

# Hardware Locality (hwloc)

## 0.9.1

Generated by Doxygen 1.6.1

Thu Nov 5 07:59:50 2009



# Contents

<b>1</b>	<b>hwloc</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	2
1.3	Examples . . . . .	2
1.4	Programming interface . . . . .	6
1.5	Interface example . . . . .	6
1.6	Questions and bugs . . . . .	8
1.7	History / credits . . . . .	8
<b>2</b>	<b>Glossary</b>	<b>9</b>
<b>3</b>	<b>Module Index</b>	<b>13</b>
3.1	Modules . . . . .	13
<b>4</b>	<b>Data Structure Index</b>	<b>15</b>
4.1	Data Structures . . . . .	15
<b>5</b>	<b>Module Documentation</b>	<b>17</b>
5.1	Topology context . . . . .	17
5.1.1	Typedef Documentation . . . . .	17
5.1.1.1	hwloc_topology_t . . . . .	17
5.2	Topology Object Types . . . . .	18
5.2.1	Define Documentation . . . . .	18
5.2.1.1	HWLOC_TYPE_UNORDERED . . . . .	18
5.2.2	Enumeration Type Documentation . . . . .	18
5.2.2.1	hwloc_obj_type_t . . . . .	18
5.2.3	Function Documentation . . . . .	19
5.2.3.1	hwloc_compare_types . . . . .	19
5.3	Topology Objects . . . . .	20
5.3.1	Typedef Documentation . . . . .	20

5.3.1.1	<code>hwloc_obj_t</code>	20
5.4	Create and Destroy Topologies	21
5.4.1	Function Documentation	21
5.4.1.1	<code>hwloc_topology_check</code>	21
5.4.1.2	<code>hwloc_topology_destroy</code>	21
5.4.1.3	<code>hwloc_topology_init</code>	21
5.4.1.4	<code>hwloc_topology_load</code>	22
5.5	Configure Topology Detection	23
5.5.1	Detailed Description	23
5.5.2	Enumeration Type Documentation	24
5.5.2.1	<code>hwloc_topology_flags_e</code>	24
5.5.3	Function Documentation	24
5.5.3.1	<code>hwloc_topology_ignore_all_keep_structure</code>	24
5.5.3.2	<code>hwloc_topology_ignore_type</code>	24
5.5.3.3	<code>hwloc_topology_ignore_type_keep_structure</code>	24
5.5.3.4	<code>hwloc_topology_set_flags</code>	24
5.5.3.5	<code>hwloc_topology_set_fsroot</code>	24
5.5.3.6	<code>hwloc_topology_set_synthetic</code>	25
5.5.3.7	<code>hwloc_topology_set_xml</code>	25
5.6	Get some Topology Information	26
5.6.1	Define Documentation	26
5.6.1.1	<code>HWLOC_TYPE_DEPTH_MULTIPLE</code>	26
5.6.1.2	<code>HWLOC_TYPE_DEPTH_UNKNOWN</code>	26
5.6.2	Function Documentation	26
5.6.2.1	<code>hwloc_get_depth_type</code>	26
5.6.2.2	<code>hwloc_get_nbobjs_by_depth</code>	27
5.6.2.3	<code>hwloc_get_nbobjs_by_type</code>	27
5.6.2.4	<code>hwloc_get_type_depth</code>	27
5.6.2.5	<code>hwloc_topology_get_depth</code>	27
5.6.2.6	<code>hwloc_topology_is_thissystem</code>	27
5.7	Retrieve Objects	28
5.7.1	Function Documentation	28
5.7.1.1	<code>hwloc_get_obj_by_depth</code>	28
5.7.1.2	<code>hwloc_get_obj_by_type</code>	28
5.8	Object/String Conversion	29
5.8.1	Function Documentation	29

5.8.1.1	<a href="#">hwloc_obj_cpuset_sprintf</a>	29
5.8.1.2	<a href="#">hwloc_obj_sprintf</a>	29
5.8.1.3	<a href="#">hwloc_obj_type_of_string</a>	29
5.8.1.4	<a href="#">hwloc_obj_type_string</a>	29
5.9	<a href="#">Binding</a>	30
5.9.1	<a href="#">Detailed Description</a>	30
5.9.2	<a href="#">Enumeration Type Documentation</a>	31
5.9.2.1	<a href="#">hwloc_cpubind_policy_t</a>	31
5.9.3	<a href="#">Function Documentation</a>	31
5.9.3.1	<a href="#">hwloc_set_cpubind</a>	31
5.9.3.2	<a href="#">hwloc_set_proc_cpubind</a>	31
5.9.3.3	<a href="#">hwloc_set_thread_cpubind</a>	31
5.10	<a href="#">Object Type Helpers</a>	32
5.10.1	<a href="#">Function Documentation</a>	32
5.10.1.1	<a href="#">hwloc_get_type_or_above_depth</a>	32
5.10.1.2	<a href="#">hwloc_get_type_or_below_depth</a>	32
5.11	<a href="#">Basic Traversal Helpers</a>	33
5.11.1	<a href="#">Function Documentation</a>	33
5.11.1.1	<a href="#">hwloc_get_common_ancestor_obj</a>	33
5.11.1.2	<a href="#">hwloc_get_next_child</a>	33
5.11.1.3	<a href="#">hwloc_get_next_obj_by_depth</a>	33
5.11.1.4	<a href="#">hwloc_get_next_obj_by_type</a>	34
5.11.1.5	<a href="#">hwloc_get_system_obj</a>	34
5.11.1.6	<a href="#">hwloc_obj_is_in_subtree</a>	34
5.12	<a href="#">Finding Objects Inside a CPU set</a>	35
5.12.1	<a href="#">Function Documentation</a>	35
5.12.1.1	<a href="#">hwloc_get_largest_objs_inside_cpuset</a>	35
5.12.1.2	<a href="#">hwloc_get_nbobjs_inside_cpuset_by_depth</a>	35
5.12.1.3	<a href="#">hwloc_get_nbobjs_inside_cpuset_by_type</a>	36
5.12.1.4	<a href="#">hwloc_get_next_obj_inside_cpuset_by_depth</a>	36
5.12.1.5	<a href="#">hwloc_get_next_obj_inside_cpuset_by_type</a>	36
5.12.1.6	<a href="#">hwloc_get_obj_inside_cpuset_by_depth</a>	36
5.12.1.7	<a href="#">hwloc_get_obj_inside_cpuset_by_type</a>	36
5.13	<a href="#">Finding a single Object covering at least CPU set</a>	37
5.13.1	<a href="#">Function Documentation</a>	37
5.13.1.1	<a href="#">hwloc_get_child_covering_cpuset</a>	37

