

# Hardware Locality (hwloc)

1.0

Generated by Doxygen 1.6.2

Mon May 17 16:39:34 2010



# Contents

<b>1</b>	<b>Hardware Locality</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	2
1.3	Examples . . . . .	3
1.4	Programming interface . . . . .	5
1.5	API example . . . . .	6
1.6	Questions and bugs . . . . .	10
1.7	History / credits . . . . .	10
<b>2</b>	<b>Terms and Definitions</b>	<b>11</b>
<b>3</b>	<b>Command-line tools</b>	<b>15</b>
3.1	lstopo . . . . .	16
3.2	hwloc-bind . . . . .	16
3.3	hwloc-calc . . . . .	16
3.4	hwloc-distrib . . . . .	17
<b>4</b>	<b>Environment variables</b>	<b>19</b>
<b>5</b>	<b>Interoperability with other software</b>	<b>21</b>
<b>6</b>	<b>Thread safety</b>	<b>23</b>
<b>7</b>	<b>Embedding hwloc in other software</b>	<b>25</b>
7.1	Using hwloc's m4 embedding capabilities . . . . .	26

7.2	Example embedding hwloc . . . . .	27
<b>8</b>	<b>Switching from PLPA to hwloc</b>	<b>29</b>
8.1	Topology context vs. caching . . . . .	30
8.2	Hierarchy vs. Core@Socket . . . . .	30
8.3	Logical vs. Physical/OS indexes . . . . .	30
8.4	Counting specification . . . . .	31
<b>9</b>	<b>Module Index</b>	<b>33</b>
9.1	Modules . . . . .	33
<b>10</b>	<b>Data Structure Index</b>	<b>35</b>
10.1	Data Structures . . . . .	35
<b>11</b>	<b>Module Documentation</b>	<b>37</b>
11.1	API version . . . . .	37
11.1.1	Define Documentation . . . . .	37
11.1.1.1	HWLOC_API_VERSION . . . . .	37
11.2	Topology context . . . . .	38
11.2.1	Typedef Documentation . . . . .	38
11.2.1.1	hwloc_topology_t . . . . .	38
11.3	Topology Object Types . . . . .	39
11.3.1	Enumeration Type Documentation . . . . .	39
11.3.1.1	hwloc_compare_types_e . . . . .	39
11.3.1.2	hwloc_obj_type_t . . . . .	39
11.3.2	Function Documentation . . . . .	40
11.3.2.1	hwloc_compare_types . . . . .	40
11.4	Topology Objects . . . . .	42
11.4.1	Typedef Documentation . . . . .	42
11.4.1.1	hwloc_obj_t . . . . .	42
11.5	Create and Destroy Topologies . . . . .	43
11.5.1	Function Documentation . . . . .	43
11.5.1.1	hwloc_topology_check . . . . .	43

---

11.5.1.2	<a href="#">hwloc_topology_destroy</a>	43
11.5.1.3	<a href="#">hwloc_topology_init</a>	44
11.5.1.4	<a href="#">hwloc_topology_load</a>	44
11.6	<a href="#">Configure Topology Detection</a>	45
11.6.1	<a href="#">Detailed Description</a>	46
11.6.2	<a href="#">Enumeration Type Documentation</a>	46
11.6.2.1	<a href="#">hwloc_topology_flags_e</a>	46
11.6.3	<a href="#">Function Documentation</a>	47
11.6.3.1	<a href="#">hwloc_topology_get_support</a>	47
11.6.3.2	<a href="#">hwloc_topology_ignore_all_keep_structure</a>	47
11.6.3.3	<a href="#">hwloc_topology_ignore_type</a>	47
11.6.3.4	<a href="#">hwloc_topology_ignore_type_keep_structure</a>	48
11.6.3.5	<a href="#">hwloc_topology_set_flags</a>	48
11.6.3.6	<a href="#">hwloc_topology_set_fsroot</a>	48
11.6.3.7	<a href="#">hwloc_topology_set_pid</a>	48
11.6.3.8	<a href="#">hwloc_topology_set_synthetic</a>	49
11.6.3.9	<a href="#">hwloc_topology_set_xml</a>	49
11.7	<a href="#">Tinker with topologies.</a>	50
11.7.1	<a href="#">Function Documentation</a>	50
11.7.1.1	<a href="#">hwloc_topology_export_xml</a>	50
11.7.1.2	<a href="#">hwloc_topology_insert_misc_object_by_cpuset</a>	50
11.7.1.3	<a href="#">hwloc_topology_insert_misc_object_by_parent</a>	51
11.8	<a href="#">Get some Topology Information</a>	52
11.8.1	<a href="#">Enumeration Type Documentation</a>	52
11.8.1.1	<a href="#">hwloc_get_type_depth_e</a>	52
11.8.2	<a href="#">Function Documentation</a>	53
11.8.2.1	<a href="#">hwloc_get_depth_type</a>	53
11.8.2.2	<a href="#">hwloc_get_nbobjs_by_depth</a>	53
11.8.2.3	<a href="#">hwloc_get_nbobjs_by_type</a>	53
11.8.2.4	<a href="#">hwloc_get_type_depth</a>	53
11.8.2.5	<a href="#">hwloc_topology_get_depth</a>	53

11.8.2.6 hwloc_topology_is_thissystem . . . . .	53
11.9 Retrieve Objects . . . . .	55
11.9.1 Function Documentation . . . . .	55
11.9.1.1 hwloc_get_obj_by_depth . . . . .	55
11.9.1.2 hwloc_get_obj_by_type . . . . .	55
11.10 Object/String Conversion . . . . .	56
11.10.1 Function Documentation . . . . .	56
11.10.1.1 hwloc_obj_attr_snprintf . . . . .	56
11.10.1.2 hwloc_obj_cpuset_snprintf . . . . .	57
11.10.1.3 hwloc_obj_snprintf . . . . .	57
11.10.1.4 hwloc_obj_type_of_string . . . . .	57
11.10.1.5 hwloc_obj_type_snprintf . . . . .	58
11.10.1.6 hwloc_obj_type_string . . . . .	58
11.11 Binding . . . . .	59
11.11.1 Detailed Description . . . . .	59
11.11.2 Enumeration Type Documentation . . . . .	60
11.11.2.1 hwloc_cpubind_policy_t . . . . .	60
11.11.3 Function Documentation . . . . .	61
11.11.3.1 hwloc_get_cpubind . . . . .	61
11.11.3.2 hwloc_get_proc_cpubind . . . . .	61
11.11.3.3 hwloc_get_thread_cpubind . . . . .	61
11.11.3.4 hwloc_set_cpubind . . . . .	62
11.11.3.5 hwloc_set_proc_cpubind . . . . .	62
11.11.3.6 hwloc_set_thread_cpubind . . . . .	62
11.12 Object Type Helpers . . . . .	63
11.12.1 Function Documentation . . . . .	63
11.12.1.1 hwloc_get_type_or_above_depth . . . . .	63
11.12.1.2 hwloc_get_type_or_below_depth . . . . .	63
11.13 Basic Traversal Helpers . . . . .	64
11.13.1 Function Documentation . . . . .	65
11.13.1.1 hwloc_get_ancestor_obj_by_depth . . . . .	65

11.13.1.2 hwloc_get_ancestor_obj_by_type . . . . .	65
11.13.1.3 hwloc_get_common_ancestor_obj . . . . .	65
11.13.1.4 hwloc_get_next_child . . . . .	65
11.13.1.5 hwloc_get_next_obj_by_depth . . . . .	65
11.13.1.6 hwloc_get_next_obj_by_type . . . . .	65
11.13.1.7 hwloc_get_pu_obj_by_os_index . . . . .	66
11.13.1.8 hwloc_get_root_obj . . . . .	66
11.13.1.9 hwloc_obj_is_in_subtree . . . . .	66
11.14 Finding Objects Inside a CPU set . . . . .	67
11.14.1 Function Documentation . . . . .	68
11.14.1.1 hwloc_get_first_largest_obj_inside_cpuset . . . . .	68
11.14.1.2 hwloc_get_largest_objs_inside_cpuset . . . . .	68
11.14.1.3 hwloc_get_nbobjs_inside_cpuset_by_depth . . . . .	68
11.14.1.4 hwloc_get_nbobjs_inside_cpuset_by_type . . . . .	68
11.14.1.5 hwloc_get_next_obj_inside_cpuset_by_depth . . . . .	69
11.14.1.6 hwloc_get_next_obj_inside_cpuset_by_type . . . . .	69
11.14.1.7 hwloc_get_obj_inside_cpuset_by_depth . . . . .	69
11.14.1.8 hwloc_get_obj_inside_cpuset_by_type . . . . .	69
11.15 Finding a single Object covering at least CPU set . . . . .	70
11.15.1 Function Documentation . . . . .	70
11.15.1.1 hwloc_get_child_covering_cpuset . . . . .	70
11.15.1.2 hwloc_get_obj_covering_cpuset . . . . .	70
11.16 Finding a set of similar Objects covering at least a CPU set . . . . .	71
11.16.1 Function Documentation . . . . .	71
11.16.1.1 hwloc_get_next_obj_covering_cpuset_by_depth . . . . .	71
11.16.1.2 hwloc_get_next_obj_covering_cpuset_by_type . . . . .	71
11.17 Cache-specific Finding Helpers . . . . .	72
11.17.1 Function Documentation . . . . .	72
11.17.1.1 hwloc_get_cache_covering_cpuset . . . . .	72
11.17.1.2 hwloc_get_shared_cache_covering_obj . . . . .	72
11.18 Advanced Traversal Helpers . . . . .	73

11.18.1 Function Documentation . . . . .	73
11.18.1.1 hwloc_get_closest_objs . . . . .	73
11.18.1.2 hwloc_get_obj_below_array_by_type . . . . .	73
11.18.1.3 hwloc_get_obj_below_by_type . . . . .	74
11.19 Binding Helpers . . . . .	75
11.19.1 Function Documentation . . . . .	75
11.19.1.1 hwloc_distribute . . . . .	75
11.20 Cpuset Helpers . . . . .	76
11.20.1 Function Documentation . . . . .	76
11.20.1.1 hwloc_topology_get_allowed_cpuset . . . . .	76
11.20.1.2 hwloc_topology_get_complete_cpuset . . . . .	76
11.20.1.3 hwloc_topology_get_online_cpuset . . . . .	76
11.20.1.4 hwloc_topology_get_topology_cpuset . . . . .	77
11.21 The Cpuset API . . . . .	78
11.21.1 Detailed Description . . . . .	81
11.21.2 Define Documentation . . . . .	82
11.21.2.1 hwloc_cpuset_foreach_begin . . . . .	82
11.21.2.2 hwloc_cpuset_foreach_end . . . . .	82
11.21.3 Typedef Documentation . . . . .	82
11.21.3.1 hwloc_const_cpuset_t . . . . .	82
11.21.3.2 hwloc_cpuset_t . . . . .	82
11.21.4 Function Documentation . . . . .	82
11.21.4.1 hwloc_cpuset_all_but_cpu . . . . .	82
11.21.4.2 hwloc_cpuset_alloc . . . . .	82
11.21.4.3 hwloc_cpuset_and . . . . .	83
11.21.4.4 hwloc_cpuset_andnot . . . . .	83
11.21.4.5 hwloc_cpuset_asprintf . . . . .	83
11.21.4.6 hwloc_cpuset_clr . . . . .	83
11.21.4.7 hwloc_cpuset_clr_range . . . . .	83
11.21.4.8 hwloc_cpuset_compare . . . . .	83
11.21.4.9 hwloc_cpuset_compare_first . . . . .	83



11.21.4.10	<code>hwloc_cpuset_copy</code>	84
11.21.4.11	<code>hwloc_cpuset_cpu</code>	84
11.21.4.12	<code>hwloc_cpuset_dup</code>	84
11.21.4.13	<code>hwloc_cpuset_fill</code>	84
11.21.4.14	<code>hwloc_cpuset_first</code>	84
11.21.4.15	<code>hwloc_cpuset_free</code>	84
11.21.4.16	<code>hwloc_cpuset_from_ith_ulong</code>	84
11.21.4.17	<code>hwloc_cpuset_from_string</code>	85
11.21.4.18	<code>hwloc_cpuset_from_ulong</code>	85
11.21.4.19	<code>hwloc_cpuset_intersects</code>	85
11.21.4.20	<code>hwloc_cpuset_isequal</code>	85
11.21.4.21	<code>hwloc_cpuset_isfull</code>	85
11.21.4.22	<code>hwloc_cpuset_isincluded</code>	85
11.21.4.23	<code>hwloc_cpuset_isset</code>	85
11.21.4.24	<code>hwloc_cpuset_iszero</code>	85
11.21.4.25	<code>hwloc_cpuset_last</code>	86
11.21.4.26	<code>hwloc_cpuset_next</code>	86
11.21.4.27	<code>hwloc_cpuset_not</code>	86
11.21.4.28	<code>hwloc_cpuset_or</code>	86
11.21.4.29	<code>hwloc_cpuset_set</code>	86
11.21.4.30	<code>hwloc_cpuset_set_range</code>	86
11.21.4.31	<code>hwloc_cpuset_singlify</code>	87
11.21.4.32	<code>hwloc_cpuset_snprintf</code>	87
11.21.4.33	<code>hwloc_cpuset_to_ith_ulong</code>	87
11.21.4.34	<code>hwloc_cpuset_to_ulong</code>	87
11.21.4.35	<code>hwloc_cpuset_weight</code>	87
11.21.4.36	<code>hwloc_cpuset_xor</code>	87
11.21.4.37	<code>hwloc_cpuset_zero</code>	88
11.22	Helpers for manipulating glibc sched affinity	89
11.22.1	Function Documentation	89
11.22.1.1	<code>hwloc_cpuset_from_glibc_sched_affinity</code>	89

11.22.1.2 hwloc_cpuset_to_glibc_sched_affinity . . . . .	89
11.23 Linux-only helpers . . . . .	90
11.23.1 Detailed Description . . . . .	90
11.23.2 Function Documentation . . . . .	90
11.23.2.1 hwloc_linux_get_tid_cpupbind . . . . .	90
11.23.2.2 hwloc_linux_parse_cpumap_file . . . . .	90
11.23.2.3 hwloc_linux_set_tid_cpupbind . . . . .	90
11.24 Helpers for manipulating Linux libnuma unsigned long masks . . . . .	91
11.24.1 Function Documentation . . . . .	91
11.24.1.1 hwloc_cpuset_from_linux_libnuma_ulongs . . . . .	91
11.24.1.2 hwloc_cpuset_to_linux_libnuma_ulongs . . . . .	91
11.25 Helpers for manipulating Linux libnuma bitmask . . . . .	92
11.25.1 Function Documentation . . . . .	92
11.25.1.1 hwloc_cpuset_from_linux_libnuma_bitmask . . . . .	92
11.25.1.2 hwloc_cpuset_to_linux_libnuma_bitmask . . . . .	92
11.26 Helpers for manipulating Linux libnuma nodemask_t . . . . .	93
11.26.1 Function Documentation . . . . .	93
11.26.1.1 hwloc_cpuset_from_linux_libnuma_nodemask . . . . .	93
11.26.1.2 hwloc_cpuset_to_linux_libnuma_nodemask . . . . .	93
11.27 OpenFabrics-Specific Functions . . . . .	94
11.27.1 Function Documentation . . . . .	94
11.27.1.1 hwloc_ibv_get_device_cpuset . . . . .	94
<b>12 Data Structure Documentation . . . . .</b>	<b>95</b>
12.1 hwloc_obj_attr_u::hwloc_cache_attr_s Struct Reference . . . . .	95
12.1.1 Detailed Description . . . . .	95
12.1.2 Field Documentation . . . . .	95
12.1.2.1 depth . . . . .	95
12.1.2.2 size . . . . .	96
12.2 hwloc_obj_attr_u::hwloc_group_attr_s Struct Reference . . . . .	97
12.2.1 Detailed Description . . . . .	97

---

12.2.2	Field Documentation . . . . .	97
12.2.2.1	depth . . . . .	97
12.3	hwloc_obj_attr_u::hwloc_machine_attr_s Struct Reference . . . . .	98
12.3.1	Detailed Description . . . . .	98
12.3.2	Field Documentation . . . . .	98
12.3.2.1	dmi_board_name . . . . .	98
12.3.2.2	dmi_board_vendor . . . . .	98
12.4	hwloc_obj Struct Reference . . . . .	99
12.4.1	Detailed Description . . . . .	101
12.4.2	Field Documentation . . . . .	101
12.4.2.1	allowed_cpuset . . . . .	101
12.4.2.2	allowed_nodeuset . . . . .	101
12.4.2.3	arity . . . . .	101
12.4.2.4	attr . . . . .	101
12.4.2.5	children . . . . .	101
12.4.2.6	complete_cpuset . . . . .	102
12.4.2.7	complete_nodeuset . . . . .	102
12.4.2.8	cpuset . . . . .	102
12.4.2.9	depth . . . . .	102
12.4.2.10	first_child . . . . .	103
12.4.2.11	last_child . . . . .	103
12.4.2.12	logical_index . . . . .	103
12.4.2.13	memory . . . . .	103
12.4.2.14	name . . . . .	103
12.4.2.15	next_cousin . . . . .	103
12.4.2.16	next_sibling . . . . .	103
12.4.2.17	nodeuset . . . . .	103
12.4.2.18	online_cpuset . . . . .	104
12.4.2.19	os_index . . . . .	104
12.4.2.20	os_level . . . . .	104
12.4.2.21	parent . . . . .	104

12.4.2.22	<a href="#">prev_cousin</a>	104
12.4.2.23	<a href="#">prev_sibling</a>	104
12.4.2.24	<a href="#">sibling_rank</a>	104
12.4.2.25	<a href="#">type</a>	104
12.4.2.26	<a href="#">userdata</a>	105
12.5	<a href="#">hwloc_obj_attr_u Union Reference</a>	106
12.5.1	<a href="#">Detailed Description</a>	106
12.5.2	<a href="#">Field Documentation</a>	106
12.5.2.1	<a href="#">cache</a>	106
12.5.2.2	<a href="#">group</a>	107
12.5.2.3	<a href="#">machine</a>	107
12.6	<a href="#">hwloc_obj_memory_s::hwloc_obj_memory_page_type_s Struct Reference</a>	108
12.6.1	<a href="#">Detailed Description</a>	108
12.6.2	<a href="#">Field Documentation</a>	108
12.6.2.1	<a href="#">count</a>	108
12.6.2.2	<a href="#">size</a>	108
12.7	<a href="#">hwloc_obj_memory_s Struct Reference</a>	109
12.7.1	<a href="#">Detailed Description</a>	109
12.7.2	<a href="#">Field Documentation</a>	109
12.7.2.1	<a href="#">local_memory</a>	109
12.7.2.2	<a href="#">page_types</a>	109
12.7.2.3	<a href="#">page_types_len</a>	110
12.7.2.4	<a href="#">total_memory</a>	110
12.8	<a href="#">hwloc_topology_cpupbind_support Struct Reference</a>	111
12.8.1	<a href="#">Detailed Description</a>	111
12.8.2	<a href="#">Field Documentation</a>	111
12.8.2.1	<a href="#">get_proc_cpupbind</a>	111
12.8.2.2	<a href="#">get_thisproc_cpupbind</a>	111
12.8.2.3	<a href="#">get_thisthread_cpupbind</a>	111
12.8.2.4	<a href="#">get_thread_cpupbind</a>	111

---

12.8.2.5	set_proc_cpubind . . . . .	112
12.8.2.6	set_thisproc_cpubind . . . . .	112
12.8.2.7	set_thisthread_cpubind . . . . .	112
12.8.2.8	set_thread_cpubind . . . . .	112
12.9	hwloc_topology_discovery_support Struct Reference . . . . .	113
12.9.1	Detailed Description . . . . .	113
12.9.2	Field Documentation . . . . .	113
12.9.2.1	pu . . . . .	113
12.10	hwloc_topology_support Struct Reference . . . . .	114
12.10.1	Detailed Description . . . . .	114
12.10.2	Field Documentation . . . . .	114
12.10.2.1	cpubind . . . . .	114
12.10.2.2	discovery . . . . .	114



# Chapter 1

## Hardware Locality

### 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 processing units (logical processors or "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. See [Switching from PLPA to hwloc](#) for more details about converting your application from PLPA to hwloc.

hwloc supports the following operating systems:

- Linux (including old kernels not having sysfs topology information, with knowledge of cpusets, offline cpus, ScaleMP vSMP, and Kerrighed support)
- Solaris

- AIX
- Darwin / OS X
- FreeBSD and its variants, such as kFreeBSD/GNU
- 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  $\geq 2.63$ , Automake  $\geq 1.10$  and Libtool  $\geq 2.2.6$  are required when building from a Subversion checkout.

