

## Hardware Locality (hwloc)

1.1

Generated by Doxygen 1.7.1

Thu Dec 16 2010 16:48:29



# Contents

<b>1</b>	<b>Hardware Locality</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	2
1.3	CLI Examples . . . . .	3
1.4	Programming Interface . . . . .	9
1.4.1	Portability . . . . .	10
1.4.2	API Example . . . . .	13
1.5	Questions and Bugs . . . . .	17
1.6	History / Credits . . . . .	17
1.7	Further Reading . . . . .	18
<b>2</b>	<b>Terms and Definitions</b>	<b>19</b>
<b>3</b>	<b>Command-line tools</b>	<b>23</b>
3.1	lstopo . . . . .	23
3.2	hwloc-bind . . . . .	23
3.3	hwloc-calc . . . . .	24
3.4	hwloc-distrib . . . . .	24
3.5	hwloc-ps . . . . .	24
<b>4</b>	<b>Environment variables</b>	<b>25</b>
<b>5</b>	<b>CPU Binding and Memory Binding</b>	<b>27</b>
<b>6</b>	<b>Interoperability with other software</b>	<b>29</b>

<b>7 Thread safety</b>	<b>31</b>
<b>8 Embedding hwloc in other software</b>	<b>33</b>
8.1 Using hwloc's M4 Embedding Capabilities . . . . .	34
8.2 Example Embedding hwloc . . . . .	36
<b>9 Switching from PLPA to hwloc</b>	<b>37</b>
9.1 Topology Context vs. Caching . . . . .	37
9.2 Hierarchy vs. Core@Socket . . . . .	37
9.3 Logical vs. Physical/OS Indexes . . . . .	38
9.4 Counting Specification . . . . .	38
<b>10 Frequently Asked Questions</b>	<b>41</b>
10.1 I do not want hwloc to rediscover my enormous machine topology everytime I rerun a process . . . . .	41
<b>11 Module Index</b>	<b>43</b>
11.1 Modules . . . . .	43
<b>12 Data Structure Index</b>	<b>45</b>
12.1 Data Structures . . . . .	45
<b>13 Module Documentation</b>	<b>47</b>
13.1 API version . . . . .	47
13.1.1 Define Documentation . . . . .	47
13.1.1.1 HWLOC_API_VERSION . . . . .	47
13.2 Topology context . . . . .	47
13.2.1 Typedef Documentation . . . . .	48
13.2.1.1 hwloc_topology_t . . . . .	48
13.3 Object sets . . . . .	48
13.3.1 Typedef Documentation . . . . .	48
13.3.1.1 hwloc_const_cpuset_t . . . . .	48
13.3.1.2 hwloc_const_node_t . . . . .	48
13.3.1.3 hwloc_cpuset_t . . . . .	48

13.3.1.4 hwloc_nodeset_t . . . . .	49
13.4 Topology Object Types . . . . .	49
13.4.1 Enumeration Type Documentation . . . . .	49
13.4.1.1 hwloc_compare_types_e . . . . .	49
13.4.1.2 hwloc_obj_type_t . . . . .	50
13.4.2 Function Documentation . . . . .	50
13.4.2.1 hwloc_compare_types . . . . .	50
13.5 Topology Objects . . . . .	51
13.5.1 Typedef Documentation . . . . .	52
13.5.1.1 hwloc_obj_t . . . . .	52
13.6 Create and Destroy Topologies . . . . .	52
13.6.1 Function Documentation . . . . .	52
13.6.1.1 hwloc_topology_check . . . . .	52
13.6.1.2 hwloc_topology_destroy . . . . .	52
13.6.1.3 hwloc_topology_init . . . . .	53
13.6.1.4 hwloc_topology_load . . . . .	53
13.7 Configure Topology Detection . . . . .	53
13.7.1 Detailed Description . . . . .	55
13.7.2 Enumeration Type Documentation . . . . .	55
13.7.2.1 hwloc_topology_flags_e . . . . .	55
13.7.3 Function Documentation . . . . .	56
13.7.3.1 hwloc_topology_get_support . . . . .	56
13.7.3.2 hwloc_topology_ignore_all_keep_structure . . . . .	56
13.7.3.3 hwloc_topology_ignore_type . . . . .	56
13.7.3.4 hwloc_topology_ignore_type_keep_structure . . . . .	57
13.7.3.5 hwloc_topology_set_flags . . . . .	57
13.7.3.6 hwloc_topology_set_fsroot . . . . .	57
13.7.3.7 hwloc_topology_set_pid . . . . .	57
13.7.3.8 hwloc_topology_set_synthetic . . . . .	58
13.7.3.9 hwloc_topology_set_xml . . . . .	58
13.7.3.10 hwloc_topology_set_xmlbuffer . . . . .	58

13.8 Tinker with topologies. . . . .	59
13.8.1 Function Documentation . . . . .	59
13.8.1.1 hwloc_topology_export_xml . . . . .	59
13.8.1.2 hwloc_topology_export_xmlbuffer . . . . .	59
13.8.1.3 hwloc_topology_insert_misc_object_by_cpuset . . . . .	60
13.8.1.4 hwloc_topology_insert_misc_object_by_parent . . . . .	60
13.9 Get some Topology Information . . . . .	60
13.9.1 Enumeration Type Documentation . . . . .	61
13.9.1.1 hwloc_get_type_depth_e . . . . .	61
13.9.2 Function Documentation . . . . .	61
13.9.2.1 hwloc_get_depth_type . . . . .	61
13.9.2.2 hwloc_get_nbojs_by_depth . . . . .	62
13.9.2.3 hwloc_get_nbojs_by_type . . . . .	62
13.9.2.4 hwloc_get_type_depth . . . . .	62
13.9.2.5 hwloc_topology_get_depth . . . . .	62
13.9.2.6 hwloc_topology_is_thissystem . . . . .	62
13.10 Retrieve Objects . . . . .	63
13.10.1 Function Documentation . . . . .	63
13.10.1.1 hwloc_get_obj_by_depth . . . . .	63
13.10.1.2 hwloc_get_obj_by_type . . . . .	63
13.11 Object/String Conversion . . . . .	63
13.11.1 Function Documentation . . . . .	64
13.11.1.1 hwloc_obj_attr_snprintf . . . . .	64
13.11.1.2 hwloc_obj_cpuset_snprintf . . . . .	64
13.11.1.3 hwloc_obj_get_info_by_name . . . . .	65
13.11.1.4 hwloc_obj_snprintf . . . . .	65
13.11.1.5 hwloc_obj_type_of_string . . . . .	65
13.11.1.6 hwloc_obj_type_snprintf . . . . .	66
13.11.1.7 hwloc_obj_type_string . . . . .	66
13.12 CPU binding . . . . .	66
13.12.1 Detailed Description . . . . .	67

13.12.2 Enumeration Type Documentation . . . . .	68
13.12.2.1 hwloc_cpubind_flags_t . . . . .	68
13.12.3 Function Documentation . . . . .	69
13.12.3.1 hwloc_get_cpubind . . . . .	69
13.12.3.2 hwloc_get_proc_cpubind . . . . .	69
13.12.3.3 hwloc_get_thread_cpubind . . . . .	69
13.12.3.4 hwloc_set_cpubind . . . . .	69
13.12.3.5 hwloc_set_proc_cpubind . . . . .	70
13.12.3.6 hwloc_set_thread_cpubind . . . . .	70
13.13 Memory binding . . . . .	70
13.13.1 Detailed Description . . . . .	72
13.13.2 Enumeration Type Documentation . . . . .	73
13.13.2.1 hwloc_membind_flags_t . . . . .	73
13.13.2.2 hwloc_membind_policy_t . . . . .	74
13.13.3 Function Documentation . . . . .	74
13.13.3.1 hwloc_alloc . . . . .	74
13.13.3.2 hwloc_alloc_membind . . . . .	74
13.13.3.3 hwloc_alloc_membind_nodeuset . . . . .	75
13.13.3.4 hwloc_free . . . . .	75
13.13.3.5 hwloc_get_area_membind . . . . .	75
13.13.3.6 hwloc_get_area_membind_nodeuset . . . . .	76
13.13.3.7 hwloc_get_membind . . . . .	76
13.13.3.8 hwloc_get_membind_nodeuset . . . . .	76
13.13.3.9 hwloc_get_proc_membind . . . . .	76
13.13.3.10 hwloc_get_proc_membind_nodeuset . . . . .	76
13.13.3.11 hwloc_set_area_membind . . . . .	76
13.13.3.12 hwloc_set_area_membind_nodeuset . . . . .	77
13.13.3.13 hwloc_set_membind . . . . .	77
13.13.3.14 hwloc_set_membind_nodeuset . . . . .	77
13.13.3.15 hwloc_set_proc_membind . . . . .	77
13.13.3.16 hwloc_set_proc_membind_nodeuset . . . . .	78

