inside a CPU set

Functions

- int hwloc_get_largest_objs_inside_cpuset (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_t ∗restrict objs, int max)

  Get the set of largest objects covering exactly a given cpuset `set`.

- static inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_cpuset_t set, unsigned depth, hwloc_obj_t prev)

  Return the next object at depth `depth` *included in CPU set* `set`.

- static inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_type_t type, hwloc_obj_t prev)

  Return the next object of type `type` *included in CPU set* `set`.

- static inline hwloc_obj_t hwloc_get_obj_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_cpuset_t set, unsigned depth, unsigned index)

  Return the `index` *-th object at depth* `depth` *included in CPU set* `set`.

- static inline hwloc_obj_t hwloc_get_obj_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_type_t type, unsigned index)

  Return the `index` *-th object of type* `type` *included in CPU set* `set`.

- static inline unsigned hwloc_get_nbobjs_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_cpuset_t set, unsigned depth)

  Return the number of objects at depth `depth` *included in CPU set* `set`.

- static inline int hwloc_get_nbobjs_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_type_t type)

  Return the number of objects of type `type` *included in CPU set* `set`.

5.12.1 Function Documentation

5.12.1.1 int hwloc_get_largest_objs_inside_cpuset (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_t ∗restrict *objs*, int *max*)

Get the set of largest objects covering exactly a given cpuset `set`.

Returns:

the number of objects returned in `objs`.

5.12.1.2 static inline unsigned hwloc_get_nbobjs_inside_cpuset_by_depth (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, unsigned *depth*) `[static]`

Return the number of objects at depth `depth` included in CPU set `set`.

5.12.1.3 static inline int hwloc_get_nbobjs_inside_cpuset_by_type (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_type_t *type*) `[static]`

Return the number of objects of type `type` included in CPU set `set`.

If no object for that type exists inside CPU set `set`, 0 is returned. If there are several levels with objects of that type inside CPU set `set`, -1 is returned.

5.12.1.4 static inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_depth (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, unsigned *depth*, hwloc_obj_t *prev*) `[static]`

Return the next object at depth `depth` included in CPU set `set`.

If `prev` is `NULL`, return the first object at depth `depth` included in `set`. The next invokation should pass the previous return value in `prev` so as to obtain the next object in `set`.

5.12.1.5 static inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_type (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_type_t *type*, hwloc_obj_t *prev*) `[static]`

Return the next object of type `type` included in CPU set `set`.

If there are multiple or no depth for given type, return `NULL` and let the caller fallback to hwloc_get_next_obj_inside_cpuset_by_depth().

5.12.1.6 static inline hwloc_obj_t hwloc_get_obj_inside_cpuset_by_depth (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, unsigned *depth*, unsigned *index*) `[static]`

Return the `index` -th object at depth `depth` included in CPU set `set`.

5.12.1.7 static inline hwloc_obj_t hwloc_get_obj_inside_cpuset_by_type (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_type_t *type*, unsigned *index*) `[static]`

Return the `index` -th object of type `type` included in CPU set `set`.

If there are multiple or no depth for given type, return `NULL` and let the caller fallback to hwloc_get_obj_inside_cpuset_by_depth().

5.13 Finding a single Object covering at least CPU set

Functions

- static hwloc_obj_t hwloc_get_child_covering_cpuset (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_t father)

  Get the child covering at least CPU set `set`*.*

- static hwloc_obj_t hwloc_get_obj_covering_cpuset (hwloc_topology_t topology, hwloc_cpuset_t set)

  Get the lowest object covering at least CPU set `set`*.*

5.13.1 Function Documentation

5.13.1.1 static hwloc_obj_t hwloc_get_child_covering_cpuset (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_t *father*) `[inline, static]`

Get the child covering at least CPU set `set`.

Returns:

`NULL` if no child matches.

5.13.1.2 static hwloc_obj_t hwloc_get_obj_covering_cpuset (hwloc_topology_t *topology*, hwloc_cpuset_t *set*) `[inline, static]`

Get the lowest object covering at least CPU set `set`.

Returns:

`NULL` if no object matches.

5.14 Finding a set of similar Objects covering at least a CPU set

Functions

- static inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_depth (hwloc_topology_t topology, hwloc_cpuset_t set, unsigned depth, hwloc_obj_t prev)

  Iterate through same-depth objects covering at least CPU set `set`.

- static inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_type (hwloc_topology_t topology, hwloc_cpuset_t set, hwloc_obj_type_t type, hwloc_obj_t prev)

  Iterate through same-type objects covering at least CPU set `set`.