5.13.1.2	<a href="#">hwloc_get_obj_covering_cpuset</a>	37
5.14	Finding a set of similar Objects covering at least a CPU set	38
5.14.1	Function Documentation	38
5.14.1.1	<a href="#">hwloc_get_next_obj_covering_cpuset_by_depth</a>	38
5.14.1.2	<a href="#">hwloc_get_next_obj_covering_cpuset_by_type</a>	38
5.15	Cache-specific Finding Helpers	39
5.15.1	Function Documentation	39
5.15.1.1	<a href="#">hwloc_get_cache_covering_cpuset</a>	39
5.15.1.2	<a href="#">hwloc_get_shared_cache_covering_obj</a>	39
5.16	Advanced Traversal Helpers	40
5.16.1	Function Documentation	40
5.16.1.1	<a href="#">hwloc_get_closest_objs</a>	40
5.17	Binding Helpers	41
5.17.1	Function Documentation	41
5.17.1.1	<a href="#">hwloc_distribute</a>	41
5.18	The Cpuset API	42
5.18.1	Detailed Description	44
5.18.2	Define Documentation	44
5.18.2.1	<a href="#">hwloc_cpuset_foreach_begin</a>	44
5.18.2.2	<a href="#">hwloc_cpuset_foreach_end</a>	44
5.18.3	Typedef Documentation	45
5.18.3.1	<a href="#">hwloc_const_cpuset_t</a>	45
5.18.3.2	<a href="#">hwloc_cpuset_t</a>	45
5.18.4	Function Documentation	45
5.18.4.1	<a href="#">hwloc_cpuset_all_but_cpu</a>	45
5.18.4.2	<a href="#">hwloc_cpuset_alloc</a>	45
5.18.4.3	<a href="#">hwloc_cpuset_andset</a>	45
5.18.4.4	<a href="#">hwloc_cpuset_asprintf</a>	45
5.18.4.5	<a href="#">hwloc_cpuset_clearset</a>	45
5.18.4.6	<a href="#">hwloc_cpuset_clr</a>	45
5.18.4.7	<a href="#">hwloc_cpuset_compar</a>	45
5.18.4.8	<a href="#">hwloc_cpuset_compar_first</a>	45
5.18.4.9	<a href="#">hwloc_cpuset_copy</a>	46
5.18.4.10	<a href="#">hwloc_cpuset_cpu</a>	46
5.18.4.11	<a href="#">hwloc_cpuset_dup</a>	46
5.18.4.12	<a href="#">hwloc_cpuset_fill</a>	46

5.18.4.13 hwloc_cpuset_first . . . . .	46
5.18.4.14 hwloc_cpuset_free . . . . .	46
5.18.4.15 hwloc_cpuset_from_ith_ulong . . . . .	46
5.18.4.16 hwloc_cpuset_from_string . . . . .	46
5.18.4.17 hwloc_cpuset_from_ulong . . . . .	46
5.18.4.18 hwloc_cpuset_intersects . . . . .	46
5.18.4.19 hwloc_cpuset_isequal . . . . .	46
5.18.4.20 hwloc_cpuset_isfull . . . . .	47
5.18.4.21 hwloc_cpuset_isincluded . . . . .	47
5.18.4.22 hwloc_cpuset_isset . . . . .	47
5.18.4.23 hwloc_cpuset_iszero . . . . .	47
5.18.4.24 hwloc_cpuset_last . . . . .	47
5.18.4.25 hwloc_cpuset_orset . . . . .	47
5.18.4.26 hwloc_cpuset_set . . . . .	47
5.18.4.27 hwloc_cpuset_set_range . . . . .	47
5.18.4.28 hwloc_cpuset_singlify . . . . .	47
5.18.4.29 hwloc_cpuset_snprintf . . . . .	47
5.18.4.30 hwloc_cpuset_to_ith_ulong . . . . .	48
5.18.4.31 hwloc_cpuset_to_ulong . . . . .	48
5.18.4.32 hwloc_cpuset_weight . . . . .	48
5.18.4.33 hwloc_cpuset_xorset . . . . .	48
5.18.4.34 hwloc_cpuset_zero . . . . .	48
5.19 Helpers for manipulating glibc sched affinity . . . . .	49
5.19.1 Function Documentation . . . . .	49
5.19.1.1 hwloc_cpuset_from_glibc_sched_affinity . . . . .	49
5.19.1.2 hwloc_cpuset_to_glibc_sched_affinity . . . . .	49
5.20 Helpers for manipulating linux kernel cpumap files . . . . .	50
5.20.1 Function Documentation . . . . .	50
5.20.1.1 hwloc_linux_parse_cpumap_file . . . . .	50
5.21 Helpers for manipulating Linux libnuma unsigned long masks . . . . .	51
5.21.1 Function Documentation . . . . .	51
5.21.1.1 hwloc_cpuset_from_linux_libnuma_ulongs . . . . .	51
5.21.1.2 hwloc_cpuset_to_linux_libnuma_ulongs . . . . .	51
5.22 Helpers for manipulating Linux libnuma bitmask . . . . .	52
5.22.1 Function Documentation . . . . .	52
5.22.1.1 hwloc_cpuset_from_linux_libnuma_bitmask . . . . .	52

5.22.1.2	hwloc_cpuset_to_linux_libnuma_bitmask	52
5.23	Helpers for manipulating Linux libnuma nodemask_t	53
5.23.1	Function Documentation	53
5.23.1.1	hwloc_cpuset_from_linux_libnuma_nodemask	53
5.23.1.2	hwloc_cpuset_to_linux_libnuma_nodemask	53
5.24	OpenFabrics-Specific Functions	54
5.24.1	Function Documentation	54
5.24.1.1	hwloc_ibv_get_device_cpuset	54
<b>6</b>	<b>Data Structure Documentation</b>	<b>55</b>
6.1	hwloc_obj_attr_u::hwloc_cache_attr_s Struct Reference	55
6.1.1	Detailed Description	55
6.1.2	Field Documentation	55
6.1.2.1	depth	55
6.1.2.2	memory_kB	55
6.2	hwloc_obj_attr_u::hwloc_machine_attr_s Struct Reference	56
6.2.1	Detailed Description	56
6.2.2	Field Documentation	56
6.2.2.1	dmi_board_name	56
6.2.2.2	dmi_board_vendor	56
6.2.2.3	huge_page_free	56
6.2.2.4	huge_page_size_kB	56
6.2.2.5	memory_kB	57
6.3	hwloc_obj_attr_u::hwloc_memory_attr_s Struct Reference	58
6.3.1	Detailed Description	58
6.3.2	Field Documentation	58
6.3.2.1	huge_page_free	58
6.3.2.2	memory_kB	58
6.4	hwloc_obj_attr_u::hwloc_misc_attr_s Struct Reference	59
6.4.1	Detailed Description	59
6.4.2	Field Documentation	59
6.4.2.1	depth	59
6.5	hwloc_obj Struct Reference	60
6.5.1	Detailed Description	61
6.5.2	Field Documentation	61
6.5.2.1	arity	61
6.5.2.2	attr	61



6.5.2.3	children	61
6.5.2.4	cpuset	61
6.5.2.5	depth	61
6.5.2.6	father	61
6.5.2.7	first_child	62
6.5.2.8	last_child	62
6.5.2.9	logical_index	62
6.5.2.10	name	62
6.5.2.11	next_cousin	62
6.5.2.12	next_sibling	62
6.5.2.13	os_index	62
6.5.2.14	os_level	62
6.5.2.15	prev_cousin	62
6.5.2.16	prev_sibling	62
6.5.2.17	sibling_rank	62
6.5.2.18	type	63
6.5.2.19	userdata	63
6.6	hwloc_obj_attr_u Union Reference	64
6.6.1	Detailed Description	64
6.6.2	Field Documentation	64
6.6.2.1	cache	64
6.6.2.2	machine	65
6.6.2.3	misc	65
6.6.2.4	node	65
6.6.2.5	system	65



# Chapter 1

## hwloc

### Portable abstraction of hierarchical architectures for high-performance computing

#### 1.1 Introduction

hwloc provides command line tools and a C API to obtain the hierarchical map of key computing elements, such as: NUMA memory nodes, shared caches, processor sockets, processor cores, and processor "threads". hwloc also gathers various attributes such as cache and memory information, and is portable across a variety of different operating systems and platforms.

hwloc primarily aims at helping high-performance computing (HPC) applications, but is also applicable to any project seeking to exploit code and/or data locality on modern computing platforms.