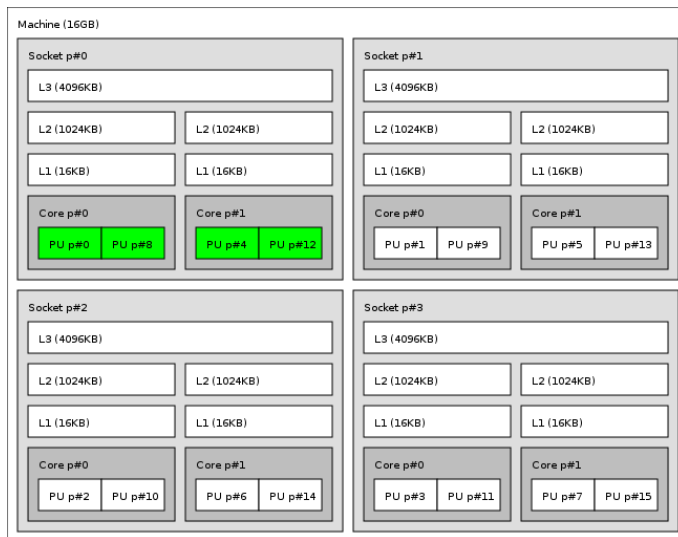
Installation by itself is the fairly common GNU-based process:

```
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:



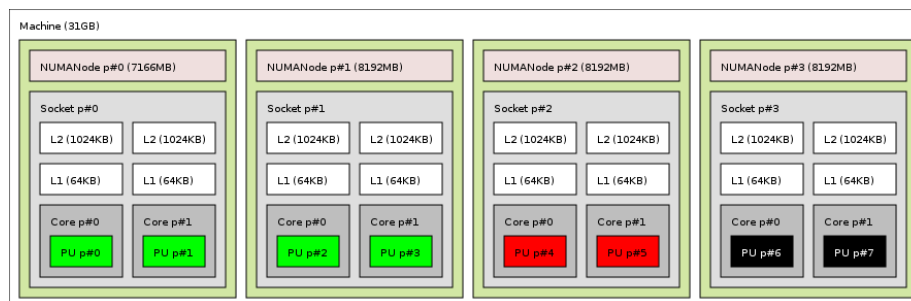
```
Machine (16GB)
  Socket #0 + L3 #0 (4096KB)
    L2 #0 (1024KB) + L1 #0 (16KB) + Core #0
      PU #0 (phys=0)
      PU #1 (phys=8)
    L2 #1 (1024KB) + L1 #1 (16KB) + Core #1
      PU #2 (phys=4)
      PU #3 (phys=12)
  Socket #1 + L3 #1 (4096KB)
```

```

L2 #2 (1024KB) + L1 #2 (16KB) + Core #2
  PU #4 (phys=1)
  PU #5 (phys=9)
L2 #3 (1024KB) + L1 #3 (16KB) + Core #3
  PU #6 (phys=5)
  PU #7 (phys=13)
Socket #2 + L3 #2 (4096KB)
  L2 #4 (1024KB) + L1 #4 (16KB) + Core #4
    PU #8 (phys=2)
    PU #9 (phys=10)
  L2 #5 (1024KB) + L1 #5 (16KB) + Core #5
    PU #10 (phys=6)
    PU #11 (phys=14)
Socket #3 + L3 #3 (4096KB)
  L2 #6 (1024KB) + L1 #6 (16KB) + Core #6
    PU #12 (phys=3)
    PU #13 (phys=11)
  L2 #7 (1024KB) + L1 #7 (16KB) + Core #7
    PU #14 (phys=7)
    PU #15 (phys=15)

```

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



```

Machine (64GB)
  NUMANode #0 (phys=0 8190MB) + Socket #0
    L2 #0 (1024KB) + L1 #0 (64KB) + Core #0 + PU #0 (phys=0)
    L2 #1 (1024KB) + L1 #1 (64KB) + Core #1 + PU #1 (phys=1)
  NUMANode #1 (phys=1 8192MB) + Socket #1
    L2 #2 (1024KB) + L1 #2 (64KB) + Core #2 + PU #2 (phys=2)
    L2 #3 (1024KB) + L1 #3 (64KB) + Core #3 + PU #3 (phys=3)
  NUMANode #2 (phys=2 8192MB) + Socket #2
    L2 #4 (1024KB) + L1 #4 (64KB) + Core #4 + PU #4 (phys=4)
    L2 #5 (1024KB) + L1 #5 (64KB) + Core #5 + PU #5 (phys=5)
  NUMANode #3 (phys=3 8192MB) + Socket #3
    L2 #6 (1024KB) + L1 #6 (64KB) + Core #6 + PU #6 (phys=6)
    L2 #7 (1024KB) + L1 #7 (64KB) + Core #7 + PU #7 (phys=7)
  NUMANode #4 (phys=4 8192MB) + Socket #4
    L2 #8 (1024KB) + L1 #8 (64KB) + Core #8 + PU #8 (phys=8)
    L2 #9 (1024KB) + L1 #9 (64KB) + Core #9 + PU #9 (phys=9)
  NUMANode #5 (phys=5 8192MB) + Socket #5
    L2 #10 (1024KB) + L1 #10 (64KB) + Core #10 + PU #10 (phys=10)

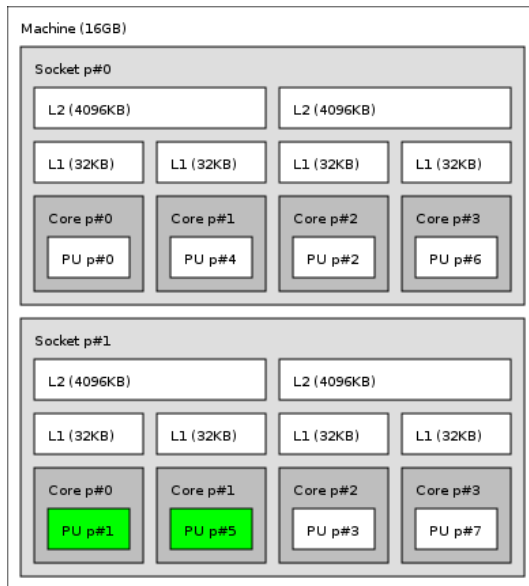
```

```

L2 #11 (1024KB) + L1 #11 (64KB) + Core #11 + PU #11 (phys=11)
NUMANode #6 (phys=6 8192MB) + Socket #6
L2 #12 (1024KB) + L1 #12 (64KB) + Core #12 + PU #12 (phys=12)
L2 #13 (1024KB) + L1 #13 (64KB) + Core #13 + PU #13 (phys=13)
NUMANode #7 (phys=7 8192MB) + Socket #7
L2 #14 (1024KB) + L1 #14 (64KB) + Core #14 + PU #14 (phys=14)
L2 #15 (1024KB) + L1 #15 (64KB) + Core #15 + PU #15 (phys=15)

```

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



```

Machine (16GB)
Socket #0
  L2 #0 (4096KB)
    L1 #0 (32KB) + Core #0 + PU #0 (phys=0)
    L1 #1 (32KB) + Core #1 + PU #1 (phys=4)
  L2 #1 (4096KB)
    L1 #2 (32KB) + Core #2 + PU #2 (phys=2)
    L1 #3 (32KB) + Core #3 + PU #3 (phys=6)
Socket #1
  L2 #2 (4096KB)
    L1 #4 (32KB) + Core #4 + PU #4 (phys=1)
    L1 #5 (32KB) + Core #5 + PU #5 (phys=5)
  L2 #3 (4096KB)
    L1 #6 (32KB) + Core #6 + PU #6 (phys=3)
    L1 #7 (32KB) + Core #7 + PU #7 (phys=7)

```

## 1.4 Programming interface

The basic interface is available in [hwloc.h](#). It essentially offers low-level routines for advanced programmers that want to manually manipulate objects and follow links

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

Each object contains a cpuset describing the list of processing units 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. See the [Interoperability with other software](#) section for details.

To precisely define the vocabulary used by hwloc, a [Terms and Definitions](#) section 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 API example

The following small C example (named "hwloc-hello.c") prints the topology of the machine and bring the process to the first logical processor of the second core of the machine.

```
/* Example hwloc API program.
 *
 * Copyright © 2009 INRIA, Université Bordeaux 1
 * Copyright © 2009-2010 Cisco Systems, Inc. All rights reserved.
 *
 * hwloc-hello.c
 */

#include <hwloc.h>

static void print_children(hwloc_topology_t topology, hwloc_obj_t obj,
                          int depth)
{
    char string[128];
    unsigned 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)
{
    int depth;
    unsigned i;
    unsigned long size;
    int levels;
    char string[128];
    int topodepth;
    hwloc_topology_t topology;
    hwloc_cpuset_t cpuset;
    hwloc_obj_t obj;

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

    /* ... Optionally, put detection configuration here to 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. */
    topodepth = hwloc_topology_get_depth(topology);

    /******
     * First example:
     * Walk the topology with an array style, from level 0 (always
     * the system level) to the lowest level (always the proc level).
     *****/
    for (depth = 0; depth < topodepth; depth++) {
        printf("*** Objects at level %d\n", 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("Index %u: %s\n", i, string);
        }
    }

    /******
     * Second example:
     * Walk the topology with a tree style.
     *****/
    printf("*** Printing overall tree\n");
    print_children(topology, hwloc_get_root_obj(topology), 0);

    /******
     * Third example:
     * 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_nobjs_by_depth(topology, depth));
}

/*****
 * Fourth example:
 * Compute the amount of cache that the first logical processor
 * has above it.
 *****/
levels = 0;
size = 0;
for (obj = hwloc_get_obj_by_type(topology, HWLOC_OBJ_PU, 0);
     obj;
     obj = obj->parent)
    if (obj->type == HWLOC_OBJ_CACHE) {
        levels++;
        size += obj->attr->cache.size;
    }
printf("*** Logical processor 0 has %d caches totaling %luKB\n",
       levels, size / 1024);

/*****
 * Fifth example:
 * Bind to only one thread of the last core of the machine.
 *
 * First find out where cores are, or else smaller sets of CPUs if
 * the OS doesn't have the notion of a "core".
 *****/
depth = hwloc_get_type_or_below_depth(topology, HWLOC_OBJ_CORE);

/* Get last core. */
obj = hwloc_get_obj_by_depth(topology, depth,
                             hwloc_get_nobjs_by_depth(topology, depth) - 1);
if (obj) {
    /* Get a copy of its cpuset that we may modify. */
    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_cpupbind(topology, cpuset, 0)) {
        char *str;
        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;
}
```

hwloc provides a `pkg-config` executable to obtain relevant compiler and linker flags. For example, it can be used thusly to compile applications that utilize the hwloc library (assuming GNU Make):

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

On a machine with 4GB of RAM and 2 processor sockets -- each socket of which has two processing cores -- the output from running `hwloc-hello` could be something like the following:

```
shell$ ./hwloc-hello
*** Objects at level 0
Index 0: Machine(3938MB)
*** Objects at level 1
Index 0: Socket#0
Index 1: Socket#1
*** Objects at level 2
Index 0: Core#0
Index 1: Core#1
Index 2: Core#3
Index 3: Core#2
*** Objects at level 3
Index 0: PU#0
Index 1: PU#1
Index 2: PU#2
Index 3: PU#3
*** Printing overall tree
Machine(3938MB)
  Socket#0
    Core#0
      PU#0
    Core#1
      PU#1
  Socket#1
    Core#3
      PU#2
    Core#2
      PU#3
*** 2 socket(s)
shell$
```

## 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/>).

If hwloc discovers an incorrect topology for your machine, the very first thing you should check is to ensure that you have the most recent updates installed for your operating system. Indeed, most of hwloc topology discovery relies on hardware information retrieved through the operation system (e.g., via the /sys virtual filesystem of the Linux kernel). If upgrading your OS or Linux kernel does not solve your problem, you may also want to ensure that you are running the most recent version of the BIOS for your machine.

If those things fail, contact us on the mailing list for additional help.

## 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**

# **Terms and Definitions**

**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\\_t](#) enumeration.

They are topologically sorted by CPU set into a tree.

**CPU set** The set of logical processors (or processing units) logically included in an object (if it makes sense). They are always expressed using physical logical processor numbers (as announced by the OS). They are just masks, they do *not* have any relation with an operating system actual binding notion like Linux' cpusets.

**Parent object** The object logically containing the current object, for example because its CPU set includes the CPU set of the current object.

**Ancestor object** The parent object, or its own parent object, and so on.

**Children object(s)** The object (or objects) 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 parent.

**Sibling rank** Index to uniquely identify objects of the same type which have the same parent, and is always in the range [0, parent\_arity).

**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 operating system (OS) uses to identify the object. This may be completely arbitrary, or it may depend on the BIOS configuration.

**Depth** Nesting level in the object tree, starting from the 0th object.

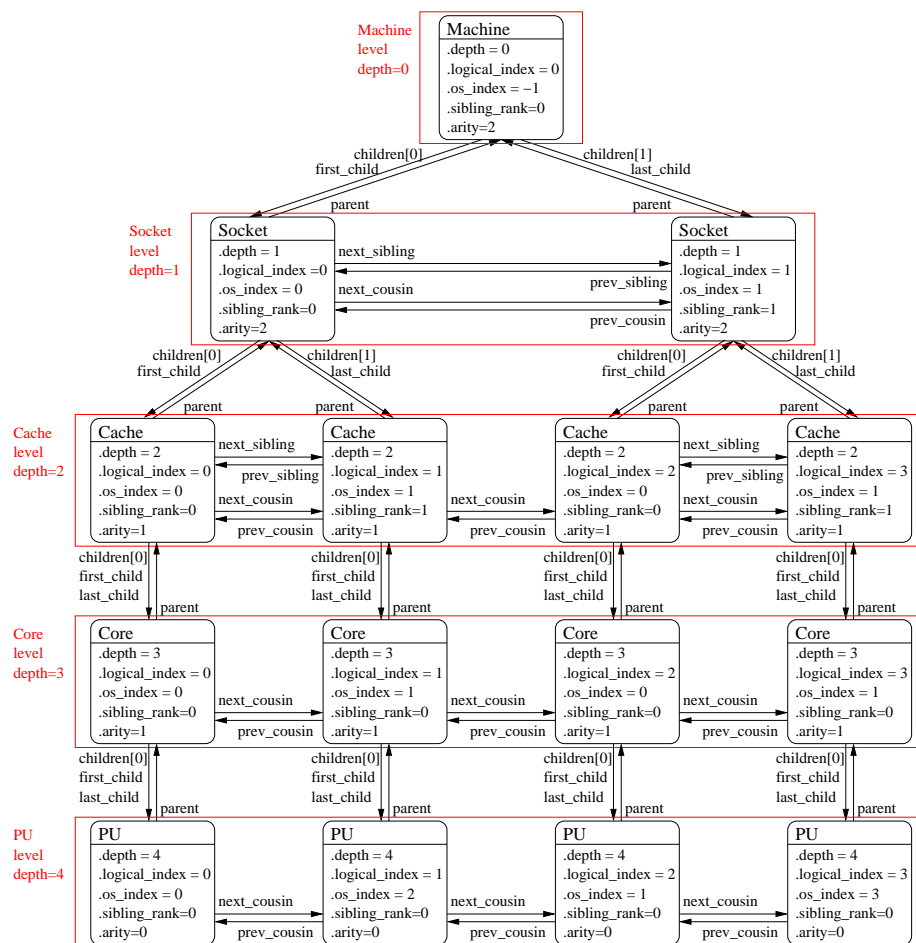
**Logical index** Index to uniquely identify objects of the same type. It expresses proximity in a generic way. This index is always linear and in the range [0, num\_objs\_same\_type\_same\_level). Think of it as "cousin rank." The ordering is based on topology first, and then on OS CPU numbers, so it is stable across everything except firmware CPU renumbering.

### Logical processor

**Processing unit** The smallest processing element that can be represented by a hwloc object. It may be a single-core processor, a core of a multicore processor, or a single thread in SMT processor.

The following diagram can help to understand the vocabulary of the relationships by showing the example of a machine with two dual core sockets (with no hardware threads); thus, a topology with 4 levels. Each box with rounded corner corresponds to

one `hwloc_obj_t`, containing the values of the different integer fields (depth, logical\_index, etc.), and arrows show to which other `hwloc_obj_t` pointers point to (first\_child, parent, etc.)



It should be noted that for PU objects, the logical index -- as computed linearly by hwloc -- is not the same as the OS index.



## **Chapter 3**

# **Command-line tools**

hwloc comes with an extensive C programming interface and several command line utilities. Each of them is fully documented in its own manual page; the following is a summary of the available command line tools.

### 3.1 lstopo

lstopo (also known as hwloc-info and hwloc-ls) displays the hierarchical topology map of the current system. The output may be graphic or textual, and can also be exported to numerous file formats such as PDF, PNG, XML, and others.

Note that lstopo can read XML files and/or alternate chroot filesystems and display topological maps representing those systems (e.g., use lstopo to output an XML file on one system, and then use lstopo to read in that XML file and display it on a different system).

### 3.2 hwloc-bind

hwloc-bind binds processes to specific hardware objects through a flexible syntax. A simple example is binding an executable to specific cores (or sockets or cpusets or ...). The hwloc-bind(1) man page provides much more detail on what is possible.

hwloc-bind can also be used to retrieve the current process' binding.

### 3.3 hwloc-calc

hwloc-calc is generally used to create cpuset strings to pass to hwloc-bind. Although hwloc-bind accepts many forms of object specification (i.e., cpuset strings are one of many forms that hwloc-bind understands), they can be useful, compact representations in shell scripts, for example.

hwloc-calc generates cpuset strings from given hardware objects with the ability to aggregate them, intersect them, and more. hwloc-calc generally uses the same syntax than hwloc-bind, but multiple instances may be composed to generate complex combinations.

Note that hwloc-calc can also generate lists of logical processors or NUMA nodes that are convenient to pass to some external tools such as taskset or numactl.

## 3.4 hwloc-distrib

hwloc-distrib generates a set of cpuset strings that are uniformly distributed across the machine for the given number of processes. These strings may be used with hwloc-bind to run processes to maximize their memory bandwidth by properly distributing them across the machine.





## **Chapter 4**

# **Environment variables**

The behavior of the hwloc library and tools may be tuned thanks to the following environment variables.

**HWLOC\_XMLFILE=/path/to/file.xml** enforces the discovery from the given XML file as if [hwloc\\_topology\\_set\\_xml\(\)](#) had been called. This file may have been generated earlier with `lstopo file.xml`. For convenience, this backend provides empty binding hooks which just return success. To have hwloc still actually call OS-specific hooks, **HWLOC\_THISSYSTEM** should be set 1 in the environment too, to assert that the loaded file is really the underlying system.

**HWLOC\_FSROOT=/path/to/linux/filesystem-root/** switches to reading the topology from the specified Linux filesystem root instead of the main file-system root, as if [hwloc\\_topology\\_set\\_fsroot\(\)](#) had been called. Not using the main file-system root causes [hwloc\\_topology\\_is\\_thissystem\(\)](#) to return 0. For convenience, this backend provides empty binding hooks which just return success. To have hwloc still actually call OS-specific hooks, **HWLOC\_THISSYSTEM** should be set 1 in the environment too, to assert that the loaded file is really the underlying system.

**HWLOC\_THISSYSTEM=1** enforces the return value of [hwloc\\_topology\\_is\\_thissystem\(\)](#). It means that it makes hwloc assume that the selected backend provides the topology for the system on which we are running, even if it is not the OS-specific backend but the XML backend for instance. This means making the binding functions actually call the OS-specific system calls and really do binding, while the XML backend would otherwise provide empty hooks just returning success. This can be used for efficiency reasons to first detect the topology once, save it to an XML file, and quickly reload it later through the XML backend, but still having binding functions actually do bind.

## **Chapter 5**

# **Interoperability with other software**

Although hwloc offers its own portable interface, it still may have to interoperate with specific or non-portable libraries that manipulate similar kinds of objects. hwloc therefore offers several specific "helpers" to assist converting between those specific interfaces and hwloc.

Some external libraries may be specific to a particular OS; others may not always be available. The hwloc core therefore generally does not explicitly depend on these types of libraries. However, when a custom application uses or otherwise depends on such a library, it may optionally include the corresponding hwloc helper to extend the hwloc interface with dedicated helpers.

**Linux specific features** [hwloc/linux.h](#) offers Linux-specific helpers that utilize some non-portable features of the Linux system, such as binding threads through their thread ID ("tid") or parsing kernel CPU mask files.

**Linux libnuma** [hwloc/linux-libnuma.h](#) provides conversion helpers between hwloc CPU sets and libnuma-specific types, such as nodemasks and bitmasks. It helps you use libnuma memory-binding functions with hwloc CPU sets.

**Glibc** [hwloc/glibc-sched.h](#) offers conversion routines between Glibc and hwloc CPU sets in order to use hwloc with functions such as `sched_setaffinity()`.

**OpenFabrics Verbs** [hwloc/openfabrics-verbs.h](#) helps interoperability with the OpenFabrics Verbs interface. For example, it can return a list of processors near an OpenFabrics device.

## **Chapter 6**

# **Thread safety**

Like most libraries that mainly fill data structures, hwloc is not thread safe but rather reentrant: all state is held in a `hwloc_topology_t` instance without mutex protection. That means, for example, that two threads can safely operate on and modify two different `hwloc_topology_t` instances, but they should not simultaneously invoke functions that modify the *same* instance. Similarly, one thread should not modify a `hwloc_topology_t` instance while another thread is reading or traversing it. However, two threads can safely read or traverse the same `hwloc_topology_t` instance concurrently.

When running in multiprocessor environments, be aware that proper thread synchronization and/or memory coherency protection is needed to pass hwloc data (such as `hwloc_topology_t` pointers) from one processor to another (e.g., a mutex, semaphore, or a memory barrier). Note that this is not a hwloc-specific requirement, but it is worth mentioning.

For reference, `hwloc_topology_t` modification operations include (but may not be limited to):

**Creation and destruction** `hwloc_topology_init()`, `hwloc_topology_load()`, `hwloc_topology_destroy()` (see [Create and Destroy Topologies](#)) imply major modifications of the structure, including freeing some objects. No other thread cannot access the topology or any of its objects at the same time.

Also references to objects inside the topology are not valid anymore after these functions return.

**Runtime topology modifications** `hwloc_topology_insert_misc_object_by_*` (see [Tinker with topologies](#).) may modify the topology significantly by adding objects inside the tree, changing the topology depth, etc.

Although references to former objects *may* still be valid after insertion, it is strongly advised to not rely on any such guarantee and always re-consult the topology to reacquire new instances of objects.

**Locating topologies** `hwloc_topology_ignore*`, `hwloc_topology_set*` (see [Configure Topology Detection](#)) do not modify the topology directly, but they do modify internal structures describing the behavior of the next invocation of `hwloc_topology_load()`. Hence, all of these functions should not be used concurrently.

Note that these functions do not modify the current topology until it is actually reloaded; it is possible to use them while other threads are only read the current topology.

## **Chapter 7**

# **Embedding hwloc in other software**

It can be desirable to include hwloc in a larger software package (be sure to check out the LICENSE file) so that users don't have to separately download and install it before installing your software. This can be advantageous to ensure that your software uses a known-tested/good version of hwloc, or for use on systems that do not have hwloc pre-installed.

When used in "embedded" mode, hwloc will:

- not install any header files
- not build any documentation files
- not build or install any executables or tests
- not build `libhwloc.*` -- instead, it will build `libhwloc_embedded.*`

There are two ways to put hwloc into "embedded" mode. The first is directly from the configure command line:

```
shell$ ./configure --enable-embedded-mode ...
```

The second requires that your software project uses the GNU Autoconf / Automake / Libtool tool chain to build your software. If you do this, you can directly integrate hwloc's m4 configure macro into your configure script. You can then invoke hwloc's configuration tests and build setup by calling an m4 macro (see below).

## 7.1 Using hwloc's m4 embedding capabilities

Every project is different, and there are many different ways of integrating hwloc into yours. What follows is *one* example of how to do it.

If your project uses recent versions Autoconf, Automake, and Libtool to build, you can use hwloc's embedded m4 capabilities. We have tested the embedded m4 with projects that use Autoconf 2.65, Automake 1.11.1, and Libtool 2.2.6b. Slightly earlier versions of may also work but are untested. Autoconf versions prior to 2.63 are almost certain to not work because hwloc uses macros that were introduced in 2.63.

You can either copy all the `config/hwloc*m4` files from the hwloc source tree to the directory where your project's m4 files reside, or you can tell aclocal to find more m4 files in the embedded hwloc's "config" subdirectory (e.g., add "`-Ipath/to/embedded/hwloc/config`" to your `Makefile.am`'s `ACLOCAL_AMFLAGS`).

The following macros can then be used from your configure script (only `HWLOC_INIT` *must* be invoked if using the m4 macros):

- `HWLOC_INIT(config-dir-prefix, action-upon-success, action-upon-failure)`: Invoke the hwloc configuration tests and setup the hwloc tree to build. The first



argument is the prefix to use for AC\_OUTPUT files -- it's where the hwloc tree is located relative to `$top_srcdir`. Hence, if your embedded hwloc is located in the source tree at `contrib/hwloc`, you should pass `[contrib/hwloc]` as the first argument. If `HWLOC_INIT` and the rest of `configure` completes successfully, then "make" traversals of the hwloc tree with standard Automake targets (all, clean, install, etc.) should behave as expected. For example, it is safe to list the hwloc directory in the SUBDIRS of a higher-level Makefile.am.

**NOTE: If the `HWLOC_SET_SYMBOL_PREFIX` macro is used, it must be invoked *before* `HWLOC_INIT`.**

- `HWLOC_SET_SYMBOL_PREFIX(foo_)`: Tells the hwloc to prefix all of hwloc's types and public symbols with "foo\_"; meaning that function `hwloc_init()` becomes `foo_hwloc_init()`. Enum values are prefixed with an upper-case translation if the prefix supplied; `HWLOC_OBJ_SYSTEM` becomes `FOO_HWLOC_OBJ_SYSTEM`. This is recommended behavior if you are including hwloc in middleware -- it is possible that your software will be combined with other software that links to another copy of hwloc. If both uses of hwloc utilize different symbol prefixes, there will be no type/symbol clashes, and everything will compile, link, and run successfully. If you both embed hwloc without changing the symbol prefix and also link against an external hwloc, you may get multiple symbol definitions when linking your final library or application.
- `HWLOC_DO_AM_CONDITIONALS`: If you embed hwloc in a larger project and build it conditionally (e.g., if `HWLOC_INIT` is invoked conditionally), you must unconditionally invoke `HWLOC_DO_AM_CONDITIONALS` to avoid warnings from Automake (for the cases where hwloc is not selected to be built). This macro is necessary because hwloc uses some `AM_CONDITIONALS` to build itself, and `AM_CONDITIONALS` cannot be defined conditionally. Note that it is safe (but unnecessary) to call `HWLOC_DO_AM_CONDITIONALS` even if `HWLOC_INIT` is invoked unconditionally.

NOTE: When using the `HWLOC_INIT` m4 macro, it may be necessary to explicitly invoke `AC_CANONICAL_TARGET` and/or `AC_USE_SYSTEM_EXTENSIONS` macros early in the configure script (e.g., after `AC_INIT` but before `AM_INIT_AUTOMAKE`). See the Autoconf documentation for further information.

## 7.2 Example embedding hwloc

Here's an example of integrating with a larger project named `sandbox` that already uses Autoconf, Automake, and Libtool to build itself:

```
# First, cd into the sandbox project source tree
shell$ cd sandbox
shell$ cp -r /somewhere/else/hwloc-<version> my-embedded-hwloc
```