13.14 Object Type Helpers . . . . .	78
13.14.1 Function Documentation . . . . .	78
13.14.1.1 hwloc_get_type_or_above_depth . . . . .	78
13.14.1.2 hwloc_get_type_or_below_depth . . . . .	78
13.15 Basic Traversal Helpers . . . . .	79
13.15.1 Function Documentation . . . . .	80
13.15.1.1 hwloc_get_ancestor_obj_by_depth . . . . .	80
13.15.1.2 hwloc_get_ancestor_obj_by_type . . . . .	80
13.15.1.3 hwloc_get_common_ancestor_obj . . . . .	80
13.15.1.4 hwloc_get_next_child . . . . .	80
13.15.1.5 hwloc_get_next_obj_by_depth . . . . .	80
13.15.1.6 hwloc_get_next_obj_by_type . . . . .	81
13.15.1.7 hwloc_get_pu_obj_by_os_index . . . . .	81
13.15.1.8 hwloc_get_root_obj . . . . .	81
13.15.1.9 hwloc_obj_is_in_subtree . . . . .	81
13.16 Finding Objects Inside a CPU set . . . . .	82
13.16.1 Function Documentation . . . . .	83
13.16.1.1 hwloc_get_first_largest_obj_inside_cpuset . . . . .	83
13.16.1.2 hwloc_get_largest_objs_inside_cpuset . . . . .	83
13.16.1.3 hwloc_get_nbobjs_inside_cpuset_by_depth . . . . .	83
13.16.1.4 hwloc_get_nbobjs_inside_cpuset_by_type . . . . .	83
13.16.1.5 hwloc_get_next_obj_inside_cpuset_by_depth . . . . .	84
13.16.1.6 hwloc_get_next_obj_inside_cpuset_by_type . . . . .	84
13.16.1.7 hwloc_get_obj_inside_cpuset_by_depth . . . . .	84
13.16.1.8 hwloc_get_obj_inside_cpuset_by_type . . . . .	84
13.17 Finding a single Object covering at least CPU set . . . . .	84
13.17.1 Function Documentation . . . . .	85
13.17.1.1 hwloc_get_child_covering_cpuset . . . . .	85
13.17.1.2 hwloc_get_obj_covering_cpuset . . . . .	85
13.18 Finding a set of similar Objects covering at least a CPU set . . . . .	85
13.18.1 Function Documentation . . . . .	86



13.18.1.1 hwloc_get_next_obj_covering_cpuset_by_depth . . .	86
13.18.1.2 hwloc_get_next_obj_covering_cpuset_by_type . . .	86
13.19 Cache-specific Finding Helpers . . . . .	86
13.19.1 Function Documentation . . . . .	87
13.19.1.1 hwloc_get_cache_covering_cpuset . . . . .	87
13.19.1.2 hwloc_get_shared_cache_covering_obj . . . . .	87
13.20 Advanced Traversal Helpers . . . . .	87
13.20.1 Function Documentation . . . . .	88
13.20.1.1 hwloc_get_closest_objs . . . . .	88
13.20.1.2 hwloc_get_obj_below_array_by_type . . . . .	88
13.20.1.3 hwloc_get_obj_below_by_type . . . . .	88
13.21 Binding Helpers . . . . .	89
13.21.1 Function Documentation . . . . .	89
13.21.1.1 hwloc_alloc_membind_policy . . . . .	89
13.21.1.2 hwloc_alloc_membind_policy_nodeset . . . . .	89
13.21.1.3 hwloc_distribute . . . . .	90
13.21.1.4 hwloc_distributev . . . . .	90
13.22 Cpuset Helpers . . . . .	90
13.22.1 Function Documentation . . . . .	91
13.22.1.1 hwloc_topology_get_allowed_cpuset . . . . .	91
13.22.1.2 hwloc_topology_get_complete_cpuset . . . . .	91
13.22.1.3 hwloc_topology_get_online_cpuset . . . . .	91
13.22.1.4 hwloc_topology_get_topology_cpuset . . . . .	92
13.23 Nodeset Helpers . . . . .	92
13.23.1 Function Documentation . . . . .	92
13.23.1.1 hwloc_topology_get_allowed_nodeset . . . . .	92
13.23.1.2 hwloc_topology_get_complete_nodeset . . . . .	93
13.23.1.3 hwloc_topology_get_topology_nodeset . . . . .	93
13.24 Conversion between cpuset and nodeset . . . . .	93
13.24.1 Detailed Description . . . . .	93
13.24.2 Function Documentation . . . . .	94

13.24.2.1 hwloc_cpuset_from_nodeset . . . . .	94
13.24.2.2 hwloc_cpuset_from_nodeset_strict . . . . .	94
13.24.2.3 hwloc_cpuset_to_nodeset . . . . .	94
13.24.2.4 hwloc_cpuset_to_nodeset_strict . . . . .	94
13.25 The bitmap API . . . . .	95
13.25.1 Detailed Description . . . . .	99
13.25.2 Define Documentation . . . . .	99
13.25.2.1 hwloc_bitmap_foreach_begin . . . . .	99
13.25.2.2 hwloc_bitmap_foreach_end . . . . .	99
13.25.3 Typedef Documentation . . . . .	100
13.25.3.1 hwloc_bitmap_t . . . . .	100
13.25.3.2 hwloc_const_bitmap_t . . . . .	100
13.25.4 Function Documentation . . . . .	100
13.25.4.1 hwloc_bitmap_allbut . . . . .	100
13.25.4.2 hwloc_bitmap_alloc . . . . .	100
13.25.4.3 hwloc_bitmap_alloc_full . . . . .	100
13.25.4.4 hwloc_bitmap_and . . . . .	100
13.25.4.5 hwloc_bitmap_andnot . . . . .	100
13.25.4.6 hwloc_bitmap_asprintf . . . . .	101
13.25.4.7 hwloc_bitmap_clr . . . . .	101
13.25.4.8 hwloc_bitmap_clr_range . . . . .	101
13.25.4.9 hwloc_bitmap_compare . . . . .	101
13.25.4.10 hwloc_bitmap_compare_first . . . . .	101
13.25.4.11 hwloc_bitmap_copy . . . . .	101
13.25.4.12 hwloc_bitmap_dup . . . . .	102
13.25.4.13 hwloc_bitmap_fill . . . . .	102
13.25.4.14 hwloc_bitmap_first . . . . .	102
13.25.4.15 hwloc_bitmap_free . . . . .	102
13.25.4.16 hwloc_bitmap_from_ith_ulong . . . . .	102
13.25.4.17 hwloc_bitmap_from_ulong . . . . .	102
13.25.4.18 hwloc_bitmap_intersects . . . . .	102

13.25.4.19	<a href="#">hwloc_bitmap_isequal</a> . . . . .	103
13.25.4.20	<a href="#">hwloc_bitmap_isfull</a> . . . . .	103
13.25.4.21	<a href="#">hwloc_bitmap_isincluded</a> . . . . .	103
13.25.4.22	<a href="#">hwloc_bitmap_isset</a> . . . . .	103
13.25.4.23	<a href="#">hwloc_bitmap_iszero</a> . . . . .	103
13.25.4.24	<a href="#">hwloc_bitmap_last</a> . . . . .	103
13.25.4.25	<a href="#">hwloc_bitmap_next</a> . . . . .	103
13.25.4.26	<a href="#">hwloc_bitmap_not</a> . . . . .	104
13.25.4.27	<a href="#">hwloc_bitmap_only</a> . . . . .	104
13.25.4.28	<a href="#">hwloc_bitmap_or</a> . . . . .	104
13.25.4.29	<a href="#">hwloc_bitmap_set</a> . . . . .	104
13.25.4.30	<a href="#">hwloc_bitmap_set_ith_ulong</a> . . . . .	104
13.25.4.31	<a href="#">hwloc_bitmap_set_range</a> . . . . .	104
13.25.4.32	<a href="#">hwloc_bitmap_singlify</a> . . . . .	104
13.25.4.33	<a href="#">hwloc_bitmap_snprintf</a> . . . . .	105
13.25.4.34	<a href="#">hwloc_bitmap_sscanf</a> . . . . .	105
13.25.4.35	<a href="#">hwloc_bitmap_taskset_asprintf</a> . . . . .	105
13.25.4.36	<a href="#">hwloc_bitmap_taskset_snprintf</a> . . . . .	105
13.25.4.37	<a href="#">hwloc_bitmap_taskset_sscanf</a> . . . . .	105
13.25.4.38	<a href="#">hwloc_bitmap_to_ith_ulong</a> . . . . .	105
13.25.4.39	<a href="#">hwloc_bitmap_to_ulong</a> . . . . .	106
13.25.4.40	<a href="#">hwloc_bitmap_weight</a> . . . . .	106
13.25.4.41	<a href="#">hwloc_bitmap_xor</a> . . . . .	106
13.25.4.42	<a href="#">hwloc_bitmap_zero</a> . . . . .	106
13.26	Helpers for manipulating glibc sched affinity . . . . .	106
13.26.1	Function Documentation . . . . .	107
13.26.1.1	<a href="#">hwloc_cpuset_from_glibc_sched_affinity</a> . . . . .	107
13.26.1.2	<a href="#">hwloc_cpuset_to_glibc_sched_affinity</a> . . . . .	107
13.27	Linux-only helpers . . . . .	107
13.27.1	Detailed Description . . . . .	108
13.27.2	Function Documentation . . . . .	108