5.14.1 Function Documentation

5.14.1.1 static inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_depth (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, unsigned *depth*, hwloc_obj_t *prev*) `[static]`

Iterate through same-depth objects covering at least CPU set `set`.

If object `prev` is `NULL`, return the first object at depth `depth` covering at least part of CPU set `set`. The next invokation should pass the previous return value in `prev` so as to obtain the next object covering at least another part of `set`.

5.14.1.2 static inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_type (hwloc_topology_t *topology*, hwloc_cpuset_t *set*, hwloc_obj_type_t *type*, hwloc_obj_t *prev*) `[static]`

Iterate through same-type objects covering at least CPU set `set`.

If object `prev` is `NULL`, return the first object of type `type` covering at least part of CPU set `set`. The next invokation should pass the previous return value in `prev` so as to obtain the next object of type `type` covering at least another part of `set`.

If there are no or multiple depths for type `type`, `NULL` is returned. The caller may fallback to hwloc_get_next_obj_covering_cpuset_by_depth() for each depth.

5.15 Cache-specific Finding Helpers

Functions

- static inline hwloc_obj_t hwloc_get_cache_covering_cpuset (hwloc_topology_t topology, hwloc_cpuset_t set)

  Get the first cache covering a cpuset `set`*.*

- static inline hwloc_obj_t hwloc_get_cache_covering_obj (hwloc_topology_t topology, hwloc_obj_t obj)

  Get the first cache shared between an object and somebody else.

5.15.1 Function Documentation

5.15.1.1 static inline hwloc_obj_t hwloc_get_cache_covering_cpuset (hwloc_topology_t *topology*, hwloc_cpuset_t *set*) `[static]`

Get the first cache covering a cpuset `set`.

Returns:

`NULL` if no cache matches

5.15.1.2 static inline hwloc_obj_t hwloc_get_cache_covering_obj (hwloc_topology_t *topology*, hwloc_obj_t *obj*) `[static]`

Get the first cache shared between an object and somebody else.

Returns:

`NULL` if no cache matches

5.16 Advanced Traversal Helpers

Functions

- int hwloc_get_closest_objs (hwloc_topology_t topology, hwloc_obj_t src, hwloc_obj_t ∗restrict objs, int max)

  Do a depth-first traversal of the topology to find and sort.

5.16.1 Function Documentation

5.16.1.1 int hwloc_get_closest_objs (hwloc_topology_t *topology*, hwloc_obj_t *src*, hwloc_obj_t ∗restrict *objs*, int *max*)

Do a depth-first traversal of the topology to find and sort.

all objects that are at the same depth than `src`. Report in `objs` up to `max` physically closest ones to `src`.

Returns:

the number of objects returned in `objs`.

5.17 Binding Helpers

Functions

- static inline void hwloc_distribute (hwloc_topology_t topology, hwloc_obj_-t root, hwloc_cpuset_t *cpuset, int n)

  Distribute `n` *items over the topology under* `root`.

5.17.1 Function Documentation

5.17.1.1 static inline void hwloc_distribute (hwloc_topology_t *topology*, hwloc_obj_t *root*, hwloc_cpuset_t * *cpuset*, int *n*) `[static]`

Distribute `n` items over the topology under `root`.

Array `cpuset` will be filled with `n` cpusets distributed linearly over the topology under `root` .

This is typically useful when an application wants to distribute `n` threads over a machine, giving each of them as much private cache as possible and keeping them locally in number order.

The caller may typicall want to additionally call hwloc_cpuset_singlify() before binding a thread, so that it doesn't move at all.

5.18 The Cpuset API

Defines

- #define hwloc_cpuset_foreach_begin(cpu, set)

  Loop macro iterating on CPU set `set`*.*

- #define hwloc_cpuset_foreach_end() }

  End of loop.

Typedefs

- typedef struct hwloc_opaque_cpuset_s * hwloc_cpuset_t

  Set of CPUs represented as an opaque pointer to an internal bitmask.

Functions

- hwloc_cpuset_t hwloc_cpuset_alloc (void)

  Allocate a new empty CPU set.

- void hwloc_cpuset_free (hwloc_cpuset_t set)

  Free CPU set `set`*.*

- hwloc_cpuset_t hwloc_cpuset_dup (hwloc_cpuset_t set)

  Duplicate CPU set `set` *by allocating a new CPU set and copying its contents.*

- void hwloc_cpuset_copy (hwloc_cpuset_t dst, hwloc_cpuset_t src)

  Copy the contents of CPU set `src` *into the already allocated CPU set* `dst`*.*

- int hwloc_cpuset_snprintf (char *restrict buf, size_t buflen, hwloc_cpuset_t set)

  Stringify a cpuset.