```
shell$ edit Makefile.am
1. Add "-Imy-embedded-hwloc/config" to ACLOCAL_AMFLAGS
2. Add "my-embedded-hwloc" to SUBDIRS
3. Add "$(HWLOC_EMBEDDED_LDADD)" to sandbox's executable's LDADD line
4. Add "$(HWLOC_EMBEDDED_CPPFLAGS)" to AM_CPPFLAGS
shell$ edit configure.ac
1. Add "HWLOC_SET_SYMBOL_PREFIX(sandbox_hwloc_)" line
2. Add "HWLOC_INIT([my-embedded-hwloc], [happy=yes], [happy=no])" line
3. Add error checking for happy=no case
shell$ edit sandbox.c
1. Add #include <hwloc.h>
2. Add calls to sandbox_hwloc_init() and other hwloc API functions
```

Now you can bootstrap, configure, build, and run the sandbox as normal -- all calls to "sandbox\_hwloc\_\*" will use the embedded hwloc rather than any system-provided copy of hwloc.

## **Chapter 8**

# **Switching from PLPA to hwloc**

Although PLPA and hwloc share some of the same ideas, their programming interfaces are quite different. After much debate, it was decided *not* to emulate the PLPA API with hwloc's API because hwloc's API is already far more rich than PLPA's.

More specifically, exploiting modern computing architecture *requires* the flexible functionality provided by the hwloc API -- the PLPA API is too rigid in its definitions and practices to handle the evolving server hardware landscape (e.g., PLPA only understands cores and sockets; hwloc understands a much larger set of hardware objects).

As such, even though it is fully possible to emulate the PLPA API with hwloc (e.g., only deal with sockets and cores), and while the documentation below describes how to do this, we encourage any existing PLPA application authors to actually re-think their application in terms of more than just sockets and cores. In short, we encourage you to use the full hwloc API to exploit *all* the hardware.

## 8.1 Topology context vs. caching

First, all hwloc functions take a `topology` parameter. This parameter serves as an internal storage for the result of the topology discovery. It replaces PLPA's caching abilities and even lets you manipulate multiple topologies at the same time, if needed.

Thus, all programs should first run `hwloc_topology_init()` and `hwloc_topology_destroy()` as they did `plpa_init()` and `plpa_finalize()` in the past.

## 8.2 Hierarchy vs. Core@Socket

PLPA was designed to understand only cores and sockets. hwloc offers many more different types of objects (e.g., cores, sockets, hardware threads, NUMA nodes, and others) and stores them within a tree of resources.

To emulate the PLPA model, it is possible to find sockets using functions such as `hwloc_get_obj_by_type()`. Iterating over sockets is also possible using `hwloc_get_next_obj_by_type()`. Then, finding a core within a socket may be done using `hwloc_get_obj_inside_cpuset_by_type()` or `hwloc_get_next_obj_inside_cpuset_by_type()`.

It is also possible to directly find an object "below" another object using `hwloc_get_obj_below_by_type()` (or `hwloc_get_obj_below_array_by_type()`).

## 8.3 Logical vs. Physical/OS indexes

hwloc manipulates logical indexes, meaning indexes specified with regard to the ordering of objects in the hwloc-provided hierarchical tree. Physical or OS indexes may be entirely hidden if not strictly required. The reason for this is that physical/OS indexes

may change with the OS or with the BIOS version. They may be non-consecutive, multiple objects may have the same physical/OS indexes, making their manipulation tricky and highly non-portable.

Note that hwloc tries very hard to always present a hierarchical tree with the same logical ordering, regardless of physical or OS index ordering.

It is still possible to retrieve physical/OS indexes through the `os_index` field of objects, but such practice should be avoided as much as possible for the reasons described above (except perhaps for prettyprinting / debugging purposes).

[HWLOC\\_OBJ\\_PU](#) objects are supposed to have different physical/OS indexes since the OS uses them for binding. The `os_index` field of these objects provides the identifier that may be used for such binding, and `hwloc_get_proc_obj_by_os_index()` finds the object associated with a specific OS index.

But as mentioned above, we discourage the use of these conversion methods for actual binding. Instead, hwloc offers its own binding model using the `cpuset` field of objects. These cpusets may be duplicated, modified, combined, etc. (see [hwloc/cpuset.h](#) for details) and then passed to `hwloc_set_cpubind()` for binding.

## 8.4 Counting specification

PLPA offers a `countspec` parameter to specify whether counting all CPUs, only the online ones or only the offline ones. However, some operating systems do not expose the topology of offline CPUs (i.e., offline CPUs are not reported at all by the OS). Also, some processors may not be visible to the current application due to administrative restrictions. Finally, some processors let you shutdown a single hardware thread in a core, making some of the PLPA features irrelevant.

hwloc stores in the hierarchical tree of objects all CPUs that have known topology information. It then provides the applications with several cpusets that contain the list of CPUs that are actually known, that have topology information, that are online, or that are available to the application. These cpusets may be retrieved with [hwloc\\_topology\\_get\\_online\\_cpuset\(\)](#) and other similar functions to filter the object that are relevant or not.



## Chapter 9

# Module Index

### 9.1 Modules

Here is a list of all modules:

API version . . . . .	37
Topology context . . . . .	38
Topology Object Types . . . . .	39
Topology Objects . . . . .	42
Create and Destroy Topologies . . . . .	43
Configure Topology Detection . . . . .	45
Tinker with topologies. . . . .	50
Get some Topology Information . . . . .	52
Retrieve Objects . . . . .	55
Object/String Conversion . . . . .	56
Binding . . . . .	59
Object Type Helpers . . . . .	63
Basic Traversal Helpers . . . . .	64
Finding Objects Inside a CPU set . . . . .	67
Finding a single Object covering at least CPU set . . . . .	70
Finding a set of similar Objects covering at least a CPU set . . . . .	71
Cache-specific Finding Helpers . . . . .	72
Advanced Traversal Helpers . . . . .	73
Binding Helpers . . . . .	75
Cpuset Helpers . . . . .	76
The Cpuset API . . . . .	78
Helpers for manipulating glibc sched affinity . . . . .	89
Linux-only helpers . . . . .	90
Helpers for manipulating Linux libnuma unsigned long masks . . . . .	91
Helpers for manipulating Linux libnuma bitmask . . . . .	92

Helpers for manipulating Linux libnuma nodemask_t . . . . .	93
OpenFabrics-Specific Functions . . . . .	94



# Chapter 10

## Data Structure Index

### 10.1 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 ) . .	95
<a href="#">hwloc_obj_attr_u::hwloc_group_attr_s</a> (Group-specific Object Attributes ) . .	97
<a href="#">hwloc_obj_attr_u::hwloc_machine_attr_s</a> (Machine-specific Object At- tributes ) . . . . .	98
<a href="#">hwloc_obj</a> (Structure of a topology object ) . . . . .	99
<a href="#">hwloc_obj_attr_u</a> (Object type-specific Attributes ) . . . . .	106
<a href="#">hwloc_obj_memory_s::hwloc_obj_memory_page_type_s</a> (Array of local memory page types, NULL if no local memory and <code>page_types</code> is 0 ) . . . . .	108
<a href="#">hwloc_obj_memory_s</a> (Object memory ) . . . . .	109
<a href="#">hwloc_topology_cpupbind_support</a> (Flags describing actual binding support for this topology ) . . . . .	111
<a href="#">hwloc_topology_discovery_support</a> (Flags describing actual discovery sup- port for this topology ) . . . . .	113
<a href="#">hwloc_topology_support</a> (Set of flags describing actual support for this topology ) . . . . .	114



# Chapter 11

## Module Documentation

### 11.1 API version

#### Defines

- `#define HWLOC_API_VERSION 0x00010000`  
*Indicate at build time which hwloc API version is being used.*

#### 11.1.1 Define Documentation

##### 11.1.1.1 `#define HWLOC_API_VERSION 0x00010000`

Indicate at build time which hwloc API version is being used.

## 11.2 Topology context

### Typedefs

- typedef struct hwloc\_topology \* [hwloc\\_topology\\_t](#)  
*Topology context.*

### 11.2.1 Typedef Documentation

#### 11.2.1.1 typedef struct hwloc\_topology\* hwloc\_topology\_t

Topology context. To be initialized with [hwloc\\_topology\\_init\(\)](#) and built with [hwloc\\_topology\\_load\(\)](#).

## 11.3 Topology Object Types

### 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\\_PU](#),  
    [HWLOC\\_OBJ\\_GROUP](#),  
    [HWLOC\\_OBJ\\_MISC](#) }  
    *Type of topology object.*
- enum `hwloc_compare_types_e` { [HWLOC\\_TYPE\\_UNORDERED](#) }

### Functions

- `HWLOC_DECLSPEC int hwloc\_compare\_types (hwloc\_obj\_type\_t type1, hwloc\_obj\_type\_t type2) __hwloc_attribute_const`  
    *Compare the depth of two object types.*

### 11.3.1 Enumeration Type Documentation

#### 11.3.1.1 enum `hwloc_compare_types_e`

##### Enumerator:

***HWLOC\_TYPE\_UNORDERED*** Value returned by `hwloc_compare_types` when types can not be compared.

#### 11.3.1.2 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. The typical root object type. 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\_PU** Processing Unit, or (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\_GROUP** Group 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. And hwloc may insert such objects to group NUMA nodes according to their distances. These objects are ignored when they do not bring any structure.

**HWLOC\_OBJ\_MISC** Miscellaneous objects. Objects without particular meaning, that can e.g. be added by the application for its own use.

## 11.3.2 Function Documentation

### 11.3.2.1 HWLOC\_DECLSPEC int hwloc\_compare\_types(hwloc\_obj\_type\_t type1, hwloc\_obj\_type\_t type2) const

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 respectively if `type1` objects usually include `type2` objects, are the same as `type2` objects, or are included in `type2` objects. If the types can not be compared (because neither is usually contained in the other), `HWLOC_TYPE_UNORDERED` is returned. Object types containing CPUs can always be compared (usually, a system contains machines which contain nodes which contain sockets which contain caches, which contain cores, which contain processors).

**Note:**

HWLOC\_OBJ\_PU will always be the deepest.

This does not mean that the actual topology will respect that order: e.g. as of today cores may also contain caches, and sockets may also contain nodes. This is thus just to be seen as a fallback comparison method.

## 11.4 Topology Objects

### Data Structures

- struct [hwloc\\_obj\\_memory\\_s](#)  
*Object memory.*
- 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](#)  
*Convenience typedef; a pointer to a struct [hwloc\\_obj](#).*

#### 11.4.1 Typedef Documentation

##### 11.4.1.1 typedef struct hwloc\_obj\* hwloc\_obj\_t

Convenience typedef; a pointer to a struct [hwloc\\_obj](#).



## 11.5 Create and Destroy Topologies

### Functions

- HWLOC\_DECLSPEC int `hwloc_topology_init` (`hwloc_topology_t *topology`)

*Allocate a topology context.*

- HWLOC\_DECLSPEC int `hwloc_topology_load` (`hwloc_topology_t topology`)

*Build the actual topology.*

- HWLOC\_DECLSPEC void `hwloc_topology_destroy` (`hwloc_topology_t topology`)

*Terminate and free a topology context.*

- HWLOC\_DECLSPEC void `hwloc_topology_check` (`hwloc_topology_t topology`)

*Run internal checks on a topology structure.*

### 11.5.1 Function Documentation

#### 11.5.1.1 HWLOC\_DECLSPEC void `hwloc_topology_check` (`hwloc_topology_t topology`)

Run internal checks on a topology structure.

##### Parameters:

*topology* is the topology to be checked

#### 11.5.1.2 HWLOC\_DECLSPEC void `hwloc_topology_destroy` (`hwloc_topology_t topology`)

Terminate and free a topology context.

##### Parameters:

*topology* is the topology to be freed

#### 11.5.1.3 HWLOC\_DECLSPEC 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.

#### 11.5.1.4 HWLOC\_DECLSPEC 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](#)

## 11.6 Configure Topology Detection

### Data Structures

- struct `hwloc_topology_discovery_support`  
*Flags describing actual discovery support for this topology.*
- struct `hwloc_topology_cpubind_support`  
*Flags describing actual binding support for this topology.*
- struct `hwloc_topology_support`  
*Set of flags describing actual support for this topology.*

### Enumerations

- enum `hwloc_topology_flags_e` { `HWLOC_TOPOLOGY_FLAG_WHOLE_SYSTEM`, `HWLOC_TOPOLOGY_FLAG_IS_THISSYSTEM` }  
*Flags to be set onto a topology context before load.*

### Functions

- `HWLOC_DECLSPEC int hwloc_topology_ignore_type (hwloc_topology_t topology, hwloc_obj_type_t type)`  
*Ignore an object type.*
- `HWLOC_DECLSPEC 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.*
- `HWLOC_DECLSPEC int hwloc_topology_ignore_all_keep_structure (hwloc_topology_t topology)`  
*Ignore all objects that do not bring any structure.*
- `HWLOC_DECLSPEC int hwloc_topology_set_flags (hwloc_topology_t topology, unsigned long flags)`  
*Set OR'ed flags to non-yet-loaded topology.*
- `HWLOC_DECLSPEC int hwloc_topology_set_fsroot (hwloc_topology_t __hwloc_restrict topology, const char *__hwloc_restrict fsroot_path)`  
*Change the file-system root path when building the topology from sysfs/procfs.*

- `HWLOC_DECLSPEC int hwloc_topology_set_pid (hwloc_topology_t __hwloc_restrict topology, hwloc_pid_t pid)`  
*Change which pid the topology is viewed from.*
- `HWLOC_DECLSPEC int hwloc_topology_set_synthetic (hwloc_topology_t __hwloc_restrict topology, const char *__hwloc_restrict description)`  
*Enable synthetic topology.*
- `HWLOC_DECLSPEC int hwloc_topology_set_xml (hwloc_topology_t __hwloc_restrict topology, const char *__hwloc_restrict xmlpath)`  
*Enable XML-file based topology.*
- `HWLOC_DECLSPEC struct hwloc_topology_support * hwloc_topology_get_support (hwloc_topology_t __hwloc_restrict topology)`  
*Retrieve the topology support.*

### 11.6.1 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 return value of `hwloc_topology_is_thissystem()`.

### 11.6.2 Enumeration Type Documentation

#### 11.6.2.1 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`** Detect the whole system, ignore reservations and offline settings. Gather all resources, even if some

were disabled by the administrator. For instance, ignore Linux Cpusets and gather all processors and memory nodes, and ignore the fact that some resources may be offline.

***HWLOC\_TOPOLOGY\_FLAG\_IS\_THISSYSTEM*** Assume that the selected backend provides the topology for the system on which we are running. This forces `hwloc_topology_is_thissystem` to return 1, i.e. makes `hwloc` assume that the selected backend provides the topology for the system on which we are running, even if it is not the OS-specific backend but the XML backend for instance. This means making the binding functions actually call the OS-specific system calls and really do binding, while the XML backend would otherwise provide empty hooks just returning success.

Setting the environment variable `HWLOC_THISSYSTEM` may also result in the same behavior.

This can be used for efficiency reasons to first detect the topology once, save it to an XML file, and quickly reload it later through the XML backend, but still having binding functions actually do bind.

### 11.6.3 Function Documentation

**11.6.3.1 `HWLOC_DECLSPEC struct hwloc_topology_support*`  
`hwloc_topology_get_support` (`hwloc_topology_t __hwloc_restrict`  
`topology`) [`read`]**

Retrieve the topology support.

**11.6.3.2 `HWLOC_DECLSPEC 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 parent.

**11.6.3.3 `HWLOC_DECLSPEC 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 bottom-level type `HWLOC_OBJ_PU` may not be ignored. The top-level object of the hierarchy will never be ignored, even if this function succeeds.

#### 11.6.3.4 **HWLOC\_DECLSPEC 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 parent. The bottom-level type HWLOC\_OBJ\_PU may not be ignored.

#### 11.6.3.5 **HWLOC\_DECLSPEC 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.

#### 11.6.3.6 **HWLOC\_DECLSPEC int hwloc\_topology\_set\_fsroot (hwloc\_topology\_t \_\_hwloc\_restrict topology, const char \*\_\_hwloc\_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\(\)](#) 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.

#### 11.6.3.7 **HWLOC\_DECLSPEC int hwloc\_topology\_set\_pid (hwloc\_topology\_t \_\_hwloc\_restrict topology, hwloc\_pid\_t pid)**

Change which pid the topology is viewed from. On some systems, processes may have different views of the machine, for instance the set of allowed CPUs. By default, hwloc exposes the view from the current process. Calling [hwloc\\_topology\\_set\\_pid\(\)](#) permits to make it expose the topology of the machine from the point of view of another process.

**Note:**

hwloc\_pid\_t is pid\_t on unix platforms, and HANDLE on native Windows platforms

The ENOSYS error is returned on platforms that does not support this feature.

**11.6.3.8 HWLOC\_DECLSPEC int hwloc\_topology\_set\_synthetic**  
(hwloc\_topology\_t \_\_hwloc\_restrict *topology*, const char  
\*\_\_hwloc\_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. If only some level types are enforced, hwloc will try to choose the other types according to usual topologies, but it may fail and you may have to specify more level types manually.

**Note:**

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

**11.6.3.9 HWLOC\_DECLSPEC int hwloc\_topology\_set\_xml** (hwloc\_topology\_t  
\_\_hwloc\_restrict *topology*, const char \*\_\_hwloc\_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.

## 11.7 Tinker with topologies.

### Functions

- **HWLOC\_DECLSPEC** void [hwloc\\_topology\\_export\\_xml](#) ([hwloc\\_topology\\_t](#) topology, const char \*xmlpath)

*Export the topology into an XML file.*

- **HWLOC\_DECLSPEC** [hwloc\\_obj\\_t](#) [hwloc\\_topology\\_insert\\_misc\\_object\\_by\\_cpuset](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_const\\_cpuset\\_t](#) cpuset, const char \*name)

*Add a MISC object to the topology.*

- **HWLOC\_DECLSPEC** [hwloc\\_obj\\_t](#) [hwloc\\_topology\\_insert\\_misc\\_object\\_by\\_parent](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_obj\\_t](#) parent, const char \*name)

*Add a MISC object to the topology.*

### 11.7.1 Function Documentation

#### 11.7.1.1 **HWLOC\_DECLSPEC** void [hwloc\\_topology\\_export\\_xml](#) ([hwloc\\_topology\\_t](#) topology, const char \*xmlpath)

Export the topology into an XML file. This file may be loaded later through [hwloc\\_topology\\_set\\_xml\(\)](#).

#### 11.7.1.2 **HWLOC\_DECLSPEC** [hwloc\\_obj\\_t](#) [hwloc\\_topology\\_insert\\_misc\\_object\\_by\\_cpuset](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_const\\_cpuset\\_t](#) cpuset, const char \* name)

Add a MISC object to the topology. A new MISC object will be created and inserted into the topology at the position given by cpuset.

cpuset and name will be copied.

#### Returns:

the newly-created object



**11.7.1.3 HWLOC\_DECLSPEC hwloc\_obj\_t hwloc\_topology\_insert\_misc\_object\_by\_parent (hwloc\_topology\_t *topology*, hwloc\_obj\_t *parent*, const char \* *name*)**

Add a MISC object to the topology. A new MISC object will be created and inserted into the topology at the position given by parent.

name will be copied.

**Returns:**

the newly-created object

## 11.8 Get some Topology Information

### Enumerations

- enum `hwloc_get_type_depth_e` { `HWLOC_TYPE_DEPTH_UNKNOWN`, `HWLOC_TYPE_DEPTH_MULTIPLE` }

### Functions

- HWLOC\_DECLSPEC unsigned `hwloc_topology_get_depth` (`hwloc_topology_t` topology, `__hwloc_restrict_t` restrict topology) `__hwloc_attribute_pure`  
*Get the depth of the hierarchical tree of objects.*
- HWLOC\_DECLSPEC int `hwloc_get_type_depth` (`hwloc_topology_t` topology, `hwloc_obj_type_t` type)  
*Returns the depth of objects of type `type`.*
- HWLOC\_DECLSPEC `hwloc_obj_type_t` `hwloc_get_depth_type` (`hwloc_topology_t` topology, unsigned depth) `__hwloc_attribute_pure`  
*Returns the type of objects at depth `depth`.*
- HWLOC\_DECLSPEC unsigned `hwloc_get_nobjs_by_depth` (`hwloc_topology_t` topology, unsigned depth) `__hwloc_attribute_pure`  
*Returns the width of level at depth `depth`.*
- static `__hwloc_inline` int `__hwloc_attribute_pure` `hwloc_get_nobjs_by_type` (`hwloc_topology_t` topology, `hwloc_obj_type_t` type)  
*Returns the width of level type `type`.*
- HWLOC\_DECLSPEC int `hwloc_topology_is_thissystem` (`hwloc_topology_t` topology, `__hwloc_restrict_t` restrict topology) `__hwloc_attribute_pure`  