13.27.2.1 hwloc_linux_get_tid_cpubind . . . . .	108
13.27.2.2 hwloc_linux_parse_cpumap_file . . . . .	108
13.27.2.3 hwloc_linux_set_tid_cpubind . . . . .	108
13.28 Helpers for manipulating Linux libnuma unsigned long masks . . . . .	108
13.28.1 Function Documentation . . . . .	109
13.28.1.1 hwloc_cpuset_from_linux_libnuma_ulongs . . . . .	109
13.28.1.2 hwloc_cpuset_to_linux_libnuma_ulongs . . . . .	109
13.28.1.3 hwloc_nodeuset_from_linux_libnuma_ulongs . . . . .	110
13.28.1.4 hwloc_nodeuset_to_linux_libnuma_ulongs . . . . .	110
13.29 Helpers for manipulating Linux libnuma bitmask . . . . .	110
13.29.1 Function Documentation . . . . .	111
13.29.1.1 hwloc_cpuset_from_linux_libnuma_bitmask . . . . .	111
13.29.1.2 hwloc_cpuset_to_linux_libnuma_bitmask . . . . .	111
13.29.1.3 hwloc_nodeuset_from_linux_libnuma_bitmask . . . . .	111
13.29.1.4 hwloc_nodeuset_to_linux_libnuma_bitmask . . . . .	112
13.30 Helpers for manipulating Linux libnuma nodemask_t . . . . .	112
13.30.1 Function Documentation . . . . .	113
13.30.1.1 hwloc_cpuset_from_linux_libnuma_nodemask . . . . .	113
13.30.1.2 hwloc_cpuset_to_linux_libnuma_nodemask . . . . .	113
13.30.1.3 hwloc_nodeuset_from_linux_libnuma_nodemask . . . . .	113
13.30.1.4 hwloc_nodeuset_to_linux_libnuma_nodemask . . . . .	113
13.31 CUDA Driver API Specific Functions . . . . .	113
13.31.1 Function Documentation . . . . .	114
13.31.1.1 hwloc_cuda_get_device_cpuset . . . . .	114
13.32 CUDA Runtime API Specific Functions . . . . .	114
13.32.1 Function Documentation . . . . .	114
13.32.1.1 hwloc_cudart_get_device_cpuset . . . . .	114
13.33 OpenFabrics-Specific Functions . . . . .	115
13.33.1 Function Documentation . . . . .	115
13.33.1.1 hwloc_ibv_get_device_cpuset . . . . .	115
13.34 Myrinet Express-Specific Functions . . . . .	115

13.34.1 Function Documentation . . . . .	116
13.34.1.1 hwloc_mx_board_get_device_cpuset . . . . .	116
13.34.1.2 hwloc_mx_endpoint_get_device_cpuset . . . . .	116
<b>14 Data Structure Documentation</b>	<b>117</b>
14.1 hwloc_obj_attr_u::hwloc_cache_attr_s Struct Reference . . . . .	117
14.1.1 Detailed Description . . . . .	117
14.1.2 Field Documentation . . . . .	118
14.1.2.1 depth . . . . .	118
14.1.2.2 linesize . . . . .	118
14.1.2.3 size . . . . .	118
14.2 hwloc_obj_attr_u::hwloc_group_attr_s Struct Reference . . . . .	118
14.2.1 Detailed Description . . . . .	118
14.2.2 Field Documentation . . . . .	118
14.2.2.1 depth . . . . .	118
14.3 hwloc_obj Struct Reference . . . . .	119
14.3.1 Detailed Description . . . . .	121
14.3.2 Field Documentation . . . . .	121
14.3.2.1 allowed_cpuset . . . . .	121
14.3.2.2 allowed_nodeset . . . . .	121
14.3.2.3 arity . . . . .	122
14.3.2.4 attr . . . . .	122
14.3.2.5 children . . . . .	122
14.3.2.6 complete_cpuset . . . . .	122
14.3.2.7 complete_nodeset . . . . .	122
14.3.2.8 cpuset . . . . .	123
14.3.2.9 depth . . . . .	123
14.3.2.10 first_child . . . . .	123
14.3.2.11 infos . . . . .	123
14.3.2.12 infos_count . . . . .	123
14.3.2.13 last_child . . . . .	123

14.3.2.14	logical_index	124
14.3.2.15	memory	124
14.3.2.16	name	124
14.3.2.17	next_cousin	124
14.3.2.18	next_sibling	124
14.3.2.19	nodeset	124
14.3.2.20	online_cpuset	125
14.3.2.21	os_index	125
14.3.2.22	os_level	125
14.3.2.23	parent	125
14.3.2.24	prev_cousin	125
14.3.2.25	prev_sibling	125
14.3.2.26	sibling_rank	125
14.3.2.27	type	125
14.3.2.28	userdata	126
14.4	hwloc_obj_attr_u Union Reference	126
14.4.1	Detailed Description	126
14.4.2	Field Documentation	126
14.4.2.1	cache	126
14.4.2.2	group	127
14.5	hwloc_obj_info_s Struct Reference	127
14.5.1	Detailed Description	127
14.5.2	Field Documentation	127
14.5.2.1	name	127
14.5.2.2	value	127
14.6	hwloc_obj_memory_s::hwloc_obj_memory_page_type_s Struct Reference	128
14.6.1	Detailed Description	128
14.6.2	Field Documentation	128
14.6.2.1	count	128
14.6.2.2	size	128

14.7 hwloc_obj_memory_s Struct Reference . . . . .	128
14.7.1 Detailed Description . . . . .	129
14.7.2 Field Documentation . . . . .	129
14.7.2.1 local_memory . . . . .	129
14.7.2.2 page_types . . . . .	129
14.7.2.3 page_types_len . . . . .	129
14.7.2.4 total_memory . . . . .	130
14.8 hwloc_topology_cpupbind_support Struct Reference . . . . .	130
14.8.1 Detailed Description . . . . .	130
14.8.2 Field Documentation . . . . .	130
14.8.2.1 get_proc_cpupbind . . . . .	130
14.8.2.2 get_thisproc_cpupbind . . . . .	130
14.8.2.3 get_thisthread_cpupbind . . . . .	131
14.8.2.4 get_thread_cpupbind . . . . .	131
14.8.2.5 set_proc_cpupbind . . . . .	131
14.8.2.6 set_thisproc_cpupbind . . . . .	131
14.8.2.7 set_thisthread_cpupbind . . . . .	131
14.8.2.8 set_thread_cpupbind . . . . .	131
14.9 hwloc_topology_discovery_support Struct Reference . . . . .	131
14.9.1 Detailed Description . . . . .	132
14.9.2 Field Documentation . . . . .	132
14.9.2.1 pu . . . . .	132
14.10hwloc_topology_membind_support Struct Reference . . . . .	132
14.10.1 Detailed Description . . . . .	133
14.10.2 Field Documentation . . . . .	133
14.10.2.1 alloc_membind . . . . .	133
14.10.2.2 bind_membind . . . . .	133
14.10.2.3 firsttouch_membind . . . . .	133
14.10.2.4 get_area_membind . . . . .	133
14.10.2.5 get_proc_membind . . . . .	133
14.10.2.6 get_thisproc_membind . . . . .	133