- int hwloc_cpuset_asprintf (char **strp, hwloc_cpuset_t set)

  Stringify a cpuset into a newly allocated string.

- hwloc_cpuset_t hwloc_cpuset_from_string (const char *restrict string)

  Parse a cpuset string.

- void hwloc_cpuset_zero (hwloc_cpuset_t set)

Primitives & macros for building, modifying and consulting "sets" of cpus.

- void hwloc_cpuset_fill (hwloc_cpuset_t set)

  Fill CPU set `set`.

- void hwloc_cpuset_from_ulong (hwloc_cpuset_t set, unsigned long mask)

  Setup CPU set `set` *from unsigned long* `mask`.

- void hwloc_cpuset_from_ith_ulong (hwloc_cpuset_t set, int i, unsigned long mask)

  Setup CPU set `set` *from unsigned long* `mask` *used as* `i` *-th subset.*

- unsigned long hwloc_cpuset_to_ulong (hwloc_cpuset_t set)

  Convert the beginning part of CPU set `set` *into unsigned long* `mask`.

- unsigned long hwloc_cpuset_to_ith_ulong (hwloc_cpuset_t set, int i)

  Convert the `i` *-th subset of CPU set* `set` *into unsigned long mask.*

- void hwloc_cpuset_cpu (hwloc_cpuset_t set, unsigned cpu)

  Clear CPU set `set` *and set CPU* `cpu`.

- void hwloc_cpuset_all_but_cpu (hwloc_cpuset_t set, unsigned cpu)

  Clear CPU set `set` *and set all but the CPU* `cpu`.

- void hwloc_cpuset_set (hwloc_cpuset_t set, unsigned cpu)

  Add CPU `cpu` *in CPU set* `set`.

- void hwloc_cpuset_set_range (hwloc_cpuset_t set, unsigned begincpu, unsigned endcpu)

  Add CPUs from `begincpu` *to* `endcpu` *in CPU set* `set`.

- void hwloc_cpuset_clr (hwloc_cpuset_t set, unsigned cpu)

  Remove CPU `cpu` *from CPU set* `set`.

- int hwloc_cpuset_isset (hwloc_cpuset_t set, unsigned cpu)

  Test whether CPU `cpu` *is part of set* `set`.

- int hwloc_cpuset_iszero (hwloc_cpuset_t set)

  Test whether set `set` *is zero.*

- int hwloc_cpuset_isfull (hwloc_cpuset_t set)

  Test whether set `set` *is full.*

- int hwloc_cpuset_isequal (hwloc_cpuset_t set1, hwloc_cpuset_t set2)

  Test whether set `set1` *is equal to set* `set2`*.*

- int hwloc_cpuset_intersects (hwloc_cpuset_t set1, hwloc_cpuset_t set2)

  Test whether sets `set1` *and* `set2` *intersects.*

- int hwloc_cpuset_isincluded (hwloc_cpuset_t sub_set, hwloc_cpuset_t super_-set)

  Test whether set `sub_set` *is part of set* `super_set`*.*

- void hwloc_cpuset_orset (hwloc_cpuset_t set, hwloc_cpuset_t modifier_set)

  Or set `modifier_set` *into set* `set`*.*

- void hwloc_cpuset_andset (hwloc_cpuset_t set, hwloc_cpuset_t modifier_set)

  And set `modifier_set` *into set* `set`*.*

- void hwloc_cpuset_clearset (hwloc_cpuset_t set, hwloc_cpuset_t modifier_-set)

  Clear set `modifier_set` *out of set* `set`*.*

- void hwloc_cpuset_xorset (hwloc_cpuset_t set, hwloc_cpuset_t modifier_set)

  Xor set `set` *with set* `modifier_set`*.*

- int hwloc_cpuset_first (hwloc_cpuset_t set)

  Compute the first CPU (least significant bit) in CPU set `set`*.*

- int hwloc_cpuset_last (hwloc_cpuset_t set)

  Compute the last CPU (most significant bit) in CPU set `set`*.*

- void hwloc_cpuset_singlify (hwloc_cpuset_t set)

  Keep a single CPU among those set in CPU set `set`*.*

- int hwloc_cpuset_compar_first (hwloc_cpuset_t set1, hwloc_cpuset_t set2)

  Compar CPU sets `set1` *and* `set2` *using their first set bit.*

- int hwloc_cpuset_compar (hwloc_cpuset_t set1, hwloc_cpuset_t set2)

  Compar CPU sets `set1` *and* `set2` *using their last bits.*

- int hwloc_cpuset_weight (hwloc_cpuset_t set)

  Compute the weight of CPU set `set`*.*

5.18.1 Detailed Description

For use in hwloc itself, a hwloc_cpuset_t represents a set of logical processors.