Note that the hwloc project represents the merger of the libtopology project from INRIA and the Portable Linux Processor Affinity (PLPA) sub-project from Open MPI. *Both of these prior projects are now deprecated.* The first hwloc release is essentially a "re-branding" of the libtopology code base, but with both a few genuinely new features and a few PLPA-like features added in. More new features and more PLPA-like features will be added to hwloc over time.

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 / OS X
- OSF/1 (a.k.a., Tru64)
- HP-UX
- Microsoft Windows

hwloc only reports the number of processors on unsupported operating systems; no topology information is available.

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 can display the topology in a human-readable format, either in graphical mode (X11), or by exporting in one of several different formats, including: plain text, PDF, PNG, and FIG (see Examples below). Note that some of the export formats require additional support libraries.

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, and so on.

## 1.2 Installation

hwloc (<http://www.open-mpi.org/projects/hwloc/>) is available under the BSD license. It is hosted as a sub-project of the overall Open MPI project (<http://www.open-mpi.org/>). Note that hwloc does not require any functionality from Open MPI -- it is a wholly separate (and much smaller!) project and code base. It just happens to be hosted as part of the overall Open MPI project.

Nightly development snapshots are available on the web site. Additionally, the code can be directly checked out of Subversion:

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

Note that GNU Autoconf  $\geq 2.60$ , Automake  $\geq 1.10$  and Libtool  $\geq 2.2.6$  are required when building from a Subversion checkout.

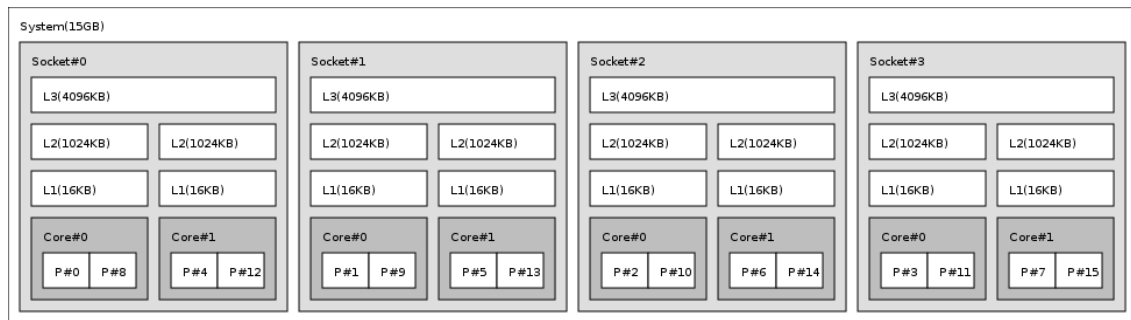
Installation by itself is the fairly common GNU-based process:

```
shell$ ./configure --prefix=...
shell$ make
shell$ make install
```

The hwloc command-line tool "lstopo" produces human-readable topology maps, as mentioned above. It can also export maps to the "fig" file format. Support for PDF, Postscript, and PNG exporting is provided if the "Cairo" development package can be found when hwloc is configured and build. Similarly, lstopo's XML support requires the libxml2 development package.

## 1.3 Examples

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

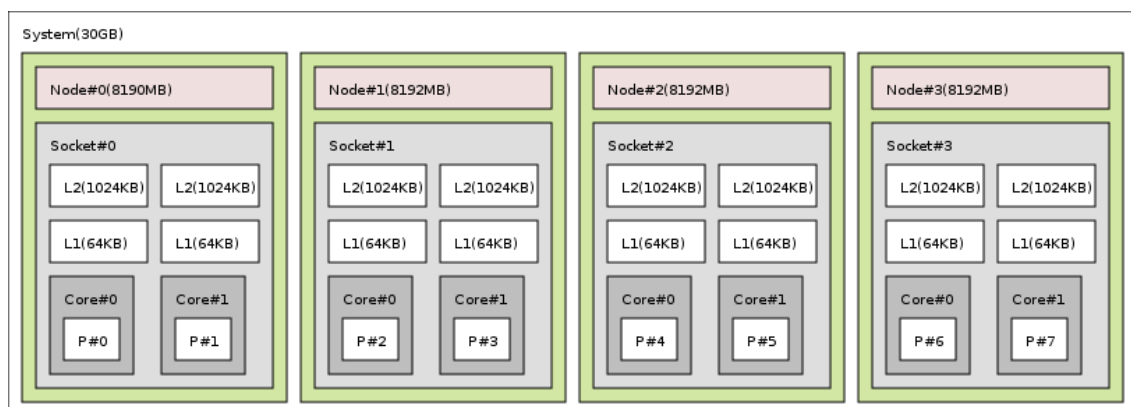


```

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

```

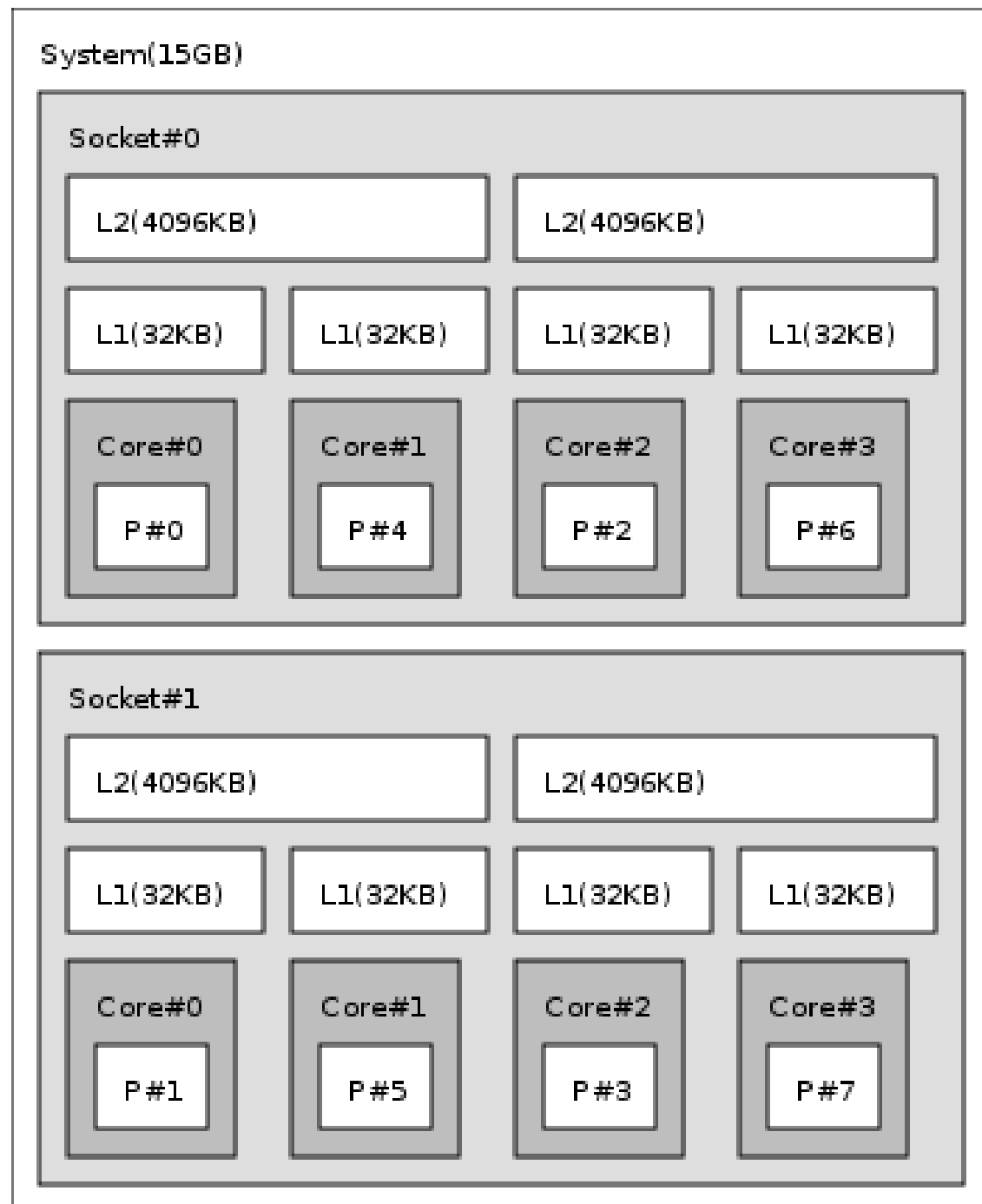
On a 4-socket 2-core Opteron NUMA machine, the `lstopo` tool may show the following outputs:



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, with 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. Developers should look at [hwloc/helper.h](#), which provides good higher-level topology traversal examples.

Each object contains a cpuset describing 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 Linux systems with the OpenFabrics verbs library, some dedicated helpers are provided in [hwloc/openfabrics-verbs.h](#) (this helper file is not yet useful on non-Linux systems with the OpenFabrics verbs library).

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

Further documentation is available in a full set of HTML pages, man pages, and self-contained PDF files (formatted for both both US letter and A4 formats) in the source tarball in doc/doxygen-doc/. If you are building from a Subversion checkout, you will need to have Doxygen and pdflatex installed -- the documentation will be built during the normal "make" process. The documentation is installed during "make install" to \$prefix/share/doc/hwloc/ and your systems default man page tree (under \$prefix, of course).

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)

```

```

/* Example hwloc API program.
 *
 * Copyright © 2009 INRIA, Université Bordeaux 1
 *
 * 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 objectst 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

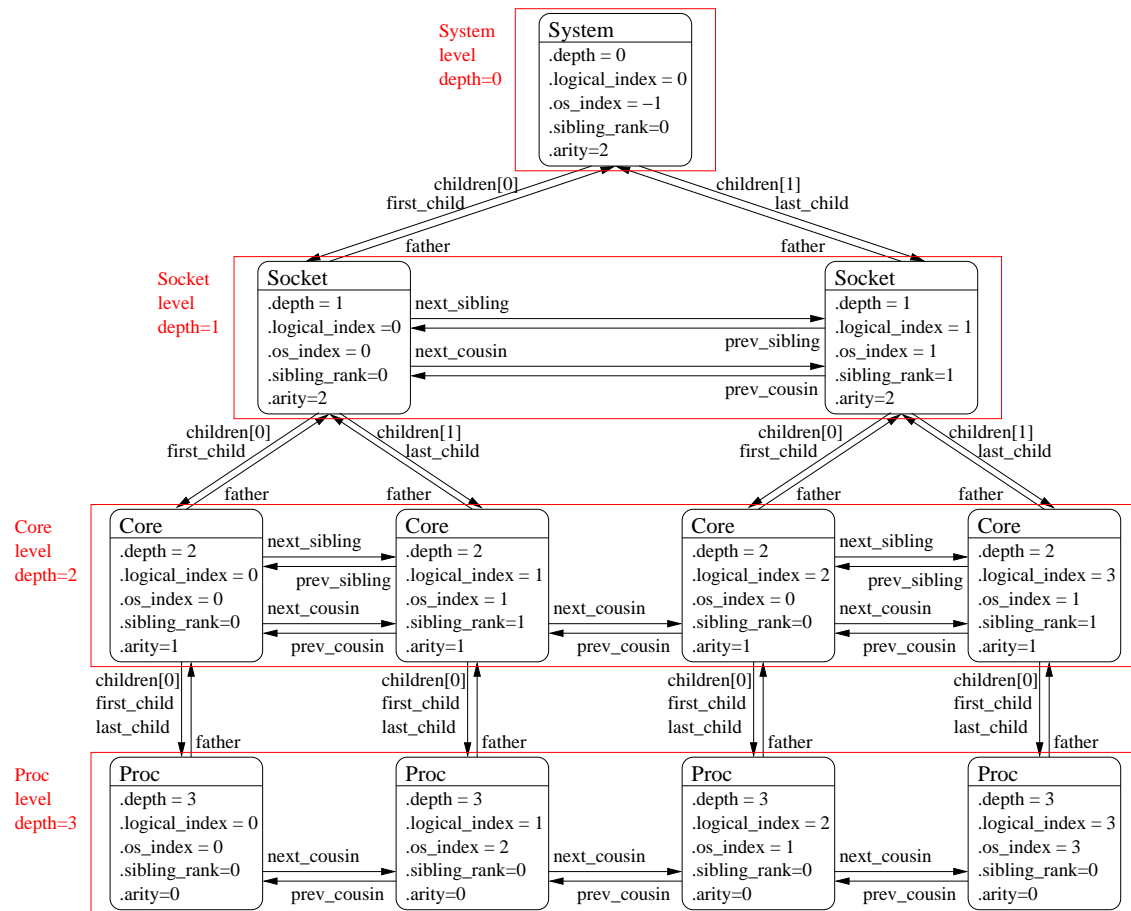
**Level** Set of objects of the same type

**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 should be noted 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:

Topology context . . . . .	17
Topology Object Types . . . . .	18
Topology Objects . . . . .	20
Create and Destroy Topologies . . . . .	21
Configure Topology Detection . . . . .	23
Get some Topology Information . . . . .	26
Retrieve Objects . . . . .	28
Object/String Conversion . . . . .	29
Binding . . . . .	30
Object Type Helpers . . . . .	32
Basic Traversal Helpers . . . . .	33
Finding Objects Inside a CPU set . . . . .	35
Finding a single Object covering at least CPU set . . . . .	37
Finding a set of similar Objects covering at least a CPU set . . . . .	38
Cache-specific Finding Helpers . . . . .	39
Advanced Traversal Helpers . . . . .	40
Binding Helpers . . . . .	41
The Cpuset API . . . . .	42
Helpers for manipulating glibc sched affinity . . . . .	49
Helpers for manipulating linux kernel cpumap files . . . . .	50
Helpers for manipulating Linux libnuma unsigned long masks . . . . .	51
Helpers for manipulating Linux libnuma bitmask . . . . .	52
Helpers for manipulating Linux libnuma nodemask_t . . . . .	53
OpenFabrics-Specific Functions . . . . .	54





## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">hwloc_obj_attr_u::hwloc_cache_attr_s</a> (Cache-specific Object Attributes ) . . . . .	55
<a href="#">hwloc_obj_attr_u::hwloc_machine_attr_s</a> (Machine-specific Object Attributes ) . . . . .	56
<a href="#">hwloc_obj_attr_u::hwloc_memory_attr_s</a> (Node-specific Object Attributes ) . . . . .	58
<a href="#">hwloc_obj_attr_u::hwloc_misc_attr_s</a> (Misc-specific Object Attributes ) . . . . .	59
<a href="#">hwloc_obj</a> (Structure of a topology object ) . . . . .	60
<a href="#">hwloc_obj_attr_u</a> (Object type-specific Attributes ) . . . . .	64



# 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.

This default behavior may also be modified through environment variables if the application did not modify it already. Setting `HWLOC_XMLFILE` in the environment enforces the discovery from a XML file as if `hwloc_topology_set_xml()` had been called. `HWLOC_FSROOT` switches to reading the topology from the specified Linux filesystem root as if `hwloc_topology_set_fsroot()` had been called. Finally, `HWLOC_THISSYSTEM` enforces the value of the `is_thissystem` field.