---

14.10.2.7	<a href="#">get_thisthread_membind</a>	134
14.10.2.8	<a href="#">interleave_membind</a>	134
14.10.2.9	<a href="#">migrate_membind</a>	134
14.10.2.10	<a href="#">nexttouch_membind</a>	134
14.10.2.11	<a href="#">replicate_membind</a>	134
14.10.2.12	<a href="#">set_area_membind</a>	134
14.10.2.13	<a href="#">set_proc_membind</a>	134
14.10.2.14	<a href="#">set_thisproc_membind</a>	134
14.10.2.15	<a href="#">set_thisthread_membind</a>	135
14.11	<a href="#">hwloc_topology_support Struct Reference</a>	135
14.11.1	<a href="#">Detailed Description</a>	135
14.11.2	<a href="#">Field Documentation</a>	135
14.11.2.1	<a href="#">cpubind</a>	135
14.11.2.2	<a href="#">discovery</a>	135
14.11.2.3	<a href="#">membind</a>	135



# 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 [CLI 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.

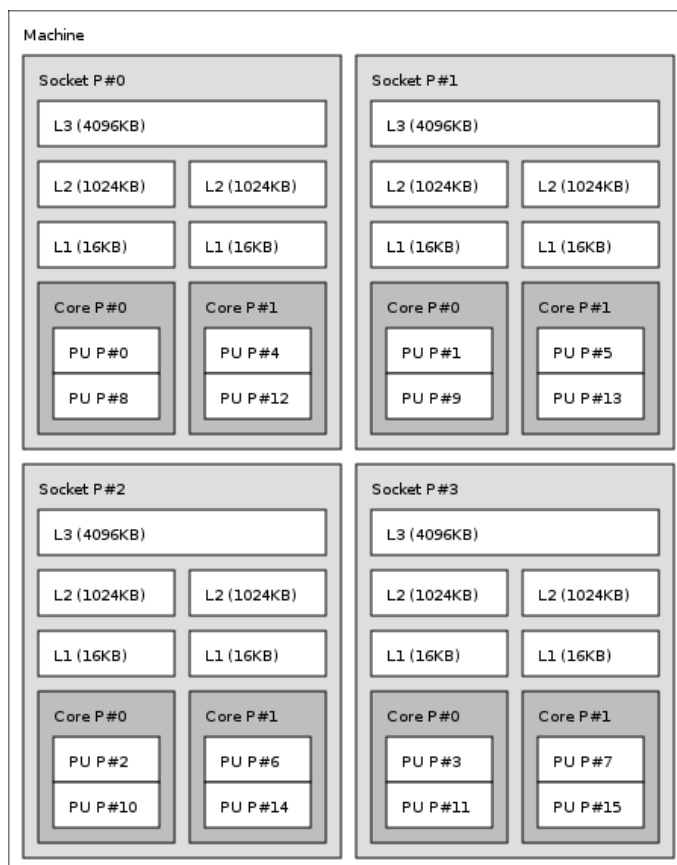
Installation by itself is the fairly common GNU-based process:

```
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 CLI Examples

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



Here's the equivalent output in textual form:

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

```

    PU L#10 (P#6)
    PU L#11 (P#14)
Socket L#3 + L3 L#3 (4096KB)
    L2 L#6 (1024KB) + L1 L#6 (16KB) + Core L#6
    PU L#12 (P#3)
    PU L#13 (P#11)
    L2 L#7 (1024KB) + L1 L#7 (16KB) + Core L#7
    PU L#14 (P#7)
    PU L#15 (P#15)

```

Finally, here's the equivalent output in XML. Long lines were artificially broken for document clarity (in the real output, each XML tag is on a single line), and only socket #0 is shown for brevity:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topology SYSTEM "hwloc.dtd">
<topology>
  <object type="Machine" os_level="-1" os_index="0" cpuset="0x0000ffff"
    complete_cpuset="0x0000ffff" online_cpuset="0x0000ffff"
    allowed_cpuset="0x0000ffff"
    dmi_board_vendor="Dell Computer Corporation" dmi_board_name="0RD318"
    local_memory="16648183808">
    <page_type size="4096" count="4064498"/>
    <page_type size="2097152" count="0"/>
    <object type="Socket" os_level="-1" os_index="0" cpuset="0x00001111"
      complete_cpuset="0x00001111" online_cpuset="0x00001111"
      allowed_cpuset="0x00001111">
      <object type="Cache" os_level="-1" cpuset="0x00001111"
        complete_cpuset="0x00001111" online_cpuset="0x00001111"
        allowed_cpuset="0x00001111" cache_size="4194304" depth="3"
        cache_linesize="64">
        <object type="Cache" os_level="-1" cpuset="0x00000101"
          complete_cpuset="0x00000101" online_cpuset="0x00000101"
          allowed_cpuset="0x00000101" cache_size="1048576" depth="2"
          cache_linesize="64">
          <object type="Cache" os_level="-1" cpuset="0x00000101"
            complete_cpuset="0x00000101" online_cpuset="0x00000101"
            allowed_cpuset="0x00000101" cache_size="16384" depth="1"
            cache_linesize="64">
            <object type="Core" os_level="-1" os_index="0" cpuset="0x00000101"
              complete_cpuset="0x00000101" online_cpuset="0x00000101"
              allowed_cpuset="0x00000101">
              <object type="PU" os_level="-1" os_index="0" cpuset="0x00000001"
                complete_cpuset="0x00000001" online_cpuset="0x00000001"
                allowed_cpuset="0x00000001"/>
              <object type="PU" os_level="-1" os_index="8" cpuset="0x00000100"
                complete_cpuset="0x00000100" online_cpuset="0x00000100"
                allowed_cpuset="0x00000100"/>
            </object>
          </object>
        </object>
      </object>
    </object>
  </object>
  <object type="Cache" os_level="-1" cpuset="0x00001010"
    complete_cpuset="0x00001010" online_cpuset="0x00001010"
    allowed_cpuset="0x00001010" cache_size="1048576" depth="2"
    cache_linesize="64">
    <object type="Cache" os_level="-1" cpuset="0x00001010"

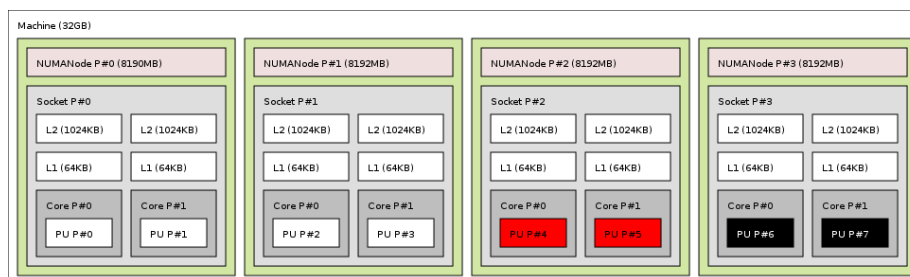
```

```

        complete_cpuset="0x00001010" online_cpuset="0x00001010"
        allowed_cpuset="0x00001010" cache_size="16384" depth="1"
        cache_linesize="64">
<object type="Core" os_level="-1" os_index="1" cpuset="0x00001010"
  complete_cpuset="0x00001010" online_cpuset="0x00001010"
  allowed_cpuset="0x00001010">
  <object type="PU" os_level="-1" os_index="4" cpuset="0x00000010"
    complete_cpuset="0x00000010" online_cpuset="0x00000010"
    allowed_cpuset="0x00000010"/>
  <object type="PU" os_level="-1" os_index="12" cpuset="0x00001000"
    complete_cpuset="0x00001000" online_cpuset="0x00001000"
    allowed_cpuset="0x00001000"/>
  </object>
</object>
</object>
</object>
</object>
<!-- ...other sockets listed here ... -->
</object>
</topology>

```

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



Here's the equivalent output in textual form:

```

Machine (32GB)
  NUMANode L#0 (P#0 8190MB) + Socket L#0
    L2 L#0 (1024KB) + L1 L#0 (64KB) + Core L#0 + PU L#0 (P#0)
    L2 L#1 (1024KB) + L1 L#1 (64KB) + Core L#1 + PU L#1 (P#1)
  NUMANode L#1 (P#1 8192MB) + Socket L#1
    L2 L#2 (1024KB) + L1 L#2 (64KB) + Core L#2 + PU L#2 (P#2)
    L2 L#3 (1024KB) + L1 L#3 (64KB) + Core L#3 + PU L#3 (P#3)
  NUMANode L#2 (P#2 8192MB) + Socket L#2
    L2 L#4 (1024KB) + L1 L#4 (64KB) + Core L#4 + PU L#4 (P#4)
    L2 L#5 (1024KB) + L1 L#5 (64KB) + Core L#5 + PU L#5 (P#5)
  NUMANode L#3 (P#3 8192MB) + Socket L#3
    L2 L#6 (1024KB) + L1 L#6 (64KB) + Core L#6 + PU L#6 (P#6)
    L2 L#7 (1024KB) + L1 L#7 (64KB) + Core L#7 + PU L#7 (P#7)