Note:

cpusets are indexed by OS logical processor number.

5.18.2 Define Documentation

5.18.2.1 #define hwloc_cpuset_foreach_begin(cpu, set)

Value:

```
for (cpu = 0; cpu < HWLOC_NBMAXCPUS; cpu++) \
                if (hwloc_cpuset_isset(set, cpu)) {
```

Loop macro iterating on CPU set `set`.

It yields on each cpu that is member of the set. It uses variables `set` (the cpu set) and `cpu` (the loop variable)

5.18.2.2 #define hwloc_cpuset_foreach_end() }

End of loop.

See also:

hwloc_cpuset_foreach_begin

5.18.3 Typedef Documentation

5.18.3.1 typedef struct hwloc_opaque_cpuset_s* hwloc_cpuset_t

Set of CPUs represented as an opaque pointer to an internal bitmask.

5.18.4 Function Documentation

5.18.4.1 void hwloc_cpuset_all_but_cpu (hwloc_cpuset_t *set*, unsigned *cpu*)

Clear CPU set `set` and set all but the CPU `cpu`.

5.18.4.2 hwloc_cpuset_t hwloc_cpuset_alloc (void)

Allocate a new empty CPU set.

5.18.4.3 void hwloc_cpuset_andset (hwloc_cpuset_t *set*, hwloc_cpuset_t *modifier_set*)

And set `modifier_set` into set `set`.

5.18.4.4 int hwloc_cpuset_asprintf (char ** *strp*, hwloc_cpuset_t *set*)

Stringify a cpuset into a newly allocated string.

Returns:

the number of character that were actually written (not including the ending \0).

5.18.4.5 void hwloc_cpuset_clearset (hwloc_cpuset_t *set*, hwloc_cpuset_t *modifier_set*)

Clear set `modifier_set` out of set `set`.

5.18.4.6 void hwloc_cpuset_clr (hwloc_cpuset_t *set*, unsigned *cpu*)

Remove CPU `cpu` from CPU set `set`.

5.18.4.7 int hwloc_cpuset_compar (hwloc_cpuset_t *set1*, hwloc_cpuset_t *set2*)

Compar CPU sets `set1` and `set2` using their last bits.

Higher most significant bit is higher. The empty CPU set is considered lower than anything.

5.18.4.8 int hwloc_cpuset_compar_first (hwloc_cpuset_t *set1*, hwloc_cpuset_t *set2*)

Compar CPU sets `set1` and `set2` using their first set bit.

Smaller least significant bit is smaller. The empty CPU set is considered higher than anything.

5.18.4.9 void hwloc_cpuset_copy (hwloc_cpuset_t *dst*, hwloc_cpuset_t *src*)

Copy the contents of CPU set `src` into the already allocated CPU set `dst`.

5.18.4.10 void hwloc_cpuset_cpu (hwloc_cpuset_t *set*, unsigned *cpu*)

Clear CPU set `set` and set CPU `cpu`.

5.18.4.11 hwloc_cpuset_t hwloc_cpuset_dup (hwloc_cpuset_t *set*)

Duplicate CPU set `set` by allocating a new CPU set and copying its contents.

5.18.4.12 void hwloc_cpuset_fill (hwloc_cpuset_t *set*)

Fill CPU set `set`.

5.18.4.13 int hwloc_cpuset_first (hwloc_cpuset_t *set*)

Compute the first CPU (least significant bit) in CPU set `set`.

5.18.4.14 void hwloc_cpuset_free (hwloc_cpuset_t *set*)

Free CPU set `set`.

5.18.4.15 void hwloc_cpuset_from_ith_ulong (hwloc_cpuset_t *set*, int *i*, unsigned long *mask*)

Setup CPU set `set` from unsigned long `mask` used as `i` -th subset.

5.18.4.16 hwloc_cpuset_t hwloc_cpuset_from_string (const char ∗restrict *string*)

Parse a cpuset string.

Must start and end with a digit.

5.18.4.17 void hwloc_cpuset_from_ulong (hwloc_cpuset_t *set*, unsigned long *mask*)

Setup CPU set `set` from unsigned long `mask`.

5.18.4.18 int hwloc_cpuset_intersects (hwloc_cpuset_t *set1*, hwloc_cpuset_t *set2*)

Test whether sets `set1` and `set2` intersects.