## 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. Setting the environment variable HWLOC\_FSROOT may also result in this behavior. 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`. Setting the environment variable `HWLOC_XMLFILE` may also result in this behavior. 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 hierarchical 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_nobjs_by_depth` (`hwloc_topology_t` topology, unsigned depth)  
*Returns the width of level at depth `depth`.*
- static inline int `hwloc_get_nobjs_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\_nobjs\_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\_nobjs\_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 hierarchical 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 idx)  
*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 idx)  
*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 idx)

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 idx) [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. Fill string `string` up to `size` characters with the description of topology object `obj` in topology `topology`.

If `verbose` is set, a longer description is used. Otherwise a short description is used.

`indexprefix` is used to prefix the `os_index` attribute number of the object in the description. If `NULL`, the `#` character is used.

##### 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.

#### 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. By default, when the designated CPUs are all busy while other CPUs are idle, OSes may execute the thread/process on those other CPUs instead of the designated CPUs, to let them progress anyway. Strict binding means that the thread/process will *never* execute on other cpus than the designated CPUs, even when those are busy with other tasks and other CPUs are idle.

#### Note:

Depending on OSes and implementations, strict binding may not be possible (implementation reason) or not allowed (administrative reasons), and the 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 `obj1` and `obj2`.*
- 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 `obj1` and `obj2`.

#### 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 idx)`  
*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 idx)`  
*Return the idx-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 invocation 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 idx) [static]`

Return the `idx`-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 idx) [static]`

Return the `idx`-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 inline `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 inline `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 inline `hwloc_obj_t hwloc_get_child_covering_cpuset` (`hwloc_topology_t` topology, `hwloc_cpuset_t` set, `hwloc_obj_t` father) [`static`]

Get the child covering at least CPU set set.

#### Returns:

NULL if no child matches.

**5.13.1.2** static inline `hwloc_obj_t hwloc_get_obj_covering_cpuset` (`hwloc_topology_t` topology, `hwloc_cpuset_t` set) [`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 invocation 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 invocation 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_shared_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_shared_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_cpuset_s * hwloc_cpuset_t`  
*Set of CPUs represented as an opaque pointer to an internal bitmask.*
- `typedef struct hwloc_cpuset_s * hwloc_const_cpuset_t`

### 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_const_cpuset_t set)`  
*Stringify a cpuset.*
- `int hwloc_cpuset_asprintf (char **strp, hwloc_const_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_const_cpuset_t` set)  
*Convert the beginning part of CPU set set into unsigned long mask.*
- unsigned long `hwloc_cpuset_to_ith_ulong` (`hwloc_const_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_const_cpuset_t` set, unsigned cpu)  
*Test whether CPU cpu is part of set set.*
- int `hwloc_cpuset_iszero` (`hwloc_const_cpuset_t` set)  
*Test whether set set is zero.*
- int `hwloc_cpuset_isfull` (`hwloc_const_cpuset_t` set)  
*Test whether set set is full.*
- int `hwloc_cpuset_isequal` (`hwloc_const_cpuset_t` set1, `hwloc_const_cpuset_t` set2)  
*Test whether set set1 is equal to set set2.*
- int `hwloc_cpuset_intersects` (`hwloc_const_cpuset_t` set1, `hwloc_const_cpuset_t` set2)  
*Test whether sets set1 and set2 intersects.*
- int `hwloc_cpuset_isincluded` (`hwloc_const_cpuset_t` sub\_set, `hwloc_const_cpuset_t` super\_set)  
*Test whether set sub\_set is part of set super\_set.*
- void `hwloc_cpuset_orset` (`hwloc_cpuset_t` set, `hwloc_const_cpuset_t` modifier\_set)  
*Or set modifier\_set into set set.*
- void `hwloc_cpuset_andset` (`hwloc_cpuset_t` set, `hwloc_const_cpuset_t` modifier\_set)  
*And set modifier\_set into set set.*
- void `hwloc_cpuset_clearset` (`hwloc_cpuset_t` set, `hwloc_const_cpuset_t` modifier\_set)

*Clear set modifier\_set out of set set.*

- void [hwloc\\_cpuset\\_xorset](#) ([hwloc\\_cpuset\\_t](#) set, [hwloc\\_const\\_cpuset\\_t](#) modifier\_set)  
*Xor set set with set modifier\_set.*
- int [hwloc\\_cpuset\\_first](#) ([hwloc\\_const\\_cpuset\\_t](#) set)  
*Compute the first CPU (least significant bit) in CPU set set.*
- int [hwloc\\_cpuset\\_last](#) ([hwloc\\_const\\_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\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2)  
*Compar CPU sets set1 and set2 using their first set bit.*
- int [hwloc\\_cpuset\\_compar](#) ([hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2)  
*Compar CPU sets set1 and set2 using their last bits.*
- int [hwloc\\_cpuset\\_weight](#) ([hwloc\\_const\\_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_cpuset_s* hwloc_const_cpuset_t`

**5.18.3.2** `typedef struct hwloc_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_const_cpuset_t modifier_set)`

And set *modifier\_set* into set *set*.

**5.18.4.4** `int hwloc_cpuset_asprintf (char **strp, hwloc_const_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_const_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_const_cpuset_t set1, hwloc_const_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_const_cpuset_t set1, hwloc_const_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\_const\_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\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

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

**5.18.4.19 int hwloc\_cpuset\_isequal (hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

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



**5.18.4.20** `int hwloc_cpuset_isfull (hwloc_const_cpuset_t set)`

Test whether set `set` is full.

**5.18.4.21** `int hwloc_cpuset_isincluded (hwloc_const_cpuset_t sub_set, hwloc_const_cpuset_t super_set)`

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

**5.18.4.22** `int hwloc_cpuset_isset (hwloc_const_cpuset_t set, unsigned cpu)`

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

**5.18.4.23** `int hwloc_cpuset_iszero (hwloc_const_cpuset_t set)`

Test whether set `set` is zero.

**5.18.4.24** `int hwloc_cpuset_last (hwloc_const_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_const_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_const_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\_const\_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\_const\_cpuset\_t *set*)**

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

**5.18.4.32 int hwloc\_cpuset\_weight (hwloc\_const\_cpuset\_t *set*)**

Compute the weight of CPU set *set*.

**5.18.4.33 void hwloc\_cpuset\_xorset (hwloc\_cpuset\_t *set*, hwloc\_const\_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 kernel cpumap files

### Functions

- [hwloc\\_cpuset\\_t hwloc\\_linux\\_parse\\_cpumap\\_file](#) (FILE \*file)  
*Convert a linux kernel cpumap file `file` into hwloc CPU set.*

### 5.20.1 Function Documentation

#### 5.20.1.1 [hwloc\\_cpuset\\_t hwloc\\_linux\\_parse\\_cpumap\\_file](#) (FILE \**file*)

Convert a linux kernel cpumap file `file` into hwloc CPU set. Might be used when reading CPU set from sysfs attributes such as topology and caches for processors, or local\_cpus for devices.