```

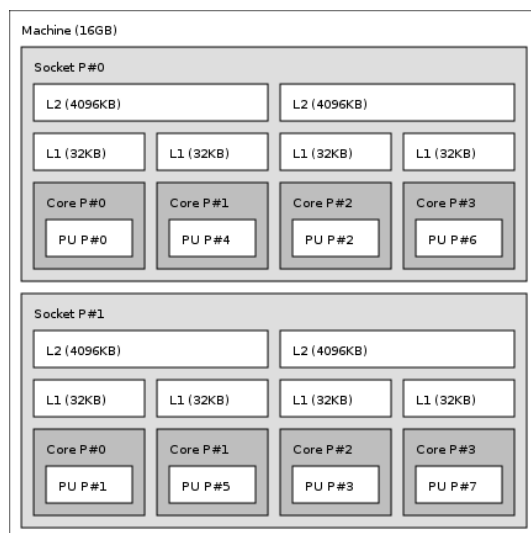
And here's the equivalent output in XML. Similar to above, line breaks were added and only PU #0 is shown for brevity:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topology SYSTEM "hwloc.dtd">
<topology>
  <object type="Machine" os_level="-1" os_index="0" cpuset="0x000000ff"
    complete_cpuset="0x000000ff" online_cpuset="0x000000ff"
    allowed_cpuset="0x000000ff" nodeset="0x000000ff"
    complete_nodeset="0x000000ff" allowed_nodeset="0x000000ff"
    dmi_board_vendor="TYAN Computer Corp" dmi_board_name="S4881 ">
    <page_type size="4096" count="0"/>
    <page_type size="2097152" count="0"/>
    <object type="NUMANode" os_level="-1" os_index="0" cpuset="0x00000003"
      complete_cpuset="0x00000003" online_cpuset="0x00000003"
      allowed_cpuset="0x00000003" nodeset="0x00000001"
      complete_nodeset="0x00000001" allowed_nodeset="0x00000001"
      local_memory="7514177536">
      <page_type size="4096" count="1834516"/>
      <page_type size="2097152" count="0"/>
      <object type="Socket" os_level="-1" os_index="0" cpuset="0x00000003"
        complete_cpuset="0x00000003" online_cpuset="0x00000003"
        allowed_cpuset="0x00000003" nodeset="0x00000001"
        complete_nodeset="0x00000001" allowed_nodeset="0x00000001">
        <object type="Cache" os_level="-1" cpuset="0x00000001"
          complete_cpuset="0x00000001" online_cpuset="0x00000001"
          allowed_cpuset="0x00000001" nodeset="0x00000001"
          complete_nodeset="0x00000001" allowed_nodeset="0x00000001"
          cache_size="1048576" depth="2" cache_linesize="64">
          <object type="Cache" os_level="-1" cpuset="0x00000001"
            complete_cpuset="0x00000001" online_cpuset="0x00000001"
            allowed_cpuset="0x00000001" nodeset="0x00000001"
            complete_nodeset="0x00000001" allowed_nodeset="0x00000001"
            cache_size="65536" depth="1" cache_linesize="64">
            <object type="Core" os_level="-1" os_index="0"
              cpuset="0x00000001" complete_cpuset="0x00000001"
              online_cpuset="0x00000001" allowed_cpuset="0x00000001"
              nodeset="0x00000001" complete_nodeset="0x00000001"
              allowed_nodeset="0x00000001">
              <object type="PU" os_level="-1" os_index="0" cpuset="0x00000001"
                complete_cpuset="0x00000001" online_cpuset="0x00000001"
                allowed_cpuset="0x00000001" nodeset="0x00000001"
                complete_nodeset="0x00000001" allowed_nodeset="0x00000001"/>
            </object>
          </object>
        </object>
      </object>
    <!-- ...more objects listed here ... -->
  </topology>

```

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



Here's the same output in textual form:

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

And the same output in XML (line breaks added, only PU #0 shown):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topology SYSTEM "hwloc.dtd">
<topology>
  <object type="Machine" os_level="-1" os_index="0" cpuset="0x000000ff"
    complete_cpuset="0x000000ff" online_cpuset="0x000000ff"
    allowed_cpuset="0x000000ff" dmi_board_vendor="Dell Inc."
    dmi_board_name="0NR282" local_memory="16865292288">
    <page_type size="4096" count="4117503"/>
    <page_type size="2097152" count="0"/>
    <object type="Socket" os_level="-1" os_index="0" cpuset="0x00000055"
      complete_cpuset="0x00000055" online_cpuset="0x00000055"
      allowed_cpuset="0x00000055">
      <object type="Cache" os_level="-1" cpuset="0x00000011"
        complete_cpuset="0x00000011" online_cpuset="0x00000011"
```



```

        allowed_cpuset="0x00000011" cache_size="4194304" depth="2"
        cache_linesize="64">
<object type="Cache" os_level="-1" cpuset="0x00000001"
    complete_cpuset="0x00000001" online_cpuset="0x00000001"
    allowed_cpuset="0x00000001" cache_size="32768" depth="1"
    cache_linesize="64">
    <object type="Core" os_level="-1" os_index="0" cpuset="0x00000001"
        complete_cpuset="0x00000001" online_cpuset="0x00000001"
        allowed_cpuset="0x00000001">
        <object type="PU" os_level="-1" os_index="0" cpuset="0x00000001"
            complete_cpuset="0x00000001" online_cpuset="0x00000001"
            allowed_cpuset="0x00000001"/>
        </object>
    </object>
</object>
<object type="Cache" os_level="-1" cpuset="0x00000010"
    complete_cpuset="0x00000010" online_cpuset="0x00000010"
    allowed_cpuset="0x00000010" cache_size="32768" depth="1"
    cache_linesize="64">
    <object type="Core" os_level="-1" os_index="1" cpuset="0x00000010"
        complete_cpuset="0x00000010" online_cpuset="0x00000010"
        allowed_cpuset="0x00000010">
        <object type="PU" os_level="-1" os_index="4" cpuset="0x00000010"
            complete_cpuset="0x00000010" online_cpuset="0x00000010"
            allowed_cpuset="0x00000010"/>
        </object>
    </object>
</object>
<!-- ...more objects listed here ... -->
</topology>

```

## 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. Documentation for everything in [hwloc.h](#) are provided later in this document. Developers should also look at [hwloc/helper.h](#) (and also in this document, which provides good higher-level topology traversal examples).

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

Each hwloc object contains a cpuset describing the list of processing units that it contains. These bitmaps may be used for [CPU binding](#) and [Memory binding](#). hwloc offers an extensive bitmap manipulation interface in [hwloc/bitmap.h](#).

Moreover, hwloc also comes with additional helpers for interoperability with several commonly used environments. See the [Interoperability with other software](#) section for details.

The complete API 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/](#).

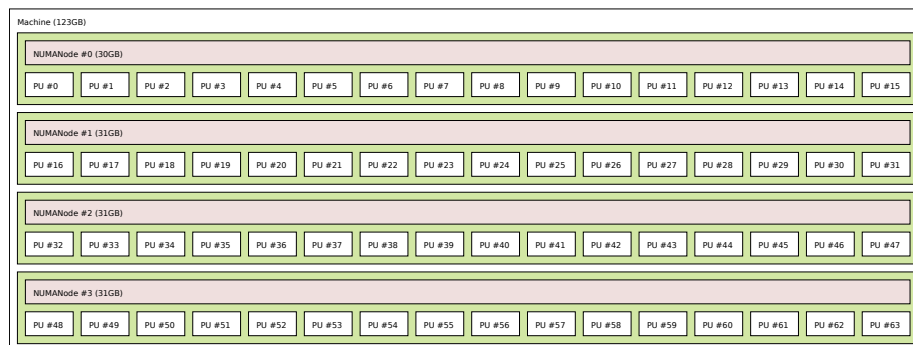
**NOTE:** If you are building the documentation from a Subversion checkout, you will need to have Doxygen and pdf<sub>l</sub>at<sub>e</sub>x installed -- the documentation will be built during the normal "make" process. The documentation is installed during "make install" to \$prefix/share/doc/hwloc/ and your systems default man page tree (under \$prefix, of course).