5.18.4.19 int hwloc_cpuset_isequal (hwloc_cpuset_t *set1*, hwloc_cpuset_t *set2*)

Test whether set `set1` is equal to set `set2`.

5.18.4.20 int hwloc_cpuset_isfull (hwloc_cpuset_t *set*)

Test whether set `set` is full.

5.18.4.21 int hwloc_cpuset_isincluded (hwloc_cpuset_t *sub_set*, hwloc_cpuset_t *super_set*)

Test whether set `sub_set` is part of set `super_set`.

5.18.4.22 int hwloc_cpuset_isset (hwloc_cpuset_t *set*, unsigned *cpu*)

Test whether CPU `cpu` is part of set `set`.

5.18.4.23 int hwloc_cpuset_iszero (hwloc_cpuset_t *set*)

Test whether set `set` is zero.

5.18.4.24 int hwloc_cpuset_last (hwloc_cpuset_t *set*)

Compute the last CPU (most significant bit) in CPU set `set`.

5.18.4.25 void hwloc_cpuset_orset (hwloc_cpuset_t *set*, hwloc_cpuset_t *modifier_set*)

Or set `modifier_set` into set `set`.

5.18.4.26 void hwloc_cpuset_set (hwloc_cpuset_t *set*, unsigned *cpu*)

Add CPU `cpu` in CPU set `set`.

5.18.4.27 void hwloc_cpuset_set_range (hwloc_cpuset_t *set*, unsigned *begincpu*, unsigned *endcpu*)

Add CPUs from `begincpu` to `endcpu` in CPU set `set`.

5.18.4.28 void hwloc_cpuset_singlify (hwloc_cpuset_t *set*)

Keep a single CPU among those set in CPU set `set`.

Might be used before binding so that the process does not have a chance of migrating between multiple logical CPUs in the original mask.

5.18.4.29 int hwloc_cpuset_snprintf (char ∗restrict *buf*, size_t *buflen*, hwloc_cpuset_t *set*)

Stringify a cpuset.

Up to `buflen` characters may be written in buffer `buf`.

Returns:

the number of character that were actually written if not truncating, or that would have been written (not including the ending \0).

5.18.4.30 unsigned long hwloc_cpuset_to_ith_ulong (hwloc_cpuset_t *set*, int *i*)

Convert the `i` -th subset of CPU set `set` into unsigned long mask.

5.18.4.31 unsigned long hwloc_cpuset_to_ulong (hwloc_cpuset_t *set*)

Convert the beginning part of CPU set `set` into unsigned long `mask`.

5.18.4.32 int hwloc_cpuset_weight (hwloc_cpuset_t *set*)

Compute the weight of CPU set `set`.

5.18.4.33 void hwloc_cpuset_xorset (hwloc_cpuset_t *set*, hwloc_cpuset_t *modifier_set*)

Xor set `set` with set `modifier_set`.

5.18.4.34 void hwloc_cpuset_zero (hwloc_cpuset_t *set*)

Primitives & macros for building, modifying and consulting "sets" of cpus.

Empty CPU set `set`

5.19 Helpers for manipulating glibc sched affinity

Functions

- static inline void hwloc_cpuset_to_glibc_sched_affinity (hwloc_topology_t topology, hwloc_cpuset_t hwlocset, cpu_set_t ∗schedset, size_t schedsetsize)

  Convert hwloc CPU set `toposet` *into glibc sched affinity CPU set* `schedset`.

- static inline hwloc_cpuset_t hwloc_cpuset_from_glibc_sched_affinity (hwloc_topology_t topology, const cpu_set_t ∗schedset, size_t schedsetsize)

  Convert glibc sched affinity CPU set `schedset` *into hwloc CPU set.*

5.19.1 Function Documentation

5.19.1.1 static inline hwloc_cpuset_t hwloc_cpuset_from_glibc_sched_affinity (hwloc_topology_t *topology*, const cpu_set_t ∗ *schedset*, size_t *schedsetsize*) `[static]`

Convert glibc sched affinity CPU set `schedset` into hwloc CPU set.

This function may be used before calling sched_setaffinity or any other function that takes a cpu_set_t as input parameter.

`schedsetsize` should be sizeof(cpu_set_t) unless `schedset` was dynamically allocated with CPU_ALLOC

5.19.1.2 static inline void hwloc_cpuset_to_glibc_sched_affinity (hwloc_topology_t *topology*, hwloc_cpuset_t *hwlocset*, cpu_set_t ∗ *schedset*, size_t *schedsetsize*) `[static]`

Convert hwloc CPU set `toposet` into glibc sched affinity CPU set `schedset`.