*Does the topology context come from this system?*

### 11.8.1 Enumeration Type Documentation

#### 11.8.1.1 enum `hwloc_get_type_depth_e`

##### Enumerator:

**`HWLOC_TYPE_DEPTH_UNKNOWN`** No object of given type exists in the topology.

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

## 11.8.2 Function Documentation

### 11.8.2.1 HWLOC\_DECLSPEC hwloc\_obj\_type\_t hwloc\_get\_depth\_type (hwloc\_topology\_t *topology*, unsigned *depth*)

Returns the type of objects at depth *depth*.

**Returns:**

-1 if depth *depth* does not exist.

### 11.8.2.2 HWLOC\_DECLSPEC unsigned hwloc\_get\_nobjs\_by\_depth (hwloc\_topology\_t *topology*, unsigned *depth*)

Returns the width of level at depth *depth*.

### 11.8.2.3 static \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure hwloc\_get\_nobjs\_by\_type (hwloc\_topology\_t *topology*, hwloc\_obj\_type\_t *type*) [static]

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

### 11.8.2.4 HWLOC\_DECLSPEC 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\(\)](#).

### 11.8.2.5 HWLOC\_DECLSPEC unsigned hwloc\_topology\_get\_depth (hwloc\_topology\_t \_\_hwloc\_restrict *topology*)

Get the depth of the hierarchical tree of objects. This is the depth of HWLOC\_OBJ\_PU objects plus one.

### 11.8.2.6 HWLOC\_DECLSPEC int hwloc\_topology\_is\_thissystem (hwloc\_topology\_t \_\_hwloc\_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).

## 11.9 Retrieve Objects

### Functions

- `HWLOC_DECLSPEC hwloc_obj_t hwloc_get_obj_by_depth (hwloc_topology_t topology, unsigned depth, unsigned idx) __hwloc_attribute_pure`  
*Returns the topology object at index `index` from depth `depth`.*
- `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure 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`.*

### 11.9.1 Function Documentation

#### 11.9.1.1 `HWLOC_DECLSPEC 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`.

#### 11.9.1.2 `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure 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 the caller may fallback to `hwloc_get_obj_by_depth()`.

## 11.10 Object/String Conversion

### Functions

- HWLOC\_DECLSPEC const char \* [hwloc\\_obj\\_type\\_string](#) (hwloc\_obj\_type\_t type) \_\_hwloc\_attribute\_const

*Return a stringified topology object type.*

- HWLOC\_DECLSPEC [hwloc\\_obj\\_type\\_t hwloc\\_obj\\_type\\_of\\_string](#) (const char \*string) \_\_hwloc\_attribute\_pure

*Return an object type from the string.*

- HWLOC\_DECLSPEC int [hwloc\\_obj\\_type\\_snprintf](#) (char \*\_\_hwloc\_restrict string, size\_t size, [hwloc\\_obj\\_t](#) obj, int verbose)

*Stringify the type of a given topology object into a human-readable form.*

- HWLOC\_DECLSPEC int [hwloc\\_obj\\_attr\\_snprintf](#) (char \*\_\_hwloc\_restrict string, size\_t size, [hwloc\\_obj\\_t](#) obj, const char \*\_\_hwloc\_restrict separator, int verbose)

*Stringify the attributes of a given topology object into a human-readable form.*

- HWLOC\_DECLSPEC int [hwloc\\_obj\\_snprintf](#) (char \*\_\_hwloc\_restrict string, size\_t size, [hwloc\\_topology\\_t](#) topology, [hwloc\\_obj\\_t](#) obj, const char \*\_\_hwloc\_restrict indexprefix, int verbose)

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

- HWLOC\_DECLSPEC int [hwloc\\_obj\\_cpuset\\_snprintf](#) (char \*\_\_hwloc\_restrict str, size\_t size, size\_t nobj, const [hwloc\\_obj\\_t](#) \*\_\_hwloc\_restrict objs)

*Stringify the cpuset containing a set of objects.*

### 11.10.1 Function Documentation

#### 11.10.1.1 HWLOC\_DECLSPEC int [hwloc\\_obj\\_attr\\_snprintf](#) (char \*\_\_hwloc\_restrict string, size\_t size, [hwloc\\_obj\\_t](#) obj, const char \*\_\_hwloc\_restrict separator, int verbose)

Stringify the attributes of a given topology object into a human-readable form. Attribute values are separated by `separator`.

Only the major attributes are printed in non-verbose mode.

**Returns:**

how many characters were actually written (not including the ending `\0`), or -1 on error.

**11.10.1.2** `HWLOC_DECLSPEC int hwloc_obj_cpuset_snprintf (char  
*__hwloc_restrict str, size_t size, size_t nobj, const hwloc_obj_t  
*__hwloc_restrict objs)`

Stringify the cpuset containing a set of objects.

**Returns:**

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

**11.10.1.3** `HWLOC_DECLSPEC int hwloc_obj_snprintf (char  
*__hwloc_restrict string, size_t size, hwloc_topology_t topology,  
hwloc_obj_t obj, const char *__hwloc_restrict indexprefix, int  
verbose)`

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

**Note:**

This function is deprecated in favor of `hwloc_obj_type_snprintf()` and `hwloc_obj_attr_snprintf()` since it is not very flexible and only prints physical/OS indexes.

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`), or -1 on error.

**11.10.1.4** `HWLOC_DECLSPEC hwloc_obj_type_t hwloc_obj_type_of_string  
(const char * string)`

Return an object type from the string.

**Returns:**

-1 if unrecognized.

**11.10.1.5 HWLOC\_DECLSPEC int hwloc\_obj\_type\_snprintf (char  
\*\_\_hwloc\_restrict *string*, size\_t *size*, hwloc\_obj\_t *obj*, int *verbose*)**

Stringify the type of a given topology object into a human-readable form. It differs from [hwloc\\_obj\\_type\\_string\(\)](#) because it prints type attributes such as cache depth.

**Returns:**

how many characters were actually written (not including the ending `\0`), or -1 on error.

**11.10.1.6 HWLOC\_DECLSPEC const char\* hwloc\_obj\_type\_string  
(hwloc\_obj\_type\_t *type*) const**

Return a stringified topology object type.



## 11.11 Binding

### Enumerations

- enum `hwloc_cpubind_policy_t` { `HWLOC_CPUBIND_PROCESS`, `HWLOC_CPUBIND_THREAD`, `HWLOC_CPUBIND_STRICT` }

*Process/Thread binding policy.*

### Functions

- `HWLOC_DECLSPEC int hwloc_set_cpubind (hwloc_topology_t topology, hwloc_const_cpuset_t set, int policy)`  
*Bind current process or thread on cpus given in cpuset set.*
- `HWLOC_DECLSPEC int hwloc_get_cpubind (hwloc_topology_t topology, hwloc_cpuset_t set, int policy)`  
*Get current process or thread binding.*
- `HWLOC_DECLSPEC int hwloc_set_proc_cpubind (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_const_cpuset_t set, int policy)`  
*Bind a process `pid` on cpus given in cpuset set.*
- `HWLOC_DECLSPEC int hwloc_get_proc_cpubind (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_cpuset_t set, int policy)`  
*Get the current binding of process `pid`.*
- `HWLOC_DECLSPEC int hwloc_set_thread_cpubind (hwloc_topology_t topology, hwloc_thread_t tid, hwloc_const_cpuset_t set, int policy)`  
*Bind a thread `tid` on cpus given in cpuset set.*
- `HWLOC_DECLSPEC int hwloc_get_thread_cpubind (hwloc_topology_t topology, hwloc_thread_t tid, hwloc_cpuset_t set, int policy)`  
*Get the current binding of thread `tid`.*

#### 11.11.1 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.

## 11.11.2 Enumeration Type Documentation

### 11.11.2.1 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.

When retrieving the binding of a process, this flag checks whether all its threads actually have the same binding. If the flag is not given, the binding of each thread will be accumulated.

**Note:**

This flag is meaningless when retrieving the binding of a thread.

### 11.11.3 Function Documentation

#### 11.11.3.1 **HWLOC\_DECLSPEC int hwloc\_get\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_cpuset\_t set**, **int policy**)

Get current process or thread binding.

#### 11.11.3.2 **HWLOC\_DECLSPEC int hwloc\_get\_proc\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_pid\_t pid**, **hwloc\_cpuset\_t set**, **int policy**)

Get the current binding of process `pid`.

**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`.

#### 11.11.3.3 **HWLOC\_DECLSPEC int hwloc\_get\_thread\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_thread\_t tid**, **hwloc\_cpuset\_t set**, **int policy**)

Get the current binding of thread `tid`.

**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`.

#### 11.11.3.4 **HWLOC\_DECLSPEC int hwloc\_set\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_const\_cpuset\_t set**, **int policy**)

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

**Returns:**

ENOSYS if the action is not supported  
EXDEV if the binding cannot be enforced

#### 11.11.3.5 **HWLOC\_DECLSPEC int hwloc\_set\_proc\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_pid\_t pid**, **hwloc\_const\_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*.

#### 11.11.3.6 **HWLOC\_DECLSPEC int hwloc\_set\_thread\_cpubind** (**hwloc\_topology\_t topology**, **hwloc\_thread\_t tid**, **hwloc\_const\_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*.

## 11.12 Object Type Helpers

### Functions

- static \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure [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 \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure [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.*

### 11.12.1 Function Documentation

**11.12.1.1** static \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure  
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.

**11.12.1.2** static \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure  
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.

## 11.13 Basic Traversal Helpers

### Functions

- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_root_obj(hwloc_topology_t topology)`  
*Returns the top-object of the topology-tree.*
- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_ancestor_obj_by_depth(hwloc_topology_t topology __hwloc_attribute_unused, unsigned depth, hwloc_obj_t obj)`  
*Returns the ancestor object of `obj` at depth `depth`.*
- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_ancestor_obj_by_type(hwloc_topology_t topology __hwloc_attribute_unused, hwloc_obj_type_t type, hwloc_obj_t obj)`  
*Returns the ancestor object of `obj` with type `type`.*
- static `__hwloc_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 `__hwloc_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 `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_by_os_index(hwloc_topology_t topology, unsigned os_index)`  
*Returns the object of type `HWLOC_OBJ_PU` with `os_index`.*
- static `__hwloc_inline hwloc_obj_t hwloc_get_next_child(hwloc_topology_t topology __hwloc_attribute_unused, hwloc_obj_t parent, hwloc_obj_t prev)`  
*Return the next child.*
- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_common_ancestor_obj(hwloc_topology_t topology __hwloc_attribute_unused, hwloc_obj_t obj1, hwloc_obj_t obj2)`  
*Returns the common parent object to objects `lvl1` and `lvl2`.*
- static `__hwloc_inline int __hwloc_attribute_pure hwloc_obj_is_in_subtree(hwloc_topology_t topology __hwloc_attribute_unused, hwloc_obj_t obj, hwloc_obj_t subtree_root)`  
*Returns true if `_obj` is inside the subtree beginning with `subtree_root`.*

### 11.13.1 Function Documentation

**11.13.1.1** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_ancestor_obj_by_depth (hwloc_topology_t topology  
__hwloc_attribute_unused, unsigned depth, hwloc_obj_t obj)  
[static]`

Returns the ancestor object of `obj` at depth `depth`.

**11.13.1.2** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_ancestor_obj_by_type (hwloc_topology_t topology  
__hwloc_attribute_unused, hwloc_obj_type_t type, hwloc_obj_t obj)  
[static]`

Returns the ancestor object of `obj` with type `type`.

**11.13.1.3** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_common_ancestor_obj (hwloc_topology_t topology  
__hwloc_attribute_unused, hwloc_obj_t obj1, hwloc_obj_t obj2)  
[static]`

Returns the common parent object to objects `lvl1` and `lvl2`.

**11.13.1.4** `static __hwloc_inline hwloc_obj_t hwloc_get_next_child  
(hwloc_topology_t topology __hwloc_attribute_unused, hwloc_obj_t  
parent, hwloc_obj_t prev) [static]`

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

**11.13.1.5** `static __hwloc_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`.

**11.13.1.6** `static __hwloc_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\(\)](#).

**11.13.1.7** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_pu_obj_by_os_index (hwloc_topology_t topology,  
unsigned os_index) [static]`

Returns the object of type [HWLOC\\_OBJ\\_PU](#) with `os_index`.

**Note:**

The `os_index` field of object should most of the times only be used for pretty-printing purpose. Type [HWLOC\\_OBJ\\_PU](#) is the only case where `os_index` could actually be useful, when manually binding to processors. However, using CPU sets to hide this complexity should often be preferred.

**11.13.1.8** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_root_obj (hwloc_topology_t topology) [static]`

Returns the top-object of the topology-tree. Its type is typically [HWLOC\\_OBJ\\_MACHINE](#) but it could be different for complex topologies. This function replaces the old deprecated `hwloc_get_system_obj()`.

**11.13.1.9** `static __hwloc_inline int __hwloc_attribute_pure  
hwloc_obj_is_in_subtree (hwloc_topology_t topology  
__hwloc_attribute_unused, hwloc_obj_t obj, hwloc_obj_t  
subtree_root) [static]`

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



## 11.14 Finding Objects Inside a CPU set

### Functions

- static `__hwloc_inline hwloc_obj_t hwloc_get_first_largest_obj_inside_cpuset` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set)  
*Get the first largest object included in the given cpuset set.*
- `HWLOC_DECLSPEC int hwloc_get_largest_objs_inside_cpuset` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `hwloc_obj_t *` `__hwloc_restrict_objs`, `int` max)  
*Get the set of largest objects covering exactly a given cpuset set.*
- static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_depth` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `unsigned` depth, `hwloc_obj_t` prev)  
*Return the next object at depth depth included in CPU set set.*
- static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_type` (`hwloc_topology_t` topology, `hwloc_const_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 `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_inside_cpuset_by_depth` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `unsigned` depth, `unsigned` idx)  
*Return the idx-th object at depth depth included in CPU set set.*
- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_inside_cpuset_by_type` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `hwloc_obj_type_t` type, `unsigned` idx)  
*Return the idx-th object of type type included in CPU set set.*
- static `__hwloc_inline unsigned __hwloc_attribute_pure hwloc_get_nbobjs_inside_cpuset_by_depth` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `unsigned` depth)  
*Return the number of objects at depth depth included in CPU set set.*
- static `__hwloc_inline int __hwloc_attribute_pure hwloc_get_nbobjs_inside_cpuset_by_type` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` set, `hwloc_obj_type_t` type)  
*Return the number of objects of type type included in CPU set set.*

### 11.14.1 Function Documentation

**11.14.1.1** `static __hwloc_inline hwloc_obj_t hwloc_get_first_largest_obj_inside_cpuset (hwloc_topology_t topology, hwloc_const_cpuset_t set)`  
**[static]**

Get the first largest object included in the given cpuset *set*.

**Returns:**

the first object that is included in *set* and whose parent is not.

This is convenient for iterating over all largest objects within a CPU set by doing a loop getting the first largest object and clearing its CPU set from the remaining CPU set.

**11.14.1.2** `HWLOC_DECLSPEC int hwloc_get_largest_objs_inside_cpuset (hwloc_topology_t topology, hwloc_const_cpuset_t set, hwloc_obj_t *__hwloc_restrict objs, int max)`

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

**Returns:**

the number of objects returned in *objs*.

**11.14.1.3** `static __hwloc_inline unsigned __hwloc_attribute_pure hwloc_get_nobjs_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_const_cpuset_t set, unsigned depth)` **[static]**

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

**11.14.1.4** `static __hwloc_inline int __hwloc_attribute_pure hwloc_get_nobjs_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_const_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.

**11.14.1.5** `static __hwloc_inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_const_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*.

**11.14.1.6** `static __hwloc_inline hwloc_obj_t hwloc_get_next_obj_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_const_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\(\)](#).

**11.14.1.7** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_inside_cpuset_by_depth (hwloc_topology_t topology, hwloc_const_cpuset_t set, unsigned depth, unsigned idx) [static]`

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

**11.14.1.8** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_inside_cpuset_by_type (hwloc_topology_t topology, hwloc_const_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\(\)](#).

## 11.15 Finding a single Object covering at least CPU set

### Functions

- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_child_covering_cpuset (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_const_cpuset_t set, hwloc_obj_t parent)`

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

- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_covering_cpuset (hwloc_topology_t topology, hwloc_const_cpuset_t set)`

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

### 11.15.1 Function Documentation

**11.15.1.1** static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_child_covering_cpuset (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_const_cpuset_t set, hwloc_obj_t parent) [static]`

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

#### Returns:

NULL if no child matches or if `set` is empty.

**11.15.1.2** static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_covering_cpuset (hwloc_topology_t topology, hwloc_const_cpuset_t set) [static]`

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

#### Returns:

NULL if no object matches or if `set` is empty.

## 11.16 Finding a set of similar Objects covering at least a CPU set

### Functions

- static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_depth` (`hwloc_topology_t topology`, `hwloc_const_cpuset_t set`, unsigned `depth`, `hwloc_obj_t prev`)

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

- static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_type` (`hwloc_topology_t topology`, `hwloc_const_cpuset_t set`, `hwloc_obj_type_t type`, `hwloc_obj_t prev`)

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

### 11.16.1 Function Documentation

**11.16.1.1** static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_depth` (`hwloc_topology_t topology`, `hwloc_const_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`.

**11.16.1.2** static `__hwloc_inline hwloc_obj_t hwloc_get_next_obj_covering_cpuset_by_type` (`hwloc_topology_t topology`, `hwloc_const_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.

## 11.17 Cache-specific Finding Helpers

### Functions

- static \_\_hwloc\_inline hwloc\_obj\_t \_\_hwloc\_attribute\_pure hwloc\_get\_cache\_covering\_cpuset (hwloc\_topology\_t topology, hwloc\_const\_cpuset\_t set)

*Get the first cache covering a cpuset set.*

- static \_\_hwloc\_inline hwloc\_obj\_t \_\_hwloc\_attribute\_pure hwloc\_get\_shared\_cache\_covering\_obj (hwloc\_topology\_t topology \_\_hwloc\_attribute\_unused, hwloc\_obj\_t obj)

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

### 11.17.1 Function Documentation

**11.17.1.1** static \_\_hwloc\_inline hwloc\_obj\_t \_\_hwloc\_attribute\_pure  
hwloc\_get\_cache\_covering\_cpuset (hwloc\_topology\_t topology,  
hwloc\_const\_cpuset\_t set) [static]

Get the first cache covering a cpuset set.

#### Returns:

NULL if no cache matches

**11.17.1.2** static \_\_hwloc\_inline hwloc\_obj\_t \_\_hwloc\_attribute\_pure  
hwloc\_get\_shared\_cache\_covering\_obj (hwloc\_topology\_t topology  
\_\_hwloc\_attribute\_unused, hwloc\_obj\_t obj) [static]

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

#### Returns:

NULL if no cache matches

## 11.18 Advanced Traversal Helpers

### Functions

- `HWLOC_DECLSPEC unsigned hwloc_get_closest_objs` (`hwloc_topology_t topology`, `hwloc_obj_t src`, `hwloc_obj_t *__hwloc_restrict objs`, unsigned `max`)

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

- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_below_by_type` (`hwloc_topology_t topology`, `hwloc_obj_type_t type1`, unsigned `idx1`, `hwloc_obj_type_t type2`, unsigned `idx2`)

*Find an object below another object, both specified by types and indexes.*

- static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_below_array_by_type` (`hwloc_topology_t topology`, int `nr`, `hwloc_obj_type_t *typev`, unsigned `*idxv`)

*Find an object below a chain of objects specified by types and indexes.*

### 11.18.1 Function Documentation

#### 11.18.1.1 `HWLOC_DECLSPEC unsigned hwloc_get_closest_objs` (`hwloc_topology_t topology`, `hwloc_obj_t src`, `hwloc_obj_t *__hwloc_restrict objs`, unsigned `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`.

#### 11.18.1.2 static `__hwloc_inline hwloc_obj_t __hwloc_attribute_pure hwloc_get_obj_below_array_by_type` (`hwloc_topology_t topology`, int `nr`, `hwloc_obj_type_t *typev`, unsigned `*idxv`) `[static]`

Find an object below a chain of objects specified by types and indexes. This is a generalized version of `hwloc_get_obj_below_by_type()`.

Arrays `typev` and `idxv` must contain `nr` types and indexes.

Start from the top system object and walk the arrays `typev` and `idxv`. For each type and index couple in the arrays, look under the previously found object to find the

index-th object of the given type. Indexes are specified within the parent, not withing the entire system.

For instance, if `nr` is 3, `typev` contains `NODE`, `SOCKET` and `CORE`, and `idxv` contains 0, 1 and 2, return the third core object below the second socket below the first NUMA node.

**11.18.1.3** `static __hwloc_inline hwloc_obj_t __hwloc_attribute_pure  
hwloc_get_obj_below_by_type (hwloc_topology_t topology,  
hwloc_obj_type_t type1, unsigned idx1, hwloc_obj_type_t type2,  
unsigned idx2) [static]`

Find an object below another object, both specified by types and indexes. Start from the top system object and find object of type `type1` and index `idx1`. Then look below this object and find another object of type `type2` and index `idx2`. Indexes are specified within the parent, not withing the entire system.

For instance, if `type1` is `SOCKET`, `idx1` is 2, `type2` is `CORE` and `idx2` is 3, return the fourth core object below the third socket.



## 11.19 Binding Helpers

### Functions

- static \_\_hwloc\_inline void [hwloc\\_distribute](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_obj\\_t](#) root, [hwloc\\_cpuset\\_t](#) \*cpuset, unsigned n)

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

#### 11.19.1 Function Documentation

**11.19.1.1** static \_\_hwloc\_inline void [hwloc\\_distribute](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_obj\\_t](#) root, [hwloc\\_cpuset\\_t](#) \*cpuset, unsigned 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 typically want to also call [hwloc\\_cpuset\\_singlify\(\)](#) before binding a thread so that it does not move at all.

## 11.20 Cpuset Helpers

### Functions

- static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_complete_cpuset(hwloc_topology_t topology)`
- static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_topology_cpuset(hwloc_topology_t topology)`
- static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_online_cpuset(hwloc_topology_t topology)`

*Get online CPU set.*

- static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_allowed_cpuset(hwloc_topology_t topology)`

*Get allowed CPU set.*

### 11.20.1 Function Documentation

**11.20.1.1** static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_allowed_cpuset(hwloc_topology_t topology)`  
[static]

Get allowed CPU set.

#### Returns:

the CPU set of allowed logical processors of the system. If the topology is the result of a combination of several systems, NULL is returned.

#### Note:

The returned cpuset is not newly allocated and should thus not be changed or freed, `hwloc_cpuset_dup` must be used to obtain a local copy.

**11.20.1.2** static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_complete_cpuset(hwloc_topology_t topology)`  
[static]

**11.20.1.3** static `__hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure hwloc_topology_get_online_cpuset(hwloc_topology_t topology)`  
[static]

Get online CPU set.

**Returns:**

the CPU set of online logical processors of the system. If the topology is the result of a combination of several systems, NULL is returned.

**Note:**

The returned cpuset is not newly allocated and should thus not be changed or freed; `hwloc_cpuset_dup` must be used to obtain a local copy.

**11.20.1.4** `static __hwloc_inline hwloc_const_cpuset_t __hwloc_attribute_pure  
hwloc_topology_get_topology_cpuset (hwloc_topology_t topology)  
[static]`

## 11.21 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. Needs a terminating `';`.*

### 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\_DECLSPEC `hwloc_cpuset_t hwloc_cpuset_alloc` (void) `__hwloc__attribute_malloc`  
*Allocate a new empty CPU set.*
- HWLOC\_DECLSPEC void `hwloc_cpuset_free` (`hwloc_cpuset_t` set)  
*Free CPU set `set`.*
- HWLOC\_DECLSPEC `hwloc_cpuset_t hwloc_cpuset_dup` (`hwloc_const_cpuset_t` set) `__hwloc__attribute_malloc`  
*Duplicate CPU set `set` by allocating a new CPU set and copying `set`'s contents.*
- HWLOC\_DECLSPEC void `hwloc_cpuset_copy` (`hwloc_cpuset_t` dst, `hwloc_const_cpuset_t` src)  
*Copy the contents of CPU set `src` into the already allocated CPU set `dst`.*
- HWLOC\_DECLSPEC int `hwloc_cpuset_sprintf` (char \*`__hwloc_restrict` buf, size\_t buflen, `hwloc_const_cpuset_t` set)  
*Stringify a cpuset.*
- HWLOC\_DECLSPEC int `hwloc_cpuset_asprintf` (char \*\*strp, `hwloc_const_cpuset_t` set)  
*Stringify a cpuset into a newly allocated string.*

- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_from\\_string](#) (hwloc\_cpuset\_t set, const char \*\_\_hwloc\_restrict string)  
*Parse a cpuset string and stores it in CPU set set.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_zero](#) (hwloc\_cpuset\_t set)  
*Empty the CPU set set.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_fill](#) (hwloc\_cpuset\_t set)  
*Fill CPU set set with all possible CPUs (even if those CPUs don't exist or are otherwise unavailable).*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_from\\_ulong](#) (hwloc\_cpuset\_t set, unsigned long mask)  
*Setup CPU set set from unsigned long mask.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_from\\_ith\\_ulong](#) (hwloc\_cpuset\_t set, unsigned i, unsigned long mask)  
*Setup CPU set set from unsigned long mask used as i -th subset.*
- HWLOC\_DECLSPEC unsigned long [hwloc\\_cpuset\\_to\\_ulong](#) (hwloc\_const\_cpuset\_t set) \_\_hwloc\_attribute\_pure  
*Convert the beginning part of CPU set set into unsigned long mask.*
- HWLOC\_DECLSPEC unsigned long [hwloc\\_cpuset\\_to\\_ith\\_ulong](#) (hwloc\_const\_cpuset\_t set, unsigned i) \_\_hwloc\_attribute\_pure  
*Convert the i -th subset of CPU set set into unsigned long mask.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_cpu](#) (hwloc\_cpuset\_t set, unsigned cpu)  
*Empty the CPU set set and add CPU cpu.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_all\\_but\\_cpu](#) (hwloc\_cpuset\_t set, unsigned cpu)  
*Empty the CPU set set and add all but the CPU cpu.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_set](#) (hwloc\_cpuset\_t set, unsigned cpu)  
*Add CPU cpu in CPU set set.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_set\\_range](#) (hwloc\_cpuset\_t set, unsigned begincpu, unsigned endcpu)  
*Add CPUs from begincpu to endcpu in CPU set set.*

- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_clr](#) ([hwloc\\_cpuset\\_t](#) set, unsigned cpu)  
*Remove CPU `cpu` from CPU set `set`.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_clr\\_range](#) ([hwloc\\_cpuset\\_t](#) set, unsigned begincpu, unsigned endcpu)  
*Remove CPUs from `begincpu` to `endcpu` in CPU set `set`.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_isset](#) ([hwloc\\_const\\_cpuset\\_t](#) set, unsigned cpu) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether CPU `cpu` is part of set `set`.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_iszero](#) ([hwloc\\_const\\_cpuset\\_t](#) set) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether set `set` is empty.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_isfull](#) ([hwloc\\_const\\_cpuset\\_t](#) set) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether set `set` is completely full.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_isequal](#) ([hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether set `set1` is equal to set `set2`.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_intersects](#) ([hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether sets `set1` and `set2` intersect.*
- HWLOC\_DECLSPEC int [hwloc\\_cpuset\\_isincluded](#) ([hwloc\\_const\\_cpuset\\_t](#) sub\_set, [hwloc\\_const\\_cpuset\\_t](#) super\_set) [\\_\\_hwloc\\_attribute\\_pure](#)  
*Test whether set `sub_set` is part of set `super_set`.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_or](#) ([hwloc\\_cpuset\\_t](#) res, [hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2)  
*Or sets `set1` and `set2` and store the result in set `res`.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_and](#) ([hwloc\\_cpuset\\_t](#) res, [hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2)  
*And sets `set1` and `set2` and store the result in set `res`.*
- HWLOC\_DECLSPEC void [hwloc\\_cpuset\\_andnot](#) ([hwloc\\_cpuset\\_t](#) res, [hwloc\\_const\\_cpuset\\_t](#) set1, [hwloc\\_const\\_cpuset\\_t](#) set2)

And set *set1* and the negation of *set2* and store the result in set *res*.

- `HWLOC_DECLSPEC void hwloc_cpuset_xor (hwloc_cpuset_t res, hwloc_const_cpuset_t set1, hwloc_const_cpuset_t set2)`  
*Xor sets *set1* and *set2* and store the result in set *res*.*
- `HWLOC_DECLSPEC void hwloc_cpuset_not (hwloc_cpuset_t res, hwloc_const_cpuset_t set)`  
*Negate set *set* and store the result in set *res*.*
- `HWLOC_DECLSPEC int hwloc_cpuset_first (hwloc_const_cpuset_t set) __hwloc_attribute_pure`  
*Compute the first CPU (least significant bit) in CPU set *set*.*
- `HWLOC_DECLSPEC int hwloc_cpuset_last (hwloc_const_cpuset_t set) __hwloc_attribute_pure`  
*Compute the last CPU (most significant bit) in CPU set *set*.*
- `HWLOC_DECLSPEC int hwloc_cpuset_next (hwloc_const_cpuset_t set, unsigned prev_cpu) __hwloc_attribute_pure`  
*Compute the next CPU in CPU set *set* which is after CPU *prev\_cpu*.*
- `HWLOC_DECLSPEC void hwloc_cpuset_singlify (hwloc_cpuset_t set)`  
*Keep a single CPU among those set in CPU set *set*.*
- `HWLOC_DECLSPEC int hwloc_cpuset_compare_first (hwloc_const_cpuset_t set1, hwloc_const_cpuset_t set2) __hwloc_attribute_pure`  
*Compare CPU sets *set1* and *set2* using their lowest index CPU.*
- `HWLOC_DECLSPEC int hwloc_cpuset_compare (hwloc_const_cpuset_t set1, hwloc_const_cpuset_t set2) __hwloc_attribute_pure`  
*Compare CPU sets *set1* and *set2* using their highest index CPU.*
- `HWLOC_DECLSPEC int hwloc_cpuset_weight (hwloc_const_cpuset_t set) __hwloc_attribute_pure`  
*Compute the "weight" of CPU set *set* (i.e., number of CPUs that are in the set).*

### 11.21.1 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.

## 11.21.2 Define Documentation

### 11.21.2.1 `#define hwloc_cpuset_foreach_begin(cpu, set)`

Loop macro iterating on CPU set `set`. `cpu` is the loop variable; it should be an unsigned int. The first iteration will set `cpu` to the lowest index CPU in the set. Successive iterations will iterate through, in order, all remaining CPUs that in the set. To be specific: each iteration will return a value for `cpu` such that `hwloc_cpuset_isset(set, cpu)` is true.

### 11.21.2.2 `#define hwloc_cpuset_foreach_end()`

End of loop. Needs a terminating `;`.

See also:

[hwloc\\_cpuset\\_foreach\\_begin](#)

## 11.21.3 Typedef Documentation

### 11.21.3.1 `typedef struct hwloc_cpuset_s* hwloc_const_cpuset_t`

### 11.21.3.2 `typedef struct hwloc_cpuset_s* hwloc_cpuset_t`

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

## 11.21.4 Function Documentation

### 11.21.4.1 `HWLOC_DECLSPEC void hwloc_cpuset_all_but_cpu(hwloc_cpuset_t set, unsigned cpu)`

Empty the CPU set `set` and add all but the CPU `cpu`.

### 11.21.4.2 `HWLOC_DECLSPEC hwloc_cpuset_t hwloc_cpuset_alloc(void)`

Allocate a new empty CPU set.

**Returns:**

A valid CPU set or NULL.

The CPU set should be freed by a corresponding call to [hwloc\\_cpuset\\_free\(\)](#).



**11.21.4.3 HWLOC\_DECLSPEC void hwloc\_cpuset\_and (hwloc\_cpuset\_t *res*, hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

And sets *set1* and *set2* and store the result in set *res*.

**11.21.4.4 HWLOC\_DECLSPEC void hwloc\_cpuset\_andnot (hwloc\_cpuset\_t *res*, hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

And set *set1* and the negation of *set2* and store the result in set *res*.

**11.21.4.5 HWLOC\_DECLSPEC 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`).

**11.21.4.6 HWLOC\_DECLSPEC void hwloc\_cpuset\_clr (hwloc\_cpuset\_t *set*, unsigned *cpu*)**

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

**11.21.4.7 HWLOC\_DECLSPEC void hwloc\_cpuset\_clr\_range (hwloc\_cpuset\_t *set*, unsigned *begincpu*, unsigned *endcpu*)**

Remove CPUs from *begincpu* to *endcpu* in CPU set *set*.

**11.21.4.8 HWLOC\_DECLSPEC int hwloc\_cpuset\_compare (hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

Compare CPU sets *set1* and *set2* using their highest index CPU. Higher most significant bit is higher. The empty CPU set is considered lower than anything.

**11.21.4.9 HWLOC\_DECLSPEC int hwloc\_cpuset\_compare\_first (hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

Compare CPU sets *set1* and *set2* using their lowest index CPU. Smaller least significant bit is smaller. The empty CPU set is considered higher than anything.

**11.21.4.10 HWLOC\_DECLSPEC void hwloc\_cpuset\_copy (hwloc\_cpuset\_t *dst*, hwloc\_const\_cpuset\_t *src*)**

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

**11.21.4.11 HWLOC\_DECLSPEC void hwloc\_cpuset\_cpu (hwloc\_cpuset\_t *set*, unsigned *cpu*)**

Empty the CPU set *set* and add CPU *cpu*.

**11.21.4.12 HWLOC\_DECLSPEC hwloc\_cpuset\_t hwloc\_cpuset\_dup (hwloc\_const\_cpuset\_t *set*)**

Duplicate CPU set *set* by allocating a new CPU set and copying *set*'s contents.

**11.21.4.13 HWLOC\_DECLSPEC void hwloc\_cpuset\_fill (hwloc\_cpuset\_t *set*)**

Fill CPU set *set* with all possible CPUs (even if those CPUs don't exist or are otherwise unavailable).

**11.21.4.14 HWLOC\_DECLSPEC int hwloc\_cpuset\_first (hwloc\_const\_cpuset\_t *set*)**

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

**Returns:**

-1 if no CPU is set.

**11.21.4.15 HWLOC\_DECLSPEC void hwloc\_cpuset\_free (hwloc\_cpuset\_t *set*)**

Free CPU set *set*.

**11.21.4.16 HWLOC\_DECLSPEC void hwloc\_cpuset\_from\_ith\_ulong (hwloc\_cpuset\_t *set*, unsigned *i*, unsigned long *mask*)**

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

**11.21.4.17** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_from\_string  
(hwloc\_cpuset\_t *set*, const char \*\_\_hwloc\_restrict *string*)

Parse a cpuset string and stores it in CPU set *set*. Must start and end with a digit.

**11.21.4.18** **HWLOC\_DECLSPEC** void hwloc\_cpuset\_from\_ulong  
(hwloc\_cpuset\_t *set*, unsigned long *mask*)

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

**11.21.4.19** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_intersects  
(hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)

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

**11.21.4.20** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_isequal  
(hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)

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

**11.21.4.21** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_isfull  
(hwloc\_const\_cpuset\_t *set*)

Test whether set *set* is completely full.

**11.21.4.22** **HWLOC\_DECLSPEC** 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*.

**11.21.4.23** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_isset (hwloc\_const\_cpuset\_t  
*set*, unsigned *cpu*)

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

**11.21.4.24** **HWLOC\_DECLSPEC** int hwloc\_cpuset\_iszero  
(hwloc\_const\_cpuset\_t *set*)

Test whether set *set* is empty.

**11.21.4.25 HWLOC\_DECLSPEC int hwloc\_cpuset\_last (hwloc\_const\_cpuset\_t *set*)**

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

**Returns:**

-1 if no CPU is set.

**11.21.4.26 HWLOC\_DECLSPEC int hwloc\_cpuset\_next (hwloc\_const\_cpuset\_t *set*, unsigned *prev\_cpu*)**

Compute the next CPU in CPU set *set* which is after CPU *prev\_cpu*.

**Returns:**

-1 if no CPU with higher index is set.

**11.21.4.27 HWLOC\_DECLSPEC void hwloc\_cpuset\_not (hwloc\_cpuset\_t *res*, hwloc\_const\_cpuset\_t *set*)**

Negate set *set* and store the result in set *res*.

**11.21.4.28 HWLOC\_DECLSPEC void hwloc\_cpuset\_or (hwloc\_cpuset\_t *res*, hwloc\_const\_cpuset\_t *set1*, hwloc\_const\_cpuset\_t *set2*)**

Or sets *set1* and *set2* and store the result in set *res*.

**11.21.4.29 HWLOC\_DECLSPEC void hwloc\_cpuset\_set (hwloc\_cpuset\_t *set*, unsigned *cpu*)**

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

**11.21.4.30 HWLOC\_DECLSPEC void hwloc\_cpuset\_set\_range (hwloc\_cpuset\_t *set*, unsigned *begincpu*, unsigned *endcpu*)**

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

#### 11.21.4.31 **HWLOC\_DECLSPEC void hwloc\_cpuset\_singlify (hwloc\_cpuset\_t set)**

Keep a single CPU among those set in CPU set *set*. May be useful before binding so that the process does not have a chance of migrating between multiple logical CPUs in the original mask.

#### 11.21.4.32 **HWLOC\_DECLSPEC int hwloc\_cpuset\_snprintf (char \*\_\_hwloc\_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`).

#### 11.21.4.33 **HWLOC\_DECLSPEC unsigned long hwloc\_cpuset\_to\_ith\_ulong (hwloc\_const\_cpuset\_t set, unsigned i)**

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

#### 11.21.4.34 **HWLOC\_DECLSPEC unsigned long hwloc\_cpuset\_to\_ulong (hwloc\_const\_cpuset\_t set)**

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

#### 11.21.4.35 **HWLOC\_DECLSPEC int hwloc\_cpuset\_weight (hwloc\_const\_cpuset\_t set)**

Compute the "weight" of CPU set *set* (i.e., number of CPUs that are in the set).

##### **Returns:**

the number of CPUs that are in the set.

#### 11.21.4.36 **HWLOC\_DECLSPEC void hwloc\_cpuset\_xor (hwloc\_cpuset\_t res, hwloc\_const\_cpuset\_t set1, hwloc\_const\_cpuset\_t set2)**

Xor sets *set1* and *set2* and store the result in set *res*.

**11.21.4.37 HWLOC\_DECLSPEC void hwloc\_cpuset\_zero (hwloc\_cpuset\_t *set*)**

Empty the CPU set *set*.

## 11.22 Helpers for manipulating glibc sched affinity

### Functions

- static `__hwloc_inline int hwloc_cpuset_to_glibc_sched_affinity (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_const_cpuset_t hwlocset, cpu_set_t *schedset, size_t schedsetsize)`

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

- static `__hwloc_inline int hwloc_cpuset_from_glibc_sched_affinity (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_cpuset_t hwlocset, const cpu_set_t *schedset, size_t schedsetsize)`

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

### 11.22.1 Function Documentation

**11.22.1.1** static `__hwloc_inline int hwloc_cpuset_from_glibc_sched_affinity (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_cpuset_t hwlocset, 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`

**11.22.1.2** static `__hwloc_inline int hwloc_cpuset_to_glibc_sched_affinity (hwloc_topology_t topology __hwloc_attribute_unused, hwloc_const_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`

## 11.23 Linux-only helpers

### Functions

- `HWLOC_DECLSPEC int hwloc_linux_parse_cpumap_file (FILE *file, hwloc_cpuset_t set)`  
*Convert a linux kernel cpumap file `file` into hwloc CPU set.*
- `HWLOC_DECLSPEC int hwloc_linux_set_tid_cpupbind (hwloc_topology_t topology, pid_t tid, hwloc_const_cpuset_t set)`  
*Bind a thread `tid` on cpus given in `cpuset set`.*
- `HWLOC_DECLSPEC int hwloc_linux_get_tid_cpupbind (hwloc_topology_t topology, pid_t tid, hwloc_cpuset_t set)`  
*Get the current binding of thread `tid`.*

### 11.23.1 Detailed Description

This includes helpers for manipulating linux kernel cpumap files, and hwloc equivalents of the Linux `sched_setaffinity` and `sched_getaffinity` system calls.