### 1.4.1 Portability

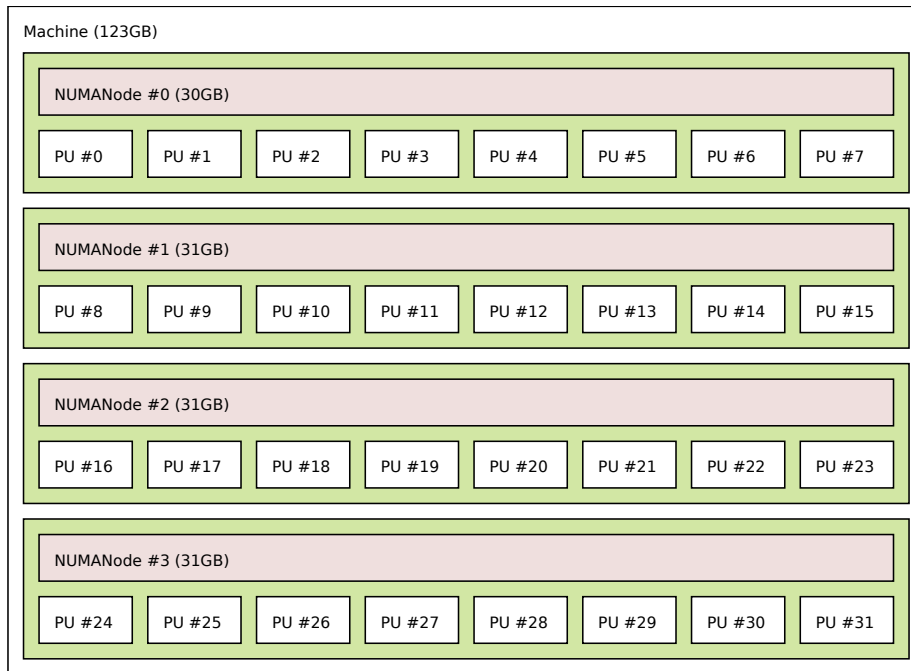
As shown in [CLI Examples](#), hwloc can obtain information on a wide variety of hardware topologies. However, some platforms and/or operating system versions will only report a subset of this information. For example, on an PPC64-based system with 32 cores (each with 2 hardware threads) running a default 2.6.18-based kernel from RHEL 5.4, hwloc is only able to glean information about NUMA nodes and processor units (PUs). No information about caches, sockets, or cores is available.

Similarly, Operating Systems have varying support for CPU and memory binding, e.g. while some Operating Systems provide interfaces for all kinds of CPU and memory bindings, some others provide only interfaces for a limited number of kinds of CPU and memory binding, and some do not provide any binding interface at all. Hwloc's binding functions would then simply return the ENOSYS error (Function not implemented), meaning that the underlying Operating System does not provide any interface for them. [CPU binding](#) and [Memory binding](#) provide more information on which hwloc binding functions should be preferred because interfaces for them are usually available on the supported Operating Systems.

Here's the graphical output from lstopo on this platform when Simultaneous Multi-Threading (SMT) is enabled:



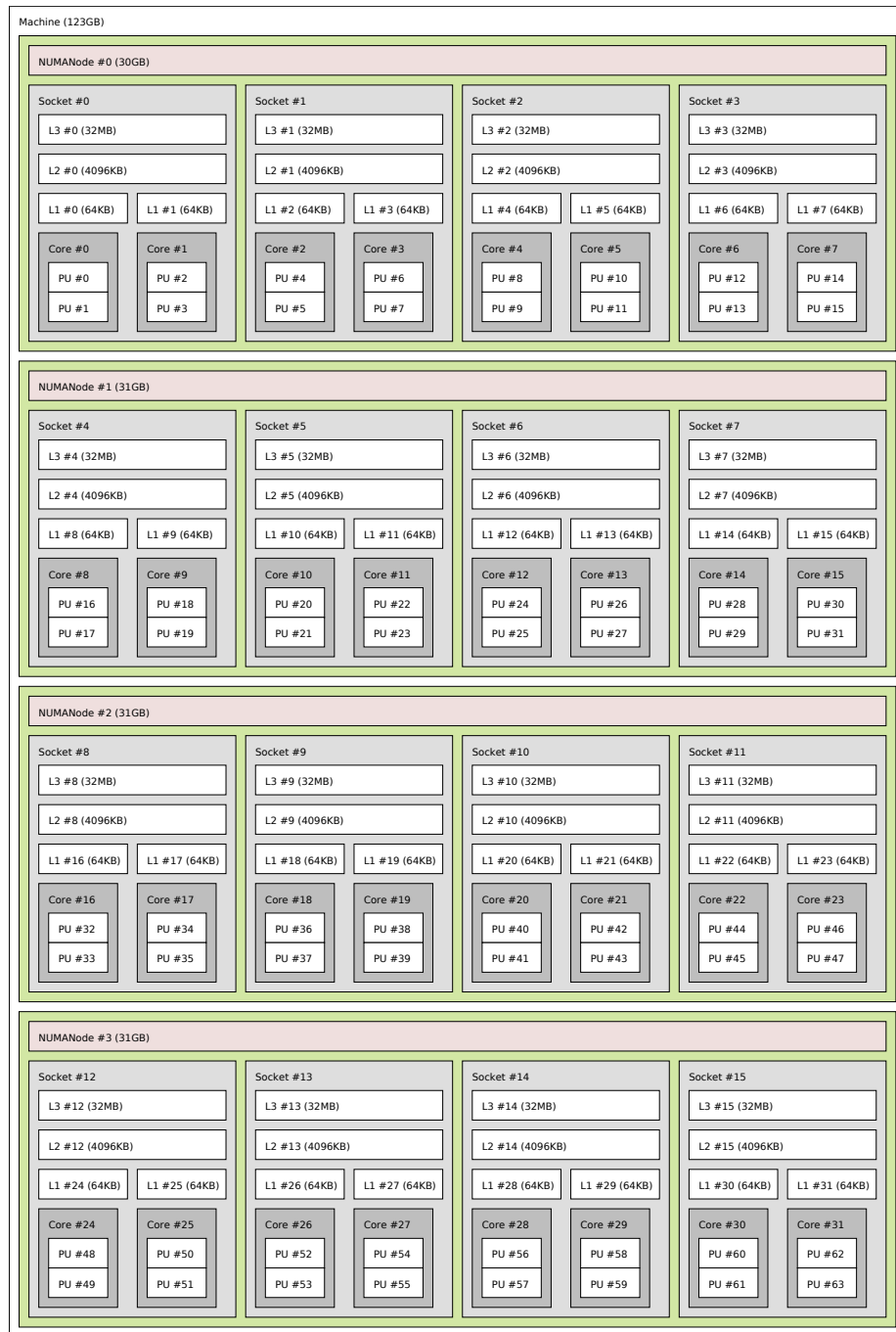
And here's the graphical output from lstopo on this platform when SMT is disabled:



Notice that hwloc only sees half the PUs when SMT is disabled. PU #15, for example, seems to change location from NUMA node #0 to #1. In reality, no PUs "moved" -- they were simply re-numbered when hwloc only saw half as many. Hence, PU #15 in the SMT-disabled picture probably corresponds to PU #30 in the SMT-enabled picture.

This same "PUs have disappeared" effect can be seen on other platforms -- even platforms / OSs that provide much more information than the above PPC64 system. This is an unfortunate side-effect of how operating systems report information to hwloc.

Note that upgrading the Linux kernel on the same PPC64 system mentioned above to 2.6.34, hwloc is able to discover all the topology information. The following picture shows the entire topology layout when SMT is enabled:



Developers using the hwloc API or XML output for portable applications should there-

fore be extremely careful to not make any assumptions about the structure of data that is returned. For example, per the above reported PPC topology, it is not safe to assume that PUs will always be descendants of cores.

Additionally, future hardware may insert new topology elements that are not available in this version of hwloc. Long-lived applications that are meant to span multiple different hardware platforms should also be careful about making structure assumptions. For example, there may someday be an element "lower" than a PU, or perhaps a new element may exist between a core and a PU.