## 5.21 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.*

### 5.21.1 Function Documentation

**5.21.1.1 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).

**5.21.1.2 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.

## 5.22 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.*

### 5.22.1 Function Documentation

#### 5.22.1.1 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.

#### 5.22.1.2 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 bitmask as an input parameter.

## 5.23 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.*

### 5.23.1 Function Documentation

#### 5.23.1.1 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.

#### 5.23.1.2 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.

## 5.24 OpenFabrics-Specific Functions

### Functions

- static inline `hwloc_cpuset_t hwloc_ibv_get_device_cpuset` (struct `ibv_device` \*`ibdev`)  
*Get the CPU set of logical processors that are physically close to device `ibdev`.*

#### 5.24.1 Function Documentation

##### 5.24.1.1 static inline `hwloc_cpuset_t hwloc_ibv_get_device_cpuset` (struct `ibv_device` \* `ibdev`) [static]

Get the CPU set of logical processors that are physically close to device `ibdev`. For the given OpenFabrics device `ibdev`, read the corresponding kernel-provided `cpumap` file and return the corresponding CPU set. This function is currently only implemented in a meaningful way for Linux; other systems will simply get a full `cpuset`.



## 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

Advanced Traversal Helpers, [40](#)

arity

hwloc\_obj, [61](#)

attr

hwloc\_obj, [61](#)

Basic Traversal Helpers, [33](#)

Binding, [30](#)

Binding Helpers, [41](#)

cache

hwloc\_obj\_attr\_u, [64](#)

Cache-specific Finding Helpers, [39](#)

children

hwloc\_obj, [61](#)

Configure Topology Detection, [23](#)

cpuset

hwloc\_obj, [61](#)

Create and Destroy Topologies, [21](#)

depth

hwloc\_obj, [61](#)

hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [55](#)

hwloc\_obj\_attr\_u::hwloc\_misc\_attr\_s, [59](#)

dmi\_board\_name

hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [56](#)

dmi\_board\_vendor

hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [56](#)

father

hwloc\_obj, [61](#)

Finding a set of similar Objects covering at least a CPU set, [38](#)

Finding a single Object covering at least CPU set, [37](#)

Finding Objects Inside a CPU set, [35](#)

first\_child

hwloc\_obj, [61](#)

Get some Topology Information, [26](#)

Helpers for manipulating glibc sched affinity, [49](#)

Helpers for manipulating linux kernel cpumap files, [50](#)

Helpers for manipulating Linux libnuma bitmask, [52](#)

Helpers for manipulating Linux libnuma nodemask\_t, [53](#)

Helpers for manipulating Linux libnuma unsigned long masks, [51](#)

huge\_page\_free

hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [56](#)

hwloc\_obj\_attr\_u::hwloc\_memory\_attr\_s, [58](#)

huge\_page\_size\_kB

hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [56](#)

HWLOC\_CPUBIND\_PROCESS

hwlocality\_binding, [31](#)

HWLOC\_CPUBIND\_STRICT

hwlocality\_binding, [31](#)

HWLOC\_CPUBIND\_THREAD

hwlocality\_binding, [31](#)

HWLOC\_OBJ\_CACHE

hwlocality\_types, [18](#)

HWLOC\_OBJ\_CORE

hwlocality\_types, [19](#)

HWLOC\_OBJ\_MACHINE

hwlocality\_types, [18](#)

HWLOC\_OBJ\_MISC

hwlocality\_types, [19](#)

HWLOC\_OBJ\_NODE

hwlocality\_types, [18](#)

HWLOC\_OBJ\_PROC

hwlocality\_types, [19](#)

HWLOC\_OBJ\_SOCKET

hwlocality\_types, [18](#)

HWLOC\_OBJ\_SYSTEM

hwlocality\_types, [18](#)

HWLOC\_TOPOLOGY\_FLAG\_IS\_THISSYSTEM

hwlocality\_configuration, [24](#)

HWLOC\_TOPOLOGY\_FLAG\_WHOLE\_-  
SYSTEM

hwlocality\_configuration, [24](#)

hwloc\_compare\_types

hwlocality\_types, [19](#)

hwloc\_const\_cpuset\_t

hwlocality\_cpuset, [45](#)

hwloc\_cpubind\_policy\_t

hwlocality\_binding, [31](#)

hwloc\_cpuset\_all\_but\_cpu

hwlocality\_cpuset, [45](#)

hwloc\_cpuset\_alloc

- hwlocality\_cpuset, 45
- hwloc\_cpuset\_andset
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_asprintf
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_clearset
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_clr
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_compar
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_compar\_first
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_copy
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_cpu
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_dup
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_fill
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_first
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_foreach\_begin
  - hwlocality\_cpuset, 44
- hwloc\_cpuset\_foreach\_end
  - hwlocality\_cpuset, 44
- hwloc\_cpuset\_free
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_from\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, 49
- hwloc\_cpuset\_from\_ith\_ulong
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_from\_linux\_libnuma\_bitmask
  - hwlocality\_linux\_libnuma\_bitmask, 52
- hwloc\_cpuset\_from\_linux\_libnuma\_nodemask
  - hwlocality\_linux\_libnuma\_nodemask, 53
- hwloc\_cpuset\_from\_linux\_libnuma\_ulongs
  - hwlocality\_linux\_libnuma\_ulongs, 51
- hwloc\_cpuset\_from\_string
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_from\_ulong
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_intersects
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_isequal
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_isfull
  - hwlocality\_cpuset, 46
- hwloc\_cpuset\_isincluded
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_isset
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_iszero
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_last
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_orset
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_set
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_set\_range
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_singlify
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_snprintf
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_t
  - hwlocality\_cpuset, 45
- hwloc\_cpuset\_to\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, 49
- hwloc\_cpuset\_to\_ith\_ulong
  - hwlocality\_cpuset, 47
- hwloc\_cpuset\_to\_linux\_libnuma\_bitmask
  - hwlocality\_linux\_libnuma\_bitmask, 52
- hwloc\_cpuset\_to\_linux\_libnuma\_nodemask
  - hwlocality\_linux\_libnuma\_nodemask, 53
- hwloc\_cpuset\_to\_linux\_libnuma\_ulongs
  - hwlocality\_linux\_libnuma\_ulongs, 51
- hwloc\_cpuset\_to\_ulong
  - hwlocality\_cpuset, 48
- hwloc\_cpuset\_weight
  - hwlocality\_cpuset, 48
- hwloc\_cpuset\_xorset
  - hwlocality\_cpuset, 48
- hwloc\_cpuset\_zero
  - hwlocality\_cpuset, 48
- hwloc\_distribute
  - hwlocality\_helper\_binding, 41
- hwloc\_get\_cache\_covering\_cpuset
  - hwlocality\_helper\_find\_cache, 39
- hwloc\_get\_child\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 37
- hwloc\_get\_closest\_objs
  - hwlocality\_helper\_traversal, 40
- hwloc\_get\_common\_ancestor\_obj
  - hwlocality\_helper\_traversal\_basic, 33
- hwloc\_get\_depth\_type
  - hwlocality\_information, 26
- hwloc\_get\_largest\_objs\_inside\_cpuset
  - hwlocality\_helper\_find\_inside, 35
- hwloc\_get\_nbobjs\_by\_depth
  - hwlocality\_information, 26
- hwloc\_get\_nbobjs\_by\_type
  - hwlocality\_information, 27
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_inside, 35
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_type