This function may be used before calling sched_setaffinity or any other function that takes a cpu_set_t as input parameter.

`schedsetsize` should be sizeof(cpu_set_t) unless `schedset` was dynamically allocated with CPU_ALLOC

5.20 Helpers for manipulating Linux libnuma unsigned long masks

Functions

- static inline void hwloc_cpuset_to_linux_libnuma_ulongs (hwloc_topology_t topology, hwloc_cpuset_t cpuset, unsigned long ∗mask, unsigned long ∗maxnode)

  Convert hwloc CPU set `cpuset` *into the array of unsigned long* `mask`*.*

- static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_ulongs (hwloc_topology_t topology, const unsigned long ∗mask, unsigned long maxnode)

  Convert the array of unsigned long `mask` *into hwloc CPU set.*

5.20.1 Function Documentation

5.20.1.1 static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_ulongs (hwloc_topology_t *topology*, const unsigned long ∗ *mask*, unsigned long *maxnode*) `[static]`

Convert the array of unsigned long `mask` into hwloc CPU set.

`mask` is a array of unsigned long that will be read. `maxnode` contains the maximal node number that may be read in `mask`.

This function may be used after calling get_mempolicy or any other function that takes an array of unsigned long as output parameter (and possibly a maximal node number as input parameter).

5.20.1.2 static inline void hwloc_cpuset_to_linux_libnuma_ulongs (hwloc_topology_t *topology*, hwloc_cpuset_t *cpuset*, unsigned long ∗ *mask*, unsigned long ∗ *maxnode*) `[static]`

Convert hwloc CPU set `cpuset` into the array of unsigned long `mask`.

`mask` is the array of unsigned long that will be filled. `maxnode` contains the maximal node number that may be stored in `mask`. `maxnode` will be set to the maximal node number that was found, plus one.

This function may be used before calling set_mempolicy, mbind, migrate_pages or any other function that takes an array of unsigned long and a maximal node number as input parameter.

5.21 Helpers for manipulating Linux libnuma bitmask

Functions

- static inline struct bitmask ∗ hwloc_cpuset_to_linux_libnuma_bitmask (hwloc_-topology_t topology, hwloc_cpuset_t cpuset)

  Convert hwloc CPU set `cpuset` *into the returned libnuma bitmask.*

- static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_bitmask (hwloc_topology_t topology, const struct bitmask ∗bitmask)

  Convert libnuma bitmask `bitmask` *into hwloc CPU set* `cpuset`.

5.21.1 Function Documentation

5.21.1.1 static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_-bitmask (hwloc_topology_t *topology*, const struct bitmask ∗ *bitmask*) `[static]`

Convert libnuma bitmask `bitmask` into hwloc CPU set `cpuset`.

This function may be used after calling many numa_ functions that use a struct bitmask as an output parameter.

5.21.1.2 static inline struct bitmask∗ hwloc_cpuset_to_linux_libnuma_bitmask (hwloc_topology_t *topology*, hwloc_cpuset_t *cpuset*) `[static, read]`

Convert hwloc CPU set `cpuset` into the returned libnuma bitmask.

The returned bitmask should later be freed with numa_bitmask_free.

This function may be used before calling many numa_ functions that use a struct bit-mask as an input parameter.

5.22 Helpers for manipulating Linux libnuma nodemask_t

Functions

- static inline void hwloc_cpuset_to_linux_libnuma_nodemask (hwloc_topology_t topology, hwloc_cpuset_t cpuset, nodemask_t ∗nodemask)

  Convert hwloc CPU set `cpuset` *into libnuma nodemask* `nodemask`.

- static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_nodemask (hwloc_topology_t topology, const nodemask_t ∗nodemask)

  Convert libnuma nodemask `nodemask` *into hwloc CPU set* `cpuset`.

5.22.1 Function Documentation

5.22.1.1 static inline hwloc_cpuset_t hwloc_cpuset_from_linux_libnuma_nodemask (hwloc_topology_t *topology*, const nodemask_t ∗ *nodemask*) `[static]`

Convert libnuma nodemask `nodemask` into hwloc CPU set `cpuset`.

This function may be used before calling some old libnuma functions that use a nodemask_t as an output parameter.

5.22.1.2 static inline void hwloc_cpuset_to_linux_libnuma_nodemask (hwloc_topology_t *topology*, hwloc_cpuset_t *cpuset*, nodemask_t ∗ *nodemask*) `[static]`

Convert hwloc CPU set `cpuset` into libnuma nodemask `nodemask`.