### 1.4.2 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 © 2009-2010 INRIA
 * Copyright © 2009-2010 Université Bordeaux 1
 * Copyright © 2009-2010 Cisco Systems, Inc. All rights reserved.
 *
 * hwloc-hello.c
 */

#include <hwloc.h>
#include <errno.h>
#include <stdio.h>
#include <string.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, n;
    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_nobjs_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_bitmap_dup(obj->cpuset);

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

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

        /* Free our cpuset copy */
        hwloc_bitmap_free(cpuset);
    }

    /******
    * Sixth example:
    * Allocate some memory on the last NUMA node, bind some existing
    * memory to the last NUMA node.
    *****/
    /* Get last node. */
    n = hwloc_get_nobjs_by_type(topology, HWLOC_OBJ_NODE);
    if (n) {
        void *m;
        size_t size = 1024*1024;

```

```

    obj = hwloc_get_obj_by_type(topology, HWLOC_OBJ_NODE, n - 1);
    m = hwloc_alloc_membind_nodeset(topology, size, obj->nodeset,
                                    HWLOC_MEMBIND_DEFAULT, 0);
    hwloc_free(topology, m, size);

    m = malloc(size);
    hwloc_set_area_membind_nodeset(topology, m, size, obj->nodeset,
                                    HWLOC_MEMBIND_DEFAULT, 0);
    free(m);
}