- hwlocality\_helper\_find\_inside, 35
- hwloc\_get\_next\_child
  - hwlocality\_helper\_traversal\_basic, 33
- hwloc\_get\_next\_obj\_by\_depth
  - hwlocality\_helper\_traversal\_basic, 33
- hwloc\_get\_next\_obj\_by\_type
  - hwlocality\_helper\_traversal\_basic, 33
- hwloc\_get\_next\_obj\_covering\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_coverings, 38
- hwloc\_get\_next\_obj\_covering\_cpuset\_by\_type
  - hwlocality\_helper\_find\_coverings, 38
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_inside, 36
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_type
  - hwlocality\_helper\_find\_inside, 36
- hwloc\_get\_obj\_by\_depth
  - hwlocality\_traversal, 28
- hwloc\_get\_obj\_by\_type
  - hwlocality\_traversal, 28
- hwloc\_get\_obj\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 37
- hwloc\_get\_obj\_inside\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_inside, 36
- hwloc\_get\_obj\_inside\_cpuset\_by\_type
  - hwlocality\_helper\_find\_inside, 36
- hwloc\_get\_shared\_cache\_covering\_obj
  - hwlocality\_helper\_find\_cache, 39
- hwloc\_get\_system\_obj
  - hwlocality\_helper\_traversal\_basic, 34
- hwloc\_get\_type\_depth
  - hwlocality\_information, 27
- hwloc\_get\_type\_or\_above\_depth
  - hwlocality\_helper\_types, 32
- hwloc\_get\_type\_or\_below\_depth
  - hwlocality\_helper\_types, 32
- hwloc\_ibv\_get\_device\_cpuset
  - hwloc\_openfabrics, 54
- hwloc\_linux\_parse\_cpumap\_file
  - hwlocality\_linux\_cpumap, 50
- hwloc\_obj, 60
  - arity, 61
  - attr, 61
  - children, 61
  - cpuset, 61
  - depth, 61
  - father, 61
  - first\_child, 61
  - last\_child, 62
  - logical\_index, 62
  - name, 62
  - next\_cousin, 62
  - next\_sibling, 62
  - os\_index, 62
  - os\_level, 62
  - prev\_cousin, 62
  - prev\_sibling, 62
  - sibling\_rank, 62
  - type, 62
  - userdata, 63
- hwloc\_obj\_attr\_u, 64
  - cache, 64
  - machine, 64
  - misc, 65
  - node, 65
  - system, 65
- hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, 55
  - depth, 55
  - memory\_kB, 55
- hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, 56
  - dmi\_board\_name, 56
  - dmi\_board\_vendor, 56
  - huge\_page\_free, 56
  - huge\_page\_size\_kB, 56
  - memory\_kB, 56
- hwloc\_obj\_attr\_u::hwloc\_memory\_attr\_s, 58
  - huge\_page\_free, 58
  - memory\_kB, 58
- hwloc\_obj\_attr\_u::hwloc\_misc\_attr\_s, 59
  - depth, 59
- hwloc\_obj\_cpuset\_snprintf
  - hwlocality\_conversion, 29
- hwloc\_obj\_is\_in\_subtree
  - hwlocality\_helper\_traversal\_basic, 34
- hwloc\_obj\_snprintf
  - hwlocality\_conversion, 29
- hwloc\_obj\_t
  - hwlocality\_objects, 20
- hwloc\_obj\_type\_of\_string
  - hwlocality\_conversion, 29
- hwloc\_obj\_type\_string
  - hwlocality\_conversion, 29
- hwloc\_obj\_type\_t
  - hwlocality\_types, 18
- hwloc\_openfabrics
  - hwloc\_ibv\_get\_device\_cpuset, 54
- hwloc\_set\_cpupbind
  - hwlocality\_binding, 31
- hwloc\_set\_proc\_cpupbind
  - hwlocality\_binding, 31
- hwloc\_set\_thread\_cpupbind
  - hwlocality\_binding, 31
- hwloc\_topology\_check
  - hwlocality\_creation, 21
- hwloc\_topology\_destroy
  - hwlocality\_creation, 21
- hwloc\_topology\_flags\_e
  - hwlocality\_configuration, 24
- hwloc\_topology\_get\_depth

- hwlocality\_information, 27
- hwloc\_topology\_ignore\_all\_keep\_structure
  - hwlocality\_configuration, 24
- hwloc\_topology\_ignore\_type
  - hwlocality\_configuration, 24
- hwloc\_topology\_ignore\_type\_keep\_structure
  - hwlocality\_configuration, 24
- hwloc\_topology\_init
  - hwlocality\_creation, 21
- hwloc\_topology\_is\_thissystem
  - hwlocality\_information, 27
- hwloc\_topology\_load
  - hwlocality\_creation, 21
- hwloc\_topology\_set\_flags
  - hwlocality\_configuration, 24
- hwloc\_topology\_set\_fsroot
  - hwlocality\_configuration, 24
- hwloc\_topology\_set\_synthetic
  - hwlocality\_configuration, 24
- hwloc\_topology\_set\_xml
  - hwlocality\_configuration, 25
- hwloc\_topology\_t
  - hwlocality\_topology, 17
- HWLOC\_TYPE\_DEPTH\_MULTIPLE
  - hwlocality\_information, 26
- HWLOC\_TYPE\_DEPTH\_UNKNOWN
  - hwlocality\_information, 26
- HWLOC\_TYPE\_UNORDERED
  - hwlocality\_types, 18
- hwlocality\_binding
  - HWLOC\_CPUBIND\_PROCESS, 31
  - HWLOC\_CPUBIND\_STRICT, 31
  - HWLOC\_CPUBIND\_THREAD, 31
- hwlocality\_configuration
  - HWLOC\_TOPOLOGY\_FLAG\_IS\_THISSYSTEM, 24
  - HWLOC\_TOPOLOGY\_FLAG\_WHOLE\_SYSTEM, 24
- hwlocality\_types
  - HWLOC\_OBJ\_CACHE, 18
  - HWLOC\_OBJ\_CORE, 19
  - HWLOC\_OBJ\_MACHINE, 18
  - HWLOC\_OBJ\_MISC, 19
  - HWLOC\_OBJ\_NODE, 18
  - HWLOC\_OBJ\_PROC, 19
  - HWLOC\_OBJ\_SOCKET, 18
  - HWLOC\_OBJ\_SYSTEM, 18
- hwlocality\_binding
  - hwloc\_cpubind\_policy\_t, 31
  - hwloc\_set\_cpubind, 31
  - hwloc\_set\_proc\_cpubind, 31
  - hwloc\_set\_thread\_cpubind, 31
- hwlocality\_configuration
  - hwloc\_topology\_flags\_e, 24
- hwloc\_topology\_ignore\_all\_keep\_structure, 24
- hwloc\_topology\_ignore\_type, 24
- hwloc\_topology\_ignore\_type\_keep\_structure, 24
- hwloc\_topology\_set\_flags, 24
- hwloc\_topology\_set\_fsroot, 24
- hwloc\_topology\_set\_synthetic, 24
- hwloc\_topology\_set\_xml, 25
- hwlocality\_conversion
  - hwloc\_obj\_cpuset\_sprintf, 29
  - hwloc\_obj\_sprintf, 29
  - hwloc\_obj\_type\_of\_string, 29
  - hwloc\_obj\_type\_string, 29
- hwlocality\_cpuset
  - hwloc\_const\_cpuset\_t, 45
  - hwloc\_cpuset\_all\_but\_cpu, 45
  - hwloc\_cpuset\_alloc, 45
  - hwloc\_cpuset\_andset, 45
  - hwloc\_cpuset\_asprintf, 45
  - hwloc\_cpuset\_clearset, 45
  - hwloc\_cpuset\_clr, 45
  - hwloc\_cpuset\_compar, 45
  - hwloc\_cpuset\_compar\_first, 45
  - hwloc\_cpuset\_copy, 45
  - hwloc\_cpuset\_cpu, 46
  - hwloc\_cpuset\_dup, 46
  - hwloc\_cpuset\_fill, 46
  - hwloc\_cpuset\_first, 46
  - hwloc\_cpuset\_foreach\_begin, 44
  - hwloc\_cpuset\_foreach\_end, 44
  - hwloc\_cpuset\_free, 46
  - hwloc\_cpuset\_from\_ith\_ulong, 46
  - hwloc\_cpuset\_from\_string, 46
  - hwloc\_cpuset\_from\_ulong, 46
  - hwloc\_cpuset\_intersects, 46
  - hwloc\_cpuset\_isequal, 46
  - hwloc\_cpuset\_isfull, 46
  - hwloc\_cpuset\_isincluded, 47
  - hwloc\_cpuset\_isset, 47
  - hwloc\_cpuset\_iszero, 47
  - hwloc\_cpuset\_last, 47
  - hwloc\_cpuset\_orset, 47
  - hwloc\_cpuset\_set, 47
  - hwloc\_cpuset\_set\_range, 47
  - hwloc\_cpuset\_singlify, 47
  - hwloc\_cpuset\_sprintf, 47
  - hwloc\_cpuset\_t, 45
  - hwloc\_cpuset\_to\_ith\_ulong, 47
  - hwloc\_cpuset\_to\_ulong, 48
  - hwloc\_cpuset\_weight, 48
  - hwloc\_cpuset\_xorset, 48
  - hwloc\_cpuset\_zero, 48
- hwlocality\_creation