This function may be used before calling some old libnuma functions that use a nodemask_t as an input parameter.

Chapter 6

Data Structure Documentation

6.1 hwloc_obj_attr_u::hwloc_cache_attr_s Struct Reference

Cache-specific Object Attributes.

`#include <hwloc.h>`

Data Fields

- unsigned long memory_kB

  Size of cache.

- unsigned depth

  Depth of cache.

6.1.1 Detailed Description

Cache-specific Object Attributes.

6.1.2 Field Documentation

6.1.2.1 unsigned hwloc_obj_attr_u::hwloc_cache_attr_s::depth

Depth of cache.

6.1.2.2 unsigned long hwloc_obj_attr_u::hwloc_cache_attr_s::memory_kB

Size of cache.

The documentation for this struct was generated from the following file:

- hwloc.h

6.2 hwloc_obj_attr_u::hwloc_machine_attr_s Struct Reference

Machine-specific Object Attributes.

`#include <hwloc.h>`

Data Fields

- char ∗ dmi_board_vendor

  DMI board vendor name.

- char ∗ dmi_board_name

  DMI board model name.

- unsigned long memory_kB

  Size of memory node.

- unsigned long huge_page_free

  Number of available huge pages.

- unsigned long huge_page_size_kB

  Size of huge pages.

6.2.1 Detailed Description

Machine-specific Object Attributes.

6.2.2 Field Documentation

6.2.2.1 char∗ hwloc_obj_attr_u::hwloc_machine_attr_s::dmi_board_name

DMI board model name.

6.2.2.2 char∗ hwloc_obj_attr_u::hwloc_machine_attr_s::dmi_board_vendor

DMI board vendor name.

6.2.2.3 unsigned long hwloc_obj_attr_u::hwloc_machine_attr_s::huge_page_-free

Number of available huge pages.

6.2.2.4 unsigned long hwloc_obj_attr_u::hwloc_machine_attr_s::huge_page_-size_kB

Size of huge pages.

6.2.2.5 unsigned long hwloc_obj_attr_u::hwloc_machine_attr_s::memory_kB

Size of memory node.

The documentation for this struct was generated from the following file:

- hwloc.h

6.3 hwloc_obj_attr_u::hwloc_memory_attr_s Struct Reference

Node-specific Object Attributes.

```
#include <hwloc.h>
```

Data Fields

- unsigned long memory_kB

  Size of memory node.

- unsigned long huge_page_free

  Number of available huge pages.

6.3.1 Detailed Description

Node-specific Object Attributes.

6.3.2 Field Documentation

6.3.2.1 unsigned long hwloc_obj_attr_u::hwloc_memory_attr_s::huge_page_free

Number of available huge pages.

6.3.2.2 unsigned long hwloc_obj_attr_u::hwloc_memory_attr_s::memory_kB

Size of memory node.

The documentation for this struct was generated from the following file:

- hwloc.h

6.4 hwloc_obj_attr_u::hwloc_misc_attr_s Struct Reference

Misc-specific Object Attributes.

```
#include <hwloc.h>
```

Data Fields

- unsigned depth

  Depth of misc object.

6.4.1 Detailed Description

Misc-specific Object Attributes.

6.4.2 Field Documentation

6.4.2.1 unsigned hwloc_obj_attr_u::hwloc_misc_attr_s::depth

Depth of misc object.

The documentation for this struct was generated from the following file:

- hwloc.h

6.5 hwloc_obj Struct Reference

Structure of a topology object.

`#include <hwloc.h>`

Data Fields

- hwloc_obj_type_t type

  Type of object.

- signed os_index

  OS-provided physical index number.

- char ∗ name

  Object description if any.

- union hwloc_obj_attr_u ∗ attr

  Object type-specific Attributes.

- unsigned depth

  Vertical index in the hierarchy.

- unsigned logical_index

  Horizontal index in the whole list of similar objects, could be a "cousin_rank" since it's the rank within the "cousin" list below.

- struct hwloc_obj ∗ next_cousin

  Next object of same type.

- struct hwloc_obj ∗ prev_cousin

  Previous object of same type.

- struct hwloc_obj ∗ father

  Father, `NULL` *if root (system object).*

- unsigned sibling_rank

  Index in father's `children[]` *array.*

- struct hwloc_obj ∗ next_sibling

  Next object below the same father.

- struct hwloc_obj ∗ prev_sibling

Previous object below the same father.

- unsigned arity

  Number of children.