### 11.23.2 Function Documentation

#### 11.23.2.1 `HWLOC_DECLSPEC int hwloc_linux_get_tid_cpupbind (hwloc_topology_t topology, pid_t tid, hwloc_cpuset_t set)`

Get the current binding of thread `tid`. The behavior is exactly the same as the Linux `sched_setaffinity` system call, but uses a hwloc cpuset.

#### 11.23.2.2 `HWLOC_DECLSPEC int hwloc_linux_parse_cpumap_file (FILE *file, hwloc_cpuset_t set)`

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.

#### 11.23.2.3 `HWLOC_DECLSPEC int hwloc_linux_set_tid_cpupbind (hwloc_topology_t topology, pid_t tid, hwloc_const_cpuset_t set)`

Bind a thread `tid` on cpus given in `cpuset set`. The behavior is exactly the same as the Linux `sched_setaffinity` system call, but uses a hwloc cpuset.



## 11.24 Helpers for manipulating Linux libnuma unsigned long masks

### Functions

- static `__hwloc_inline int hwloc_cpuset_to_linux_libnuma_ulongs (hwloc_topology_t topology, hwloc_const_cpuset_t cpuset, unsigned long *mask, unsigned long *maxnode)`

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

- static `__hwloc_inline int hwloc_cpuset_from_linux_libnuma_ulongs (hwloc_topology_t topology, hwloc_cpuset_t cpuset, const unsigned long *mask, unsigned long maxnode)`

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

### 11.24.1 Function Documentation

**11.24.1.1** static `__hwloc_inline int hwloc_cpuset_from_linux_libnuma_ulongs (hwloc_topology_t topology, hwloc_cpuset_t cpuset, 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).

**11.24.1.2** static `__hwloc_inline int hwloc_cpuset_to_linux_libnuma_ulongs (hwloc_topology_t topology, hwloc_const_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.

## 11.25 Helpers for manipulating Linux libnuma bitmask

### Functions

- static \_\_hwloc\_inline struct bitmask\* \_\_hwloc\_attribute\_malloc hwloc\_cpuset\_to\_linux\_libnuma\_bitmask (hwloc\_topology\_t topology, hwloc\_const\_cpuset\_t cpuset)

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

- static \_\_hwloc\_inline int hwloc\_cpuset\_from\_linux\_libnuma\_bitmask (hwloc\_topology\_t topology, hwloc\_cpuset\_t cpuset, const struct bitmask \*bitmask)

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

### 11.25.1 Function Documentation

**11.25.1.1 static \_\_hwloc\_inline int hwloc\_cpuset\_from\_linux\_libnuma\_bitmask (hwloc\_topology\_t topology, hwloc\_cpuset\_t cpuset, 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.

**11.25.1.2 static \_\_hwloc\_inline struct bitmask\* \_\_hwloc\_attribute\_malloc hwloc\_cpuset\_to\_linux\_libnuma\_bitmask (hwloc\_topology\_t topology, hwloc\_const\_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.

#### Returns:

newly allocated struct bitmask.

## 11.26 Helpers for manipulating Linux libnuma nodemask\_t

### Functions

- static \_\_hwloc\_inline int `hwloc_cpuset_to_linux_libnuma_nodemask` (`hwloc_topology_t` topology, `hwloc_const_cpuset_t` cpuset, `nodemask_t *nodemask`)  
*Convert hwloc CPU set cpuset into libnuma nodemask nodemask.*
- static \_\_hwloc\_inline int `hwloc_cpuset_from_linux_libnuma_nodemask` (`hwloc_topology_t` topology, `hwloc_cpuset_t` cpuset, const `nodemask_t *nodemask`)  
*Convert libnuma nodemask nodemask into hwloc CPU set cpuset.*

### 11.26.1 Function Documentation

**11.26.1.1** static \_\_hwloc\_inline int `hwloc_cpuset_from_linux_libnuma_nodemask` (`hwloc_topology_t` topology, `hwloc_cpuset_t` cpuset, 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.

**11.26.1.2** static \_\_hwloc\_inline int `hwloc_cpuset_to_linux_libnuma_nodemask` (`hwloc_topology_t` topology, `hwloc_const_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.

## 11.27 OpenFabrics-Specific Functions

### Functions

- static \_\_hwloc\_inline int `hwloc_ibv_get_device_cpuset` (`hwloc_topology_t` topology \_\_hwloc\_attribute\_unused, struct `ibv_device` \*ibdev, `hwloc_cpuset_t` set)

*Get the CPU set of logical processors that are physically close to device `ibdev`.*

### 11.27.1 Function Documentation

**11.27.1.1** static \_\_hwloc\_inline int `hwloc_ibv_get_device_cpuset` (`hwloc_topology_t` topology \_\_hwloc\_attribute\_unused, struct `ibv_device` \*ibdev, `hwloc_cpuset_t` set) [**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 12

## Data Structure Documentation

### 12.1 hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s Struct Reference

Cache-specific Object Attributes.

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

#### Data Fields

- uint64\_t [size](#)  
*Size of cache in bytes.*
- unsigned [depth](#)  
*Depth of cache.*

#### 12.1.1 Detailed Description

Cache-specific Object Attributes.

#### 12.1.2 Field Documentation

##### 12.1.2.1 unsigned hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s::depth

Depth of cache.

**12.1.2.2 uint64\_t hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s::size**

Size of cache in bytes.

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

- hwloc.h

## 12.2 hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s Struct Reference

Group-specific Object Attributes.

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

### Data Fields

- unsigned [depth](#)  
*Depth of group object.*

### 12.2.1 Detailed Description

Group-specific Object Attributes.

### 12.2.2 Field Documentation

#### 12.2.2.1 unsigned hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s::depth

Depth of group object.

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

- hwloc.h

## 12.3 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.*

### 12.3.1 Detailed Description

Machine-specific Object Attributes.

### 12.3.2 Field Documentation

#### 12.3.2.1 char\* hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s::dmi\_board\_name

DMI board model name.

#### 12.3.2.2 char\* hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s::dmi\_board\_vendor

DMI board vendor name.

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

- hwloc.h



## 12.4 hwloc\_obj Struct Reference

Structure of a topology object.

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

### Data Fields

- [hwloc\\_obj\\_type\\_t](#) `type`  
*Type of object.*
- unsigned [os\\_index](#)  
*OS-provided physical index number.*
- `char *` [name](#)  
*Object description if any.*
- struct [hwloc\\_obj\\_memory\\_s](#) `memory`  
*Memory attributes.*
- 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.*
- signed [os\\_level](#)  
*OS-provided physical level, -1 if unknown or meaningless.*
- struct [hwloc\\_obj](#) \* [next\\_cousin](#)  
*Next object of same type.*
- struct [hwloc\\_obj](#) \* [prev\\_cousin](#)  
*Previous object of same type.*
- struct [hwloc\\_obj](#) \* [parent](#)  
*Parent, NULL if root (system object).*
- unsigned [sibling\\_rank](#)

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

- struct `hwloc_obj * next_sibling`  
*Next object below the same parent.*
- struct `hwloc_obj * prev_sibling`  
*Previous object below the same parent.*
- 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.*
- `hwloc_cpuset_t complete_cpuset`  
*The complete CPU set of logical processors of this object,.*
- `hwloc_cpuset_t online_cpuset`  
*The CPU set of online logical processors.*
- `hwloc_cpuset_t allowed_cpuset`  
*The CPU set of allowed logical processors.*
- `hwloc_cpuset_t nodeset`  
*NUMA nodes covered by this object or containing this object.*
- `hwloc_cpuset_t complete_nodeset`  
*The complete NUMA node set of this object,.*
- `hwloc_cpuset_t allowed_nodeset`  
*The set of allowed NUMA memory nodes.*

### 12.4.1 Detailed Description

Structure of a topology object. Applications mustn't modify any field except userdata .

### 12.4.2 Field Documentation

#### 12.4.2.1 hwloc\_cpuset\_t hwloc\_obj::allowed\_cpuset

The CPU set of allowed logical processors. This includes the CPUs contained in this object which are allowed for binding, i.e. passing them to the hwloc binding functions should not return permission errors. This is usually restricted by administration rules. Some of them may however be offline so binding to them may still not be possible, see `online_cpuset`.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

#### 12.4.2.2 hwloc\_cpuset\_t hwloc\_obj::allowed\_nodeset

The set of allowed NUMA memory nodes. This includes the NUMA memory nodes contained in this object which are allowed for memory allocation, i.e. passing them to NUMA node-directed memory allocation should not return permission errors. This is usually restricted by administration rules.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

#### 12.4.2.3 unsigned hwloc\_obj::arity

Number of children.

#### 12.4.2.4 union hwloc\_obj\_attr\_u\* hwloc\_obj::attr

Object type-specific Attributes.

#### 12.4.2.5 struct hwloc\_obj\*\* hwloc\_obj::children

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

#### 12.4.2.6 `hwloc_cpuset_t hwloc_obj::complete_cpuset`

The complete CPU set of logical processors of this object,. This includes not only the same as the `cpuset` field, but also the CPUs for which topology information is unknown or incomplete, and the CPUs that are ignored when the `HWLOC_TOPOLOGY_FLAG_WHOLE_SYSTEM` flag is not set. Thus no corresponding PU object may be found in the topology, because the precise position is undefined. It is however known that it would be somewhere under this object.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

#### 12.4.2.7 `hwloc_cpuset_t hwloc_obj::complete_nodeset`

The complete NUMA node set of this object,. This includes not only the same as the `nodeset` field, but also the NUMA nodes for which topology information is unknown or incomplete, and the nodes that are ignored when the `HWLOC_TOPOLOGY_FLAG_WHOLE_SYSTEM` flag is not set. Thus no corresponding NODE object may be found in the topology, because the precise position is undefined. It is however known that it would be somewhere under this object.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

#### 12.4.2.8 `hwloc_cpuset_t hwloc_obj::cpuset`

CPUs covered by this object. This is the set of CPUs for which there are PU objects in the topology under this object, i.e. which are known to be physically contained in this object and known how (the children path between this object and the PU objects).

If the `HWLOC_TOPOLOGY_FLAG_WHOLE_SYSTEM` configuration flag is set, some of these CPUs may be offline, or not allowed for binding, see `online_cpuset` and `allowed_cpuset`.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

#### 12.4.2.9 `unsigned hwloc_obj::depth`

Vertical index in the hierarchy.

**12.4.2.10 struct hwloc\_obj\* hwloc\_obj::first\_child**

First child.

**12.4.2.11 struct hwloc\_obj\* hwloc\_obj::last\_child**

Last child.

**12.4.2.12 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.

**12.4.2.13 struct hwloc\_obj\_memory\_s hwloc\_obj::memory**

Memory attributes.

**12.4.2.14 char\* hwloc\_obj::name**

Object description if any.

**12.4.2.15 struct hwloc\_obj\* hwloc\_obj::next\_cousin**

Next object of same type.

**12.4.2.16 struct hwloc\_obj\* hwloc\_obj::next\_sibling**

Next object below the same parent.

**12.4.2.17 hwloc\_cpuset\_t hwloc\_obj::nodeset**

NUMA nodes covered by this object or containing this object. This is the set of NUMA nodes for which there are NODE objects in the topology under or above this object, i.e. which are known to be physically contained in this object or containing it and known how (the children path between this object and the NODE objects).

If the HWLOC\_TOPOLOGY\_FLAG\_WHOLE\_SYSTEM configuration flag is set, some of these nodes may not be allowed for allocation, see allowed\_nodeset.

**Note:**

Its value must not be changed, hwloc\_cpuset\_dup must be used instead.

**12.4.2.18 hwloc\_cpuset\_t hwloc\_obj::online\_cpuset**

The CPU set of online logical processors. This includes the CPUs contained in this object that are online, i.e. draw power and can execute threads. It may however not be allowed to bind to them due to administration rules, see `allowed_cpuset`.

**Note:**

Its value must not be changed, `hwloc_cpuset_dup` must be used instead.

**12.4.2.19 unsigned hwloc\_obj::os\_index**

OS-provided physical index number.

**12.4.2.20 signed hwloc\_obj::os\_level**

OS-provided physical level, -1 if unknown or meaningless.

**12.4.2.21 struct hwloc\_obj\* hwloc\_obj::parent**

Parent, NULL if root (system object).

**12.4.2.22 struct hwloc\_obj\* hwloc\_obj::prev\_cousin**

Previous object of same type.

**12.4.2.23 struct hwloc\_obj\* hwloc\_obj::prev\_sibling**

Previous object below the same parent.

**12.4.2.24 unsigned hwloc\_obj::sibling\_rank**

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

**12.4.2.25 hwloc\_obj\_type\_t hwloc\_obj::type**

Type of object.

**12.4.2.26 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

## 12.5 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\\_group\\_attr\\_s](#)  
*Group-specific Object Attributes.*
- struct [hwloc\\_machine\\_attr\\_s](#)  
*Machine-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\\_machine\\_attr\\_s](#) machine  
*Machine-specific Object Attributes.*
- struct [hwloc\\_obj\\_attr\\_u::hwloc\\_group\\_attr\\_s](#) group  
*Group-specific Object Attributes.*

#### 12.5.1 Detailed Description

Object type-specific Attributes.

#### 12.5.2 Field Documentation

##### 12.5.2.1 struct [hwloc\\_obj\\_attr\\_u::hwloc\\_cache\\_attr\\_s](#) [hwloc\\_obj\\_attr\\_u::cache](#)

Cache-specific Object Attributes.



**12.5.2.2 struct hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s  
hwloc\_obj\_attr\_u::group**

Group-specific Object Attributes.

**12.5.2.3 struct hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s  
hwloc\_obj\_attr\_u::machine**

Machine-specific Object Attributes.

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

- hwloc.h

## 12.6 hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s Struct Reference

Array of local memory page types, NULL if no local memory and `page_types` is 0.

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

### Data Fields

- `uint64_t size`  
*Size of pages.*
- `uint64_t count`  
*Number of pages of this size.*

### 12.6.1 Detailed Description

Array of local memory page types, NULL if no local memory and `page_types` is 0. The array is sorted by increasing `size` fields. It contains `page_types_len` slots.

### 12.6.2 Field Documentation

#### 12.6.2.1 `uint64_t hwloc_obj_memory_s::hwloc_obj_memory_page_type_s::count`

Number of pages of this size.

#### 12.6.2.2 `uint64_t hwloc_obj_memory_s::hwloc_obj_memory_page_type_s::size`

Size of pages.

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

- `hwloc.h`

## 12.7 hwloc\_obj\_memory\_s Struct Reference

Object memory.

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

### Data Structures

- struct [hwloc\\_obj\\_memory\\_page\\_type\\_s](#)  
*Array of local memory page types, NULL if no local memory and page\_types is 0.*

### Data Fields

- uint64\_t [total\\_memory](#)  
*Total memory (in bytes) in this object and its children.*
- uint64\_t [local\\_memory](#)  
*Local memory (in bytes).*
- unsigned [page\\_types\\_len](#)  
*Size of array page\_types.*
- struct [hwloc\\_obj\\_memory\\_s::hwloc\\_obj\\_memory\\_page\\_type\\_s](#) \* [page\\_types](#)  
*Array of local memory page types, NULL if no local memory and page\_types is 0.*

### 12.7.1 Detailed Description

Object memory.

### 12.7.2 Field Documentation

#### 12.7.2.1 uint64\_t hwloc\_obj\_memory\_s::local\_memory

Local memory (in bytes).

#### 12.7.2.2 struct hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s \* hwloc\_obj\_memory\_s::page\_types

Array of local memory page types, NULL if no local memory and page\_types is 0. The array is sorted by increasing size fields. It contains page\_types\_len slots.

**12.7.2.3 unsigned hwloc\_obj\_memory\_s::page\_types\_len**

Size of array `page_types`.

**12.7.2.4 uint64\_t hwloc\_obj\_memory\_s::total\_memory**

Total memory (in bytes) in this object and its children.

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

- `hwloc.h`

## 12.8 hwloc\_topology\_cpupbind\_support Struct Reference

Flags describing actual binding support for this topology.

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

### Data Fields

- unsigned char [set\\_thisproc\\_cpupbind](#)
- unsigned char [get\\_thisproc\\_cpupbind](#)
- unsigned char [set\\_proc\\_cpupbind](#)
- unsigned char [get\\_proc\\_cpupbind](#)
- unsigned char [set\\_thisthread\\_cpupbind](#)
- unsigned char [get\\_thisthread\\_cpupbind](#)
- unsigned char [set\\_thread\\_cpupbind](#)
- unsigned char [get\\_thread\\_cpupbind](#)

### 12.8.1 Detailed Description

Flags describing actual binding support for this topology.

### 12.8.2 Field Documentation

#### 12.8.2.1 unsigned char hwloc\_topology\_cpupbind\_support::get\_proc\_cpupbind

Getting the binding of a whole given process is supported.

#### 12.8.2.2 unsigned char hwloc\_topology\_cpupbind\_support::get\_thisproc\_cpupbind

Getting the binding of the whole current process is supported.

#### 12.8.2.3 unsigned char hwloc\_topology\_cpupbind\_support::get\_thisthread\_cpupbind

Getting the binding of the current thread only is supported.

#### 12.8.2.4 unsigned char hwloc\_topology\_cpupbind\_support::get\_thread\_cpupbind

Getting the binding of a given thread only is supported.

**12.8.2.5 unsigned char hwloc\_topology\_cpupbind\_support::set\_proc\_cpupbind**

Binding a whole given process is supported.

**12.8.2.6 unsigned char hwloc\_topology\_cpupbind\_support::set\_thisproc\_cpupbind**

Binding the whole current process is supported.

**12.8.2.7 unsigned char hwloc\_topology\_cpupbind\_support::set\_thisthread\_cpupbind**

Binding the current thread only is supported.

**12.8.2.8 unsigned char hwloc\_topology\_cpupbind\_support::set\_thread\_cpupbind**

Binding a given thread only is supported.

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

- hwloc.h

## 12.9 hwloc\_topology\_discovery\_support Struct Reference

Flags describing actual discovery support for this topology.

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

### Data Fields

- unsigned char [pu](#)  
*Detecting the number of PU objects is supported.*

### 12.9.1 Detailed Description

Flags describing actual discovery support for this topology.

### 12.9.2 Field Documentation

#### 12.9.2.1 unsigned char hwloc\_topology\_discovery\_support::pu

Detecting the number of PU objects is supported.

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

- hwloc.h

## 12.10 hwloc\_topology\_support Struct Reference

Set of flags describing actual support for this topology.

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

### Data Fields

- struct [hwloc\\_topology\\_discovery\\_support](#) \* [discovery](#)
- struct [hwloc\\_topology\\_cpubind\\_support](#) \* [cpubind](#)

#### 12.10.1 Detailed Description

Set of flags describing actual support for this topology. This is retrieved with [hwloc\\_topology\\_get\\_support\(\)](#) and will be valid until the topology object is destroyed. Note: the values are correct only after discovery.

#### 12.10.2 Field Documentation

**12.10.2.1** struct [hwloc\\_topology\\_cpubind\\_support](#)\*  
[hwloc\\_topology\\_support::cpubind](#)

**12.10.2.2** struct [hwloc\\_topology\\_discovery\\_support](#)\*  
[hwloc\\_topology\\_support::discovery](#)

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

- [hwloc.h](#)



# Index

- Advanced Traversal Helpers, [73](#)
- allowed\_cpuset
  - hwloc\_obj, [101](#)
- allowed\_nodeset
  - hwloc\_obj, [101](#)
- API version, [37](#)
- arity
  - hwloc\_obj, [101](#)
- attr
  - hwloc\_obj, [101](#)
- Basic Traversal Helpers, [64](#)
- Binding, [59](#)
- Binding Helpers, [75](#)
- cache
  - hwloc\_obj\_attr\_u, [106](#)
- Cache-specific Finding Helpers, [72](#)
- children
  - hwloc\_obj, [101](#)
- complete\_cpuset
  - hwloc\_obj, [101](#)
- complete\_nodeset
  - hwloc\_obj, [102](#)
- Configure Topology Detection, [45](#)
- count
  - hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s, [108](#)
- cpubind
  - hwloc\_topology\_support, [114](#)
- cpuset
  - hwloc\_obj, [102](#)
- Cpuset Helpers, [76](#)
- Create and Destroy Topologies, [43](#)
- depth
  - hwloc\_obj, [102](#)
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [95](#)
  - hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s, [97](#)
- discovery
  - hwloc\_topology\_support, [114](#)
- dmi\_board\_name
  - hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [98](#)
- dmi\_board\_vendor
  - hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_s, [98](#)
- Finding a set of similar Objects covering at least a CPU set, [71](#)