- hwloc\_topology\_check, 21
- hwloc\_topology\_destroy, 21
- hwloc\_topology\_init, 21
- hwloc\_topology\_load, 21
- hwlocality\_glibc\_sched
  - hwloc\_cpuset\_from\_glibc\_sched\_affinity, 49
  - hwloc\_cpuset\_to\_glibc\_sched\_affinity, 49
- hwlocality\_helper\_binding
  - hwloc\_distribute, 41
- hwlocality\_helper\_find\_cache
  - hwloc\_get\_cache\_covering\_cpuset, 39
  - hwloc\_get\_shared\_cache\_covering\_obj, 39
- hwlocality\_helper\_find\_covering
  - hwloc\_get\_child\_covering\_cpuset, 37
  - hwloc\_get\_obj\_covering\_cpuset, 37
- hwlocality\_helper\_find\_coverings
  - hwloc\_get\_next\_obj\_covering\_cpuset\_by\_depth, 38
  - hwloc\_get\_next\_obj\_covering\_cpuset\_by\_type, 38
- hwlocality\_helper\_find\_inside
  - hwloc\_get\_largest\_objs\_inside\_cpuset, 35
  - hwloc\_get\_nbobjs\_inside\_cpuset\_by\_depth, 35
  - hwloc\_get\_nbobjs\_inside\_cpuset\_by\_type, 35
  - hwloc\_get\_next\_obj\_inside\_cpuset\_by\_depth, 36
  - hwloc\_get\_next\_obj\_inside\_cpuset\_by\_type, 36
  - hwloc\_get\_obj\_inside\_cpuset\_by\_depth, 36
  - hwloc\_get\_obj\_inside\_cpuset\_by\_type, 36
- hwlocality\_helper\_traversal
  - hwloc\_get\_closest\_objs, 40
- hwlocality\_helper\_traversal\_basic
  - hwloc\_get\_common\_ancestor\_obj, 33
  - hwloc\_get\_next\_child, 33
  - hwloc\_get\_next\_obj\_by\_depth, 33
  - hwloc\_get\_next\_obj\_by\_type, 33
  - hwloc\_get\_system\_obj, 34
  - hwloc\_obj\_is\_in\_subtree, 34
- hwlocality\_helper\_types
  - hwloc\_get\_type\_or\_above\_depth, 32
  - hwloc\_get\_type\_or\_below\_depth, 32
- hwlocality\_information
  - hwloc\_get\_depth\_type, 26
  - hwloc\_get\_nbobjs\_by\_depth, 26
  - hwloc\_get\_nbobjs\_by\_type, 27
  - hwloc\_get\_type\_depth, 27
  - hwloc\_topology\_get\_depth, 27
  - hwloc\_topology\_is\_thissystem, 27
  - HWLOC\_TYPE\_DEPTH\_MULTIPLE, 26
  - HWLOC\_TYPE\_DEPTH\_UNKNOWN, 26
- hwlocality\_linux\_cpumap
  - hwloc\_linux\_parse\_cpumap\_file, 50
- hwlocality\_linux\_libnuma\_bitmask
  - hwloc\_cpuset\_from\_linux\_libnuma\_bitmask, 52
  - hwloc\_cpuset\_to\_linux\_libnuma\_bitmask, 52
- hwlocality\_linux\_libnuma\_nodemask
  - hwloc\_cpuset\_from\_linux\_libnuma\_nodemask, 53
  - hwloc\_cpuset\_to\_linux\_libnuma\_nodemask, 53
- hwlocality\_linux\_libnuma\_ulong
  - hwloc\_cpuset\_from\_linux\_libnuma\_ulong, 51
  - hwloc\_cpuset\_to\_linux\_libnuma\_ulong, 51
- hwlocality\_objects
  - hwloc\_obj\_t, 20
- hwlocality\_topology
  - hwloc\_topology\_t, 17
- hwlocality\_traversal
  - hwloc\_get\_obj\_by\_depth, 28
  - hwloc\_get\_obj\_by\_type, 28
- hwlocality\_types
  - hwloc\_compare\_types, 19
  - hwloc\_obj\_type\_t, 18
  - HWLOC\_TYPE\_UNORDERED, 18
- last\_child
  - hwloc\_obj, 62
- logical\_index
  - hwloc\_obj, 62
- machine
  - hwloc\_obj\_attr\_u, 64
- memory\_kB
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, 55
  - hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, 56
  - hwloc\_obj\_attr\_u::hwloc\_memory\_attr\_s, 58
- misc
  - hwloc\_obj\_attr\_u, 65
- name
  - hwloc\_obj, 62
- next\_cousin
  - hwloc\_obj, 62
- next\_sibling
  - hwloc\_obj, 62
- node
  - hwloc\_obj\_attr\_u, 65
- Object Type Helpers, 32
- Object/String Conversion, 29
- OpenFabrics-Specific Functions, 54
- os\_index
  - hwloc\_obj, 62
- os\_level



---

- hwloc\_obj, [62](#)
- prev\_cousin
  - hwloc\_obj, [62](#)
- prev\_sibling
  - hwloc\_obj, [62](#)
- Retrieve Objects, [28](#)
- sibling\_rank
  - hwloc\_obj, [62](#)
- system
  - hwloc\_obj\_attr\_u, [65](#)
- The Cpuset API, [42](#)
- Topology context, [17](#)
- Topology Object Types, [18](#)
- Topology Objects, [20](#)
- type
  - hwloc\_obj, [62](#)
- userdata
  - hwloc\_obj, [63](#)