- struct hwloc_obj ** children

  Children, `children`*[0 .. arity -1].*

- struct hwloc_obj * first_child

  First child.

- struct hwloc_obj * last_child

  Last child.

- void * userdata

  Application-given private data pointer, initialized to `NULL`*, use it as you wish.*

- hwloc_cpuset_t cpuset

  CPUs covered by this object.

- signed os_level

  OS-provided physical level.

6.5.1 Detailed Description

Structure of a topology object.

Applications mustn't modify any field except userdata .

6.5.2 Field Documentation

6.5.2.1 unsigned hwloc_obj::arity

Number of children.

6.5.2.2 union hwloc_obj_attr_u* hwloc_obj::attr `[write]`

Object type-specific Attributes.

6.5.2.3 struct hwloc_obj** hwloc_obj::children [read]

Children, `children`[0 .. arity -1].

6.5.2.4 hwloc_cpuset_t hwloc_obj::cpuset

CPUs covered by this object.

6.5.2.5 unsigned hwloc_obj::depth

Vertical index in the hierarchy.

6.5.2.6 struct hwloc_obj* hwloc_obj::father [read]

Father, `NULL` if root (system object).

6.5.2.7 struct hwloc_obj* hwloc_obj::first_child [read]

First child.

6.5.2.8 struct hwloc_obj* hwloc_obj::last_child [read]

Last child.

6.5.2.9 unsigned hwloc_obj::logical_index

Horizontal index in the whole list of similar objects, could be a "cousin_rank" since it's the rank within the "cousin" list below.

6.5.2.10 char* hwloc_obj::name

Object description if any.

6.5.2.11 struct hwloc_obj* hwloc_obj::next_cousin [read]

Next object of same type.

6.5.2.12 struct hwloc_obj* hwloc_obj::next_sibling [read]

Next object below the same father.

6.5.2.13 signed hwloc_obj::os_index

OS-provided physical index number.

6.5.2.14 signed hwloc_obj::os_level

OS-provided physical level.

6.5.2.15 struct hwloc_obj∗ hwloc_obj::prev_cousin `[read]`

Previous object of same type.

6.5.2.16 struct hwloc_obj∗ hwloc_obj::prev_sibling `[read]`

Previous object below the same father.

6.5.2.17 unsigned hwloc_obj::sibling_rank

Index in father's `children`[] array.

6.5.2.18 hwloc_obj_type_t hwloc_obj::type

Type of object.

6.5.2.19 void∗ hwloc_obj::userdata

Application-given private data pointer, initialized to `NULL`, use it as you wish.

The documentation for this struct was generated from the following file:

- hwloc.h

6.6 hwloc_obj_attr_u Union Reference

Object type-specific Attributes.

```
#include <hwloc.h>
```

Data Structures

- struct hwloc_cache_attr_s

  Cache-specific Object Attributes.
- struct hwloc_machine_attr_s

  Machine-specific Object Attributes.
- struct hwloc_memory_attr_s

  Node-specific Object Attributes.
- struct hwloc_misc_attr_s

  Misc-specific Object Attributes.

Data Fields

- struct hwloc_obj_attr_u::hwloc_cache_attr_s cache

  Cache-specific Object Attributes.
- struct hwloc_obj_attr_u::hwloc_memory_attr_s node

  Node-specific Object Attributes.
- struct hwloc_obj_attr_u::hwloc_machine_attr_s machine

  Machine-specific Object Attributes.
- struct hwloc_machine_attr_s system

  System-specific Object Attributes.
- struct hwloc_obj_attr_u::hwloc_misc_attr_s misc

  Misc-specific Object Attributes.

6.6.1 Detailed Description

Object type-specific Attributes.

6.6.2 Field Documentation

6.6.2.1 struct hwloc_obj_attr_u::hwloc_cache_attr_s hwloc_obj_attr_u::cache

Cache-specific Object Attributes.

6.6.2.2 struct hwloc_obj_attr_u::hwloc_machine_attr_s hwloc_obj_attr_u::machine

Machine-specific Object Attributes.

6.6.2.3 struct hwloc_obj_attr_u::hwloc_misc_attr_s hwloc_obj_attr_u::misc

Misc-specific Object Attributes.

6.6.2.4 struct hwloc_obj_attr_u::hwloc_memory_attr_s hwloc_obj_attr_u::node

Node-specific Object Attributes.

6.6.2.5 struct hwloc_machine_attr_s hwloc_obj_attr_u::system `[read]`

System-specific Object Attributes.

The documentation for this union was generated from the following file:

- hwloc.h

Index