- Finding a single Object covering at least CPU set, [70](#)
- Finding Objects Inside a CPU set, [67](#)
- first\_child
  - hwloc\_obj, [102](#)
- Get some Topology Information, [52](#)
- get\_proc\_cpubind
  - hwloc\_topology\_cpubind\_support, [111](#)
- get\_thisproc\_cpubind
  - hwloc\_topology\_cpubind\_support, [111](#)
- get\_thisthread\_cpubind
  - hwloc\_topology\_cpubind\_support, [111](#)
- get\_thread\_cpubind
  - hwloc\_topology\_cpubind\_support, [111](#)
- group
  - hwloc\_obj\_attr\_u, [106](#)

- Helpers for manipulating glibc sched affinity, [89](#)
- Helpers for manipulating Linux libnuma bitmask, [92](#)
- Helpers for manipulating Linux libnuma nodemask\_t, [93](#)
- Helpers for manipulating Linux libnuma unsigned long masks, [91](#)
- HWLOC\_CPUBIND\_PROCESS
  - hwlocality\_binding, [60](#)
- HWLOC\_CPUBIND\_STRICT
  - hwlocality\_binding, [60](#)
- HWLOC\_CPUBIND\_THREAD
  - hwlocality\_binding, [60](#)
- HWLOC\_OBJ\_CACHE
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_CORE
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_GROUP
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_MACHINE
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_MISC
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_NODE
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_PU
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_SOCKET
  - hwlocality\_types, [40](#)
- HWLOC\_OBJ\_SYSTEM
  - hwlocality\_types, [39](#)
- HWLOC\_TOPOLOGY\_FLAG\_IS\_THISSYSTEM
  - hwlocality\_configuration, [47](#)
- HWLOC\_TOPOLOGY\_FLAG\_WHOLE\_SYSTEM
  - hwlocality\_configuration, [46](#)
- HWLOC\_TYPE\_DEPTH\_MULTIPLE
  - hwlocality\_information, [52](#)
- HWLOC\_TYPE\_DEPTH\_UNKNOWN
  - hwlocality\_information, [52](#)
- HWLOC\_TYPE\_UNORDERED
  - hwlocality\_types, [39](#)
- HWLOC\_API\_VERSION
  - hwlocality\_api\_version, [37](#)
- hwloc\_compare\_types
  - hwlocality\_types, [40](#)
- hwloc\_compare\_types\_e
  - hwlocality\_types, [39](#)
- hwloc\_const\_cpuset\_t
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpубind\_policy\_t
  - hwlocality\_binding, [60](#)
- hwloc\_cpuset\_all\_but\_cpu
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpuset\_alloc
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpuset\_and
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpuset\_andnot
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_asprintf
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_clr
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_clr\_range
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_compare
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_compare\_first
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_copy
  - hwlocality\_cpuset, [83](#)
- hwloc\_cpuset\_cpu
  - hwlocality\_cpuset, [84](#)
- hwloc\_cpuset\_dup
  - hwlocality\_cpuset, [84](#)
- hwloc\_cpuset\_fill
  - hwlocality\_cpuset, [84](#)
- hwloc\_cpuset\_first
  - hwlocality\_cpuset, [84](#)
- hwloc\_cpuset\_foreach\_begin
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpuset\_foreach\_end
  - hwlocality\_cpuset, [82](#)
- hwloc\_cpuset\_free
  - hwlocality\_cpuset, [84](#)
- hwloc\_cpuset\_from\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, [89](#)
- hwloc\_cpuset\_from\_ith\_ulong
  - hwlocality\_cpuset, [84](#)

- hwloc\_cpuset\_from\_linux\_libnuma\_  
bitmask
  - hwlocality\_linux\_libnuma\_bitmask,  
92
- hwloc\_cpuset\_from\_linux\_libnuma\_  
nodemask
  - hwlocality\_linux\_libnuma\_  
nodemask, 93
- hwloc\_cpuset\_from\_linux\_libnuma\_  
ulongs
  - hwlocality\_linux\_libnuma\_ulongs,  
91
- hwloc\_cpuset\_from\_string
  - hwlocality\_cpuset, 84
- hwloc\_cpuset\_from\_ulong
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_intersects
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_isequal
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_isfull
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_isincluded
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_isset
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_iszero
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_last
  - hwlocality\_cpuset, 85
- hwloc\_cpuset\_next
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_not
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_or
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_set
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_set\_range
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_singlify
  - hwlocality\_cpuset, 86
- hwloc\_cpuset\_snprintf
  - hwlocality\_cpuset, 87
- hwloc\_cpuset\_t
  - hwlocality\_cpuset, 82
- hwloc\_cpuset\_to\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, 89
- hwloc\_cpuset\_to\_ith\_ulong
  - hwlocality\_cpuset, 87
- hwloc\_cpuset\_to\_linux\_libnuma\_  
bitmask
  - hwlocality\_linux\_libnuma\_bitmask,  
92
- hwloc\_cpuset\_to\_linux\_libnuma\_  
nodemask
  - hwlocality\_linux\_libnuma\_  
nodemask, 93
- hwloc\_cpuset\_to\_linux\_libnuma\_ulongs
  - hwlocality\_linux\_libnuma\_ulongs,  
91
- hwloc\_cpuset\_to\_ulong
  - hwlocality\_cpuset, 87
- hwloc\_cpuset\_weight
  - hwlocality\_cpuset, 87
- hwloc\_cpuset\_xor
  - hwlocality\_cpuset, 87
- hwloc\_cpuset\_zero
  - hwlocality\_cpuset, 87
- hwloc\_distribute
  - hwlocality\_helper\_binding, 75
- hwloc\_get\_ancestor\_obj\_by\_depth
  - hwlocality\_helper\_traversal\_basic,  
65
- hwloc\_get\_ancestor\_obj\_by\_type
  - hwlocality\_helper\_traversal\_basic,  
65
- hwloc\_get\_cache\_covering\_cpuset
  - hwlocality\_helper\_find\_cache, 72
- hwloc\_get\_child\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 70
- hwloc\_get\_closest\_objs
  - hwlocality\_helper\_traversal, 73
- hwloc\_get\_common\_ancestor\_obj
  - hwlocality\_helper\_traversal\_basic,  
65
- hwloc\_get\_cpubind
  - hwlocality\_binding, 61
- hwloc\_get\_depth\_type
  - hwlocality\_information, 53
- hwloc\_get\_first\_largest\_obj\_inside\_  
cpuset

- hwlocality\_helper\_find\_inside, 68
- hwloc\_get\_largest\_objs\_inside\_cpuset
  - hwlocality\_helper\_find\_inside, 68
- hwloc\_get\_nbobjs\_by\_depth
  - hwlocality\_information, 53
- hwloc\_get\_nbobjs\_by\_type
  - hwlocality\_information, 53
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_-depth
  - hwlocality\_helper\_find\_inside, 68
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_-type
  - hwlocality\_helper\_find\_inside, 68
- hwloc\_get\_next\_child
  - hwlocality\_helper\_traversal\_basic, 65
- hwloc\_get\_next\_obj\_by\_depth
  - hwlocality\_helper\_traversal\_basic, 65
- hwloc\_get\_next\_obj\_by\_type
  - hwlocality\_helper\_traversal\_basic, 65
- hwloc\_get\_next\_obj\_covering\_cpuset\_-by\_depth
  - hwlocality\_helper\_find\_coverings, 71
- hwloc\_get\_next\_obj\_covering\_cpuset\_-by\_type
  - hwlocality\_helper\_find\_coverings, 71
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_-depth
  - hwlocality\_helper\_find\_inside, 68
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_-type
  - hwlocality\_helper\_find\_inside, 69
- hwloc\_get\_obj\_below\_array\_by\_type
  - hwlocality\_helper\_traversal, 73
- hwloc\_get\_obj\_below\_by\_type
  - hwlocality\_helper\_traversal, 74
- hwloc\_get\_obj\_by\_depth
  - hwlocality\_traversal, 55
- hwloc\_get\_obj\_by\_type
  - hwlocality\_traversal, 55
- hwloc\_get\_obj\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 70
- hwloc\_get\_obj\_inside\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_inside, 69
- hwloc\_get\_obj\_inside\_cpuset\_by\_type
  - hwlocality\_helper\_find\_inside, 69
- hwloc\_get\_proc\_cpupbind
  - hwlocality\_binding, 61
- hwloc\_get\_pu\_obj\_by\_os\_index
  - hwlocality\_helper\_traversal\_basic, 66
- hwloc\_get\_root\_obj
  - hwlocality\_helper\_traversal\_basic, 66
- hwloc\_get\_shared\_cache\_covering\_obj
  - hwlocality\_helper\_find\_cache, 72
- hwloc\_get\_thread\_cpupbind
  - hwlocality\_binding, 61
- hwloc\_get\_type\_depth
  - hwlocality\_information, 53
- hwloc\_get\_type\_depth\_e
  - hwlocality\_information, 52
- hwloc\_get\_type\_or\_above\_depth
  - hwlocality\_helper\_types, 63
- hwloc\_get\_type\_or\_below\_depth
  - hwlocality\_helper\_types, 63
- hwloc\_ibv\_get\_device\_cpuset
  - hwloc\_openfabrics, 94
- hwloc\_linux\_get\_tid\_cpupbind
  - hwlocality\_linux, 90
- hwloc\_linux\_parse\_cpumap\_file
  - hwlocality\_linux, 90
- hwloc\_linux\_set\_tid\_cpupbind
  - hwlocality\_linux, 90
- hwloc\_obj, 99
  - allowed\_cpuset, 101
  - allowed\_nodeset, 101
  - arity, 101
  - attr, 101
  - children, 101
  - complete\_cpuset, 101
  - complete\_nodeset, 102
  - cpuset, 102
  - depth, 102
  - first\_child, 102
  - last\_child, 103
  - logical\_index, 103
  - memory, 103

- name, 103
- next\_cousin, 103
- next\_sibling, 103
- nodeset, 103
- online\_cpuset, 103
- os\_index, 104
- os\_level, 104
- parent, 104
- prev\_cousin, 104
- prev\_sibling, 104
- sibling\_rank, 104
- type, 104
- userdata, 104
- hwloc\_obj\_attr\_snprintf
  - hwlocality\_conversion, 56
- hwloc\_obj\_attr\_u, 106
  - cache, 106
  - group, 106
  - machine, 107
- hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s,  
95
  - depth, 95
  - size, 95
- hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s,  
97
  - depth, 97
- hwloc\_obj\_attr\_u::hwloc\_machine\_attr\_  
s, 98
  - dmi\_board\_name, 98
  - dmi\_board\_vendor, 98
- hwloc\_obj\_cpuset\_snprintf
  - hwlocality\_conversion, 57
- hwloc\_obj\_is\_in\_subtree
  - hwlocality\_helper\_traversal\_basic,  
66
- hwloc\_obj\_memory\_s, 109
  - local\_memory, 109
  - page\_types, 109
  - page\_types\_len, 109
  - total\_memory, 110
- hwloc\_obj\_memory\_s::hwloc\_obj\_  
memory\_page\_type\_s, 108
  - count, 108
  - size, 108
- hwloc\_obj\_snprintf
  - hwlocality\_conversion, 57
- hwloc\_obj\_t
  - hwlocality\_objects, 42
- hwloc\_obj\_type\_of\_string
  - hwlocality\_conversion, 57
- hwloc\_obj\_type\_snprintf
  - hwlocality\_conversion, 58
- hwloc\_obj\_type\_string
  - hwlocality\_conversion, 58
- hwloc\_obj\_type\_t
  - hwlocality\_types, 39
- hwloc\_openfabrics
  - hwloc\_ibv\_get\_device\_cpuset, 94
- hwloc\_set\_cpubind
  - hwlocality\_binding, 61
- hwloc\_set\_proc\_cpubind
  - hwlocality\_binding, 62
- hwloc\_set\_thread\_cpubind
  - hwlocality\_binding, 62
- hwloc\_topology\_check
  - hwlocality\_creation, 43
- hwloc\_topology\_cpubind\_support, 111
  - get\_proc\_cpubind, 111
  - get\_thisproc\_cpubind, 111
  - get\_thisthread\_cpubind, 111
  - get\_thread\_cpubind, 111
  - set\_proc\_cpubind, 111
  - set\_thisproc\_cpubind, 112
  - set\_thisthread\_cpubind, 112
  - set\_thread\_cpubind, 112
- hwloc\_topology\_destroy
  - hwlocality\_creation, 43
- hwloc\_topology\_discovery\_support, 113
  - pu, 113
- hwloc\_topology\_export\_xml
  - hwlocality\_tinker, 50
- hwloc\_topology\_flags\_e
  - hwlocality\_configuration, 46
- hwloc\_topology\_get\_allowed\_cpuset
  - hwlocality\_helper\_cpuset, 76
- hwloc\_topology\_get\_complete\_cpuset
  - hwlocality\_helper\_cpuset, 76
- hwloc\_topology\_get\_depth
  - hwlocality\_information, 53
- hwloc\_topology\_get\_online\_cpuset
  - hwlocality\_helper\_cpuset, 76
- hwloc\_topology\_get\_support

- hwlocality\_configuration, [47](#)
- hwloc\_topology\_get\_topology\_cpuset
  - hwlocality\_helper\_cpuset, [77](#)
- hwloc\_topology\_ignore\_all\_keep\_-structure
  - hwlocality\_configuration, [47](#)
- hwloc\_topology\_ignore\_type
  - hwlocality\_configuration, [47](#)
- hwloc\_topology\_ignore\_type\_keep\_-structure
  - hwlocality\_configuration, [47](#)
- hwloc\_topology\_init
  - hwlocality\_creation, [43](#)
- hwloc\_topology\_insert\_misc\_object\_-by\_cpuset
  - hwlocality\_tinker, [50](#)
- hwloc\_topology\_insert\_misc\_object\_-by\_parent
  - hwlocality\_tinker, [50](#)
- hwloc\_topology\_is\_thissystem
  - hwlocality\_information, [53](#)
- hwloc\_topology\_load
  - hwlocality\_creation, [44](#)
- hwloc\_topology\_set\_flags
  - hwlocality\_configuration, [48](#)
- hwloc\_topology\_set\_fsroot
  - hwlocality\_configuration, [48](#)
- hwloc\_topology\_set\_pid
  - hwlocality\_configuration, [48](#)
- hwloc\_topology\_set\_synthetic
  - hwlocality\_configuration, [49](#)
- hwloc\_topology\_set\_xml
  - hwlocality\_configuration, [49](#)
- hwloc\_topology\_support, [114](#)
  - cpubind, [114](#)
  - discovery, [114](#)
- hwloc\_topology\_t
  - hwlocality\_topology, [38](#)
- hwlocality\_binding
  - HWLOC\_CPUBIND\_PROCESS, [60](#)
  - HWLOC\_CPUBIND\_STRICT, [60](#)
  - HWLOC\_CPUBIND\_THREAD, [60](#)
- hwlocality\_configuration
  - HWLOC\_TOPOLOGY\_FLAG\_-IS\_THISSYSTEM, [47](#)
- HWLOC\_TOPOLOGY\_FLAG\_-WHOLE\_SYSTEM, [46](#)
- hwlocality\_information
  - HWLOC\_TYPE\_DEPTH\_-MULTIPLE, [52](#)
  - HWLOC\_TYPE\_DEPTH\_-UNKNOWN, [52](#)
- hwlocality\_types
  - HWLOC\_OBJ\_CACHE, [40](#)
  - HWLOC\_OBJ\_CORE, [40](#)
  - HWLOC\_OBJ\_GROUP, [40](#)
  - HWLOC\_OBJ\_MACHINE, [40](#)
  - HWLOC\_OBJ\_MISC, [40](#)
  - HWLOC\_OBJ\_NODE, [40](#)
  - HWLOC\_OBJ\_PU, [40](#)
  - HWLOC\_OBJ\_SOCKET, [40](#)
  - HWLOC\_OBJ\_SYSTEM, [39](#)
  - HWLOC\_TYPE\_UNORDERED, [39](#)
- hwlocality\_api\_version
  - HWLOC\_API\_VERSION, [37](#)
- hwlocality\_binding
  - hwloc\_cpubind\_policy\_t, [60](#)
  - hwloc\_get\_cpubind, [61](#)
  - hwloc\_get\_proc\_cpubind, [61](#)
  - hwloc\_get\_thread\_cpubind, [61](#)
  - hwloc\_set\_cpubind, [61](#)
  - hwloc\_set\_proc\_cpubind, [62](#)
  - hwloc\_set\_thread\_cpubind, [62](#)
- hwlocality\_configuration
  - hwloc\_topology\_flags\_e, [46](#)
  - hwloc\_topology\_get\_support, [47](#)
  - hwloc\_topology\_ignore\_all\_keep\_-structure, [47](#)
  - hwloc\_topology\_ignore\_type, [47](#)
  - hwloc\_topology\_ignore\_type\_-keep\_structure, [47](#)
  - hwloc\_topology\_set\_flags, [48](#)
  - hwloc\_topology\_set\_fsroot, [48](#)
  - hwloc\_topology\_set\_pid, [48](#)
  - hwloc\_topology\_set\_synthetic, [49](#)
  - hwloc\_topology\_set\_xml, [49](#)
- hwlocality\_conversion
  - hwloc\_obj\_attr\_snprintf, [56](#)
  - hwloc\_obj\_cpuset\_snprintf, [57](#)
  - hwloc\_obj\_snprintf, [57](#)

- hwloc\_obj\_type\_of\_string, 57
- hwloc\_obj\_type\_snprintf, 58
- hwloc\_obj\_type\_string, 58
- hwlocality\_cpuset
  - hwloc\_const\_cpuset\_t, 82
  - hwloc\_cpuset\_all\_but\_cpu, 82
  - hwloc\_cpuset\_alloc, 82
  - hwloc\_cpuset\_and, 82
  - hwloc\_cpuset\_andnot, 83
  - hwloc\_cpuset\_asprintf, 83
  - hwloc\_cpuset\_clr, 83
  - hwloc\_cpuset\_clr\_range, 83
  - hwloc\_cpuset\_compare, 83
  - hwloc\_cpuset\_compare\_first, 83
  - hwloc\_cpuset\_copy, 83
  - hwloc\_cpuset\_cpu, 84
  - hwloc\_cpuset\_dup, 84
  - hwloc\_cpuset\_fill, 84
  - hwloc\_cpuset\_first, 84
  - hwloc\_cpuset\_foreach\_begin, 82
  - hwloc\_cpuset\_foreach\_end, 82
  - hwloc\_cpuset\_free, 84
  - hwloc\_cpuset\_from\_ith\_ulong, 84
  - hwloc\_cpuset\_from\_string, 84
  - hwloc\_cpuset\_from\_ulong, 85
  - hwloc\_cpuset\_intersects, 85
  - hwloc\_cpuset\_isequal, 85
  - hwloc\_cpuset\_isfull, 85
  - hwloc\_cpuset\_isincluded, 85
  - hwloc\_cpuset\_isset, 85
  - hwloc\_cpuset\_iszero, 85
  - hwloc\_cpuset\_last, 85
  - hwloc\_cpuset\_next, 86
  - hwloc\_cpuset\_not, 86
  - hwloc\_cpuset\_or, 86
  - hwloc\_cpuset\_set, 86
  - hwloc\_cpuset\_set\_range, 86
  - hwloc\_cpuset\_singlify, 86
  - hwloc\_cpuset\_snprintf, 87
  - hwloc\_cpuset\_t, 82
  - hwloc\_cpuset\_to\_ith\_ulong, 87
  - hwloc\_cpuset\_to\_ulong, 87
  - hwloc\_cpuset\_weight, 87
  - hwloc\_cpuset\_xor, 87
  - hwloc\_cpuset\_zero, 87
- hwlocality\_creation
  - hwloc\_topology\_check, 43
  - hwloc\_topology\_destroy, 43
  - hwloc\_topology\_init, 43
  - hwloc\_topology\_load, 44
- hwlocality\_glibc\_sched
  - hwloc\_cpuset\_from\_glibc\_sched\_  
affinity, 89
  - hwloc\_cpuset\_to\_glibc\_sched\_  
affinity, 89
- hwlocality\_helper\_binding
  - hwloc\_distribute, 75
- hwlocality\_helper\_cpuset
  - hwloc\_topology\_get\_allowed\_  
cpuset, 76
  - hwloc\_topology\_get\_complete\_  
cpuset, 76
  - hwloc\_topology\_get\_online\_cpuset,  
76
  - hwloc\_topology\_get\_topology\_  
cpuset, 77
- hwlocality\_helper\_find\_cache
  - hwloc\_get\_cache\_covering\_cpuset,  
72
  - hwloc\_get\_shared\_cache\_  
covering\_obj, 72
- hwlocality\_helper\_find\_covering
  - hwloc\_get\_child\_covering\_cpuset,  
70
  - hwloc\_get\_obj\_covering\_cpuset, 70
- hwlocality\_helper\_find\_coverings
  - hwloc\_get\_next\_obj\_covering\_  
cpuset\_by\_depth, 71
  - hwloc\_get\_next\_obj\_covering\_  
cpuset\_by\_type, 71
- hwlocality\_helper\_find\_inside
  - hwloc\_get\_first\_largest\_obj\_  
inside\_cpuset, 68
  - hwloc\_get\_largest\_objs\_inside\_  
cpuset, 68
  - hwloc\_get\_nbobjs\_inside\_cpuset\_  
by\_depth, 68
  - hwloc\_get\_nbobjs\_inside\_cpuset\_  
by\_type, 68
  - hwloc\_get\_next\_obj\_inside\_  
cpuset\_by\_depth, 68

- hwloc\_get\_next\_obj\_inside\_-cpuset\_by\_type, [69](#)
- hwloc\_get\_obj\_inside\_cpuset\_by\_depth, [69](#)
- hwloc\_get\_obj\_inside\_cpuset\_by\_type, [69](#)
- hwlocality\_helper\_traversal
  - hwloc\_get\_closest\_objs, [73](#)
  - hwloc\_get\_obj\_below\_array\_by\_type, [73](#)
  - hwloc\_get\_obj\_below\_by\_type, [74](#)
- hwlocality\_helper\_traversal\_basic
  - hwloc\_get\_ancestor\_obj\_by\_depth, [65](#)
  - hwloc\_get\_ancestor\_obj\_by\_type, [65](#)
  - hwloc\_get\_common\_ancestor\_obj, [65](#)
  - hwloc\_get\_next\_child, [65](#)
  - hwloc\_get\_next\_obj\_by\_depth, [65](#)
  - hwloc\_get\_next\_obj\_by\_type, [65](#)
  - hwloc\_get\_pu\_obj\_by\_os\_index, [66](#)
  - hwloc\_get\_root\_obj, [66](#)
  - hwloc\_obj\_is\_in\_subtree, [66](#)
- hwlocality\_helper\_types
  - hwloc\_get\_type\_or\_above\_depth, [63](#)
  - hwloc\_get\_type\_or\_below\_depth, [63](#)
- hwlocality\_information
  - hwloc\_get\_depth\_type, [53](#)
  - hwloc\_get\_nbobjs\_by\_depth, [53](#)
  - hwloc\_get\_nbobjs\_by\_type, [53](#)
  - hwloc\_get\_type\_depth, [53](#)
  - hwloc\_get\_type\_depth\_e, [52](#)
  - hwloc\_topology\_get\_depth, [53](#)
  - hwloc\_topology\_is\_thissystem, [53](#)
- hwlocality\_linux
  - hwloc\_linux\_get\_tid\_cpupbind, [90](#)
  - hwloc\_linux\_parse\_cpumap\_file, [90](#)
  - hwloc\_linux\_set\_tid\_cpupbind, [90](#)
- hwlocality\_linux\_libnuma\_bitmask
  - hwloc\_cpuset\_from\_linux\_-libnuma\_bitmask, [92](#)
  - hwloc\_cpuset\_to\_linux\_libnuma\_-bitmask, [92](#)
- hwlocality\_linux\_libnuma\_nodemask
  - hwloc\_cpuset\_from\_linux\_-libnuma\_nodemask, [93](#)
  - hwloc\_cpuset\_to\_linux\_libnuma\_-nodemask, [93](#)
- hwlocality\_linux\_libnuma\_ulongs
  - hwloc\_cpuset\_from\_linux\_-libnuma\_ulongs, [91](#)
  - hwloc\_cpuset\_to\_linux\_libnuma\_-ulongs, [91](#)
- hwlocality\_objects
  - hwloc\_obj\_t, [42](#)
- hwlocality\_tinker
  - hwloc\_topology\_export\_xml, [50](#)
  - hwloc\_topology\_insert\_misc\_-object\_by\_cpuset, [50](#)
  - hwloc\_topology\_insert\_misc\_-object\_by\_parent, [50](#)
- hwlocality\_topology
  - hwloc\_topology\_t, [38](#)
- hwlocality\_traversal
  - hwloc\_get\_obj\_by\_depth, [55](#)
  - hwloc\_get\_obj\_by\_type, [55](#)
- hwlocality\_types
  - hwloc\_compare\_types, [40](#)
  - hwloc\_compare\_types\_e, [39](#)
  - hwloc\_obj\_type\_t, [39](#)
- last\_child
  - hwloc\_obj, [103](#)
- Linux-only helpers, [90](#)
- local\_memory
  - hwloc\_obj\_memory\_s, [109](#)
- logical\_index
  - hwloc\_obj, [103](#)
- machine
  - hwloc\_obj\_attr\_u, [107](#)
- memory
  - hwloc\_obj, [103](#)
- name
  - hwloc\_obj, [103](#)
- next\_cousin
  - hwloc\_obj, [103](#)
- next\_sibling



- hwloc\_obj, [103](#)
- nodeset
  - hwloc\_obj, [103](#)
- Object Type Helpers, [63](#)
- Object/String Conversion, [56](#)
- online\_cpuset
  - hwloc\_obj, [103](#)
- OpenFabrics-Specific Functions, [94](#)
- os\_index
  - hwloc\_obj, [104](#)
- os\_level
  - hwloc\_obj, [104](#)
- page\_types
  - hwloc\_obj\_memory\_s, [109](#)
- page\_types\_len
  - hwloc\_obj\_memory\_s, [109](#)
- parent
  - hwloc\_obj, [104](#)
- prev\_cousin
  - hwloc\_obj, [104](#)
- prev\_sibling
  - hwloc\_obj, [104](#)
- pu
  - hwloc\_topology\_discovery\_support, [113](#)
- Retrieve Objects, [55](#)
- set\_proc\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [111](#)
- set\_thisproc\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [112](#)
- set\_thisthread\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [112](#)
- set\_thread\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [112](#)
- sibling\_rank
  - hwloc\_obj, [104](#)
- size
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [95](#)
  - hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s, [108](#)
- The Cpuset API, [78](#)
- Tinker with topologies., [50](#)
- Topology context, [38](#)
- Topology Object Types, [39](#)
- Topology Objects, [42](#)
- total\_memory
  - hwloc\_obj\_memory\_s, [110](#)
- type
  - hwloc\_obj, [104](#)
- userdata
  - hwloc\_obj, [104](#)