/* 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$
```

## 1.5 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. Please attach the output of lstopo after having given the --enable-debug option to ./configure and rebuilt completely, to get debugging output.

## 1.6 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.

## 1.7 Further Reading

The documentation chapters include

- [Terms and Definitions](#)
- [Command-line tools](#)
- [Environment variables](#)
- [CPU Binding and Memory Binding](#)
- [Interoperability with other software](#)
- [Thread safety](#)
- [Embedding hwloc in other software](#)
- [Switching from PLPA to hwloc](#)
- [Frequently Asked Questions](#)

Make sure to have had a look at those too!

## 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 implemented as the [hwloc\\_bitmap\\_t](#) opaque structure. hwloc CPU sets are just masks, they do *not* have any relation with an operating system actual binding notion like Linux' cpusets.

**Node set** The set of NUMA memory nodes logically included in an object (if it makes sense). They are always expressed using physical node numbers (as announced by the OS). They are implemented with the [hwloc\\_bitmap\\_t](#) opaque structure. as bitmaps.

**Bitmap** A possibly-infinite set of bits used for describing sets of objects such as CPUs (CPU sets) or memory nodes (Node sets). They are implemented with the [hwloc\\_bitmap\\_t](#) opaque structure.

**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 or physical 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.

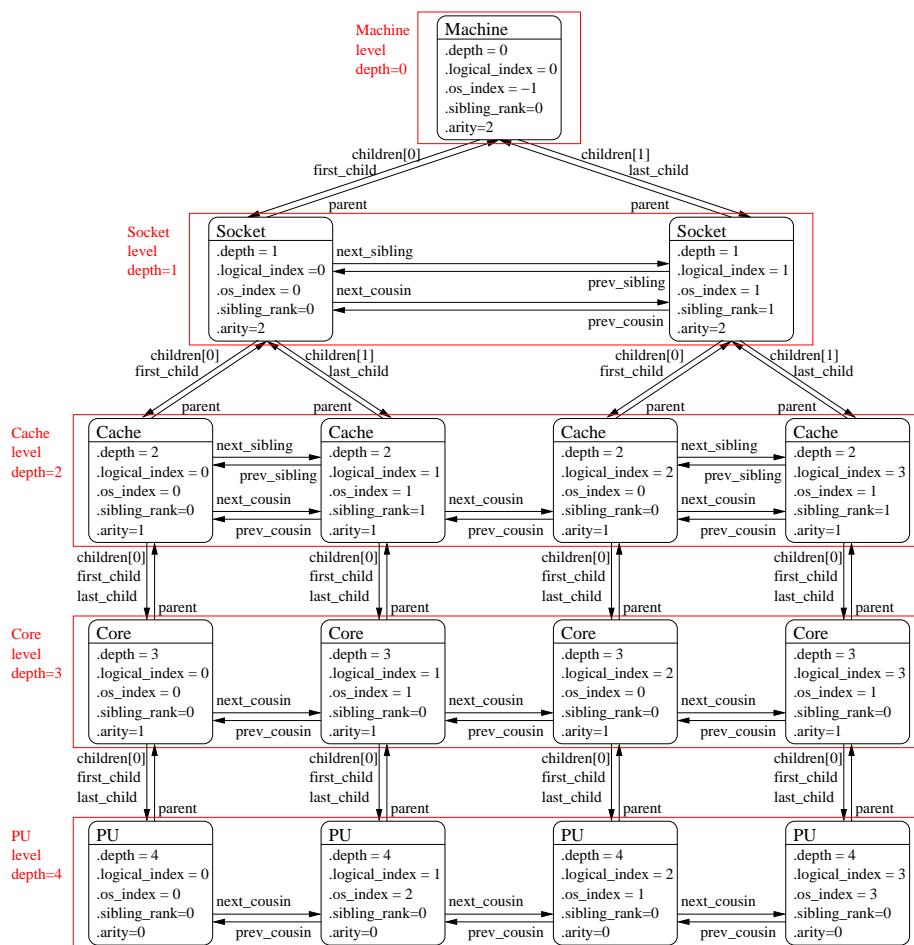
**Depth** Nesting level in the object tree, starting from the 0th object.

**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 graphical 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 bitmaps 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 bitmap strings to pass to hwloc-bind. Although hwloc-bind accepts many forms of object specification (i.e., bitmap strings are one of many forms that hwloc-bind understands), they can be useful, compact representations in shell scripts, for example.

hwloc-calc generates bitmap 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 bitmap 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.

### 3.5 hwloc-ps

hwloc-ps is a tool to display the bindings of processes that are currently running on the local machine. By default, hwloc-ps only lists processes that are bound; unbound process (and Linux kernel threads) are not displayed.



## 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

# CPU Binding and Memory Binding

Some OSes do not systematically provide separate functions for CPU and Memory binding. This means that CPU binding functions may have effects on the memory binding policy, and changing the memory binding policy may change the CPU binding of the current thread. This is often not a problem for the application, so by default hwloc will make use of these functions when they provide better binding support.

If the application does not want any CPU binding change when changing the memory policy, it needs to use the `HWLOC_MEMBIND_NOCPUBIND` flag to prevent hwloc from using OS functions which would change the CPU binding. Conversely, `HWLOC_CPUBIND_NOMEMBIND` can be passed to cpu binding function to prevent hwloc from using OS functions would change the memory binding policy. Of course, this will thus reduce hwloc's support for binding, so their use is discouraged.

One can however avoid using these flags but still closely control both memory and CPU binding, by allocating memory and touching it, and then changing the CPU binding. The already-really-allocated memory will not be migrated, thus even if the memory binding policy gets changed by the CPU binding order, the effect will have been achieved. On binding and allocating further memory, the CPU binding should be performed again in case the memory binding altered the previously-selected CPU binding.

Not all OSes support the notion of a current memory binding policy for the current process but those often still provide a way to allocate data on a given node set. Conversely, some OSes support the notion of a current memory binding policy, and do not permit to allocate data on a given node set without just changing the current policy and allocate the data. Hwloc provides functions that set the current memory binding policies (if supported) as well as functions which allocate memory bound to given node set. By default, it does not use the former to achieve the latter, so that users can use both on OSes where they are both supported, and get both effects at the same time.

For convenience, hwloc however also provides the `hwloc_alloc_membind_policy` and `hwloc_alloc_membind_policy_nodeset` helpers which are allowed to change the current memory binding policy of the process, in order to achieve memory binding even if that means having to change the current memory binding policy.

## Chapter 6

# 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.

**Myrinet Express** [hwloc/myriexpress.h](#) offers interoperability with the Myrinet Express interface. It can return the list of processors near a Myrinet board managed by the MX driver.

**NVIDIA CUDA** [hwloc/cuda.h](#) and [hwloc/cudart.h](#) enable interoperability with NVIDIA CUDA Driver and Runtime interfaces. For instance, it may return the list of processors near NVIDIA GPUs.

**Taskset command-line tool** The taskset command-line tool is widely used for binding processes. It manipulates CPU set strings in a format that is slightly different from hwloc's one (it does not divide the string in fixed-size subsets and separates them with commas). To ease interoperability, hwloc offers routines to convert hwloc CPU sets from/to taskset-specific string format. Most hwloc command-line tools also support the `--taskset` option to manipulate taskset-specific strings.

## Chapter 7

# 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 8

# 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).

## 8.1 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 2.65 are almost certain to not work.

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_SETUP_CORE` *must* be invoked if using the m4 macros):

- `HWLOC_SETUP_CORE(config-dir-prefix, action-upon-success, action-upon-failure, print_banner_or_not)`: 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_SETUP_CORE` 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`. The last argument, if not empty, will cause the macro to display an announcement banner that it is starting the hwloc core configuration tests.

`HWLOC_SETUP_CORE` will set the following environment variables and `AC_SUBST` them: `HWLOC_EMBEDDED_CFLAGS`, `HWLOC_EMBEDDED_CPPFLAGS`, and `HWLOC_EMBEDDED_LIBS`. These flags are filled with the values discovered in the hwloc-specific m4 tests, and can be used in your build process as relevant. The `_CFLAGS`, `_CPPFLAGS`, and `_LIBS` variables are necessary to build `libhwloc` (or `libhwloc_embedded`) itself.

`HWLOC_SETUP_CORE` also sets `HWLOC_EMBEDDED_LDADD` environment variable (and `AC_SUBST`s it) to contain the location of the `libhwloc_embedded.la` convenience Libtool archive. It can be used in your build process to link an application or other library against the embedded hwloc library.

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

- `HWLOC_BUILD_STANDALONE`: `HWLOC_SETUP_CORE` defaults to

building hwloc in an "embedded" mode (described above). If `HWLOC_BUILD_STANDALONE` is invoked *\*before\** `HWLOC_SETUP_CORE`, the embedded definitions will not apply (e.g., `libhwloc.la` will be built, not `libhwloc_embedded.la`).

- `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_SETUP_DOCS`, `HWLOC_SETUP_UTILS`, `HWLOC_SETUP_TESTS`: These three macros only apply when hwloc is built in "standalone" mode (i.e., they should NOT be invoked unless `HWLOC_BUILD_STANDALONE` has already been invoked).
- `HWLOC_DO_AM_CONDITIONALS`: If you embed hwloc in a larger project and build it conditionally with Automake (e.g., if `HWLOC_SETUP_CORE` 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_SETUP_CORE` is invoked unconditionally. If you are not using Automake to build hwloc, this macro is unnecessary (and will actually cause errors because it invoked `AM_*` macros that will be undefined).

**NOTE:** When using the `HWLOC_SETUP_CORE` m4 macro, it may be necessary to explicitly invoke `AC_CANONICAL_TARGET` (which requires `config.sub` and `config.guess`) 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.

Also note that hwloc's top-level `configure.ac` script uses exactly the macros described above to build hwloc in a standalone mode (by default). You may want to examine it for one example of how these macros are used.

## 8.2 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 "$(HWLOC_EMBEDDED_LDADD)" and "$(HWLOC_EMBEDDED_LIBS)" to
    sandbox's executable's LDADD line. The former is the name of the
    Libtool convenience library that hwloc will generate. The latter
    is any dependent support libraries that may be needed by
    $(HWLOC_EMBEDDED_LDADD).
  4. Add "$(HWLOC_EMBEDDED_CFLAGS)" to AM_CFLAGS
  5. Add "$(HWLOC_EMBEDDED_CPPFLAGS)" to AM_CPPFLAGS
shell$ edit configure.ac
  1. Add "HWLOC_SET_SYMBOL_PREFIX(sandbox_hwloc_)" line
  2. Add "HWLOC_SETUP_CORE([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 9

# 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.

### 9.1 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.

### 9.2 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\(\)](#)).

### 9.3 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/bitmap.h](#) for details) and then passed to [hwloc\\_set\\_cpubind\(\)](#) for binding.

### 9.4 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 10

# Frequently Asked Questions

### **10.1 I do not want hwloc to rediscover my enormous machine topology everytime I rerun a process**

Although the topology discovery is not expensive on common machines, its overhead may become significant when multiple processes repeat the discovery on large machines (for instance when starting one process per core in a parallel application). The machine topology usually does not vary much, except if some cores are stopped/restarted or if the administrator restrictions are modified. Thus rediscovering the whole topology again and again may look useless.

For this purpose, hwloc offers XML import/export features. It lets you save the discovered topology to a file (for instance with the `lstopo` program) and reload it later by setting the `HWLOC_XMLFILE` environment variable. Loading a XML topology is usually much faster than querying multiple files or calling multiple functions of the operating system. It is also possible to manipulate such XML files with the C programming interface, and the import/export may also be directed to memory buffer (that may for instance be transmitted between applications through a socket).



# Chapter 11

## Module Index

### 11.1 Modules

Here is a list of all modules:

API version . . . . .	47
Topology context . . . . .	47
Object sets . . . . .	48
Topology Object Types . . . . .	49
Topology Objects . . . . .	51
Create and Destroy Topologies . . . . .	52
Configure Topology Detection . . . . .	53
Tinker with topologies. . . . .	59
Get some Topology Information . . . . .	60
Retrieve Objects . . . . .	63
Object/String Conversion . . . . .	63
CPU binding . . . . .	66
Memory binding . . . . .	70
Object Type Helpers . . . . .	78
Basic Traversal Helpers . . . . .	79
Finding Objects Inside a CPU set . . . . .	82
Finding a single Object covering at least CPU set . . . . .	84
Finding a set of similar Objects covering at least a CPU set . . . . .	85
Cache-specific Finding Helpers . . . . .	86
Advanced Traversal Helpers . . . . .	87
Binding Helpers . . . . .	89
Cpuset Helpers . . . . .	90
Nodeset Helpers . . . . .	92
Conversion between cpuset and nodeset . . . . .	93
The bitmap API . . . . .	95

Helpers for manipulating glibc sched affinity . . . . .	106
Linux-only helpers . . . . .	107
Helpers for manipulating Linux libnuma unsigned long masks . . . . .	108
Helpers for manipulating Linux libnuma bitmask . . . . .	110
Helpers for manipulating Linux libnuma nodemask_t . . . . .	112
CUDA Driver API Specific Functions . . . . .	113
CUDA Runtime API Specific Functions . . . . .	114
OpenFabrics-Specific Functions . . . . .	115
Myrinet Express-Specific Functions . . . . .	115

## Chapter 12

# Data Structure Index

### 12.1 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 ) . .	117
<a href="#">hwloc_obj_attr_u::hwloc_group_attr_s</a> (Group-specific Object Attributes ) . .	118
<a href="#">hwloc_obj</a> (Structure of a topology object ) . . . . .	119
<a href="#">hwloc_obj_attr_u</a> (Object type-specific Attributes ) . . . . .	126
<a href="#">hwloc_obj_info_s</a> (Object info ) . . . . .	127
<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 ) . . . . .	128
<a href="#">hwloc_obj_memory_s</a> (Object memory ) . . . . .	128
<a href="#">hwloc_topology_cpubind_support</a> (Flags describing actual PU binding sup- port for this topology ) . . . . .	130
<a href="#">hwloc_topology_discovery_support</a> (Flags describing actual discovery sup- port for this topology ) . . . . .	131
<a href="#">hwloc_topology_membind_support</a> (Flags describing actual memory bind- ing support for this topology ) . . . . .	132
<a href="#">hwloc_topology_support</a> (Set of flags describing actual support for this topology ) . . . . .	135



# Chapter 13

## Module Documentation

### 13.1 API version

#### Defines

- `#define HWLOC_API_VERSION 0x00010100`

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

#### 13.1.1 Define Documentation

##### 13.1.1.1 `#define HWLOC_API_VERSION 0x00010100`

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

### 13.2 Topology context

#### Typedefs

- `typedef struct hwloc_topology * hwloc_topology_t`

*Topology context.*

## 13.2.1 Typedef Documentation

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

Topology context.

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

## 13.3 Object sets

### Typedefs

- typedef [hwloc\\_bitmap\\_t](#) [hwloc\\_cpuset\\_t](#)  
*A CPU set is a bitmap whose bits are set according to CPU physical OS indexes.*
- typedef [hwloc\\_const\\_bitmap\\_t](#) [hwloc\\_const\\_cpuset\\_t](#)  
*A non-modifiable [hwloc\\_cpuset\\_t](#).*
- typedef [hwloc\\_bitmap\\_t](#) [hwloc\\_nodeset\\_t](#)  
*A node set is a bitmap whose bits are set according to NUMA memory node physical OS indexes.*
- typedef [hwloc\\_const\\_bitmap\\_t](#) [hwloc\\_const\\_nodeset\\_t](#)  
*A non-modifiable [hwloc\\_nodeset\\_t](#).*

## 13.3.1 Typedef Documentation

### 13.3.1.1 typedef hwloc\_const\_bitmap\_t hwloc\_const\_cpuset\_t

A non-modifiable [hwloc\\_cpuset\\_t](#).

### 13.3.1.2 typedef hwloc\_const\_bitmap\_t hwloc\_const\_nodeset\_t

A non-modifiable [hwloc\\_nodeset\\_t](#).

### 13.3.1.3 typedef hwloc\_bitmap\_t hwloc\_cpuset\_t

A CPU set is a bitmap whose bits are set according to CPU physical OS indexes.

It may be consulted and modified with the bitmap API as any [hwloc\\_bitmap\\_t](#) (see [hwloc/bitmap.h](#)).



#### 13.3.1.4 typedef hwloc\_bitmap\_t hwloc\_nodeset\_t

A node set is a bitmap whose bits are set according to NUMA memory node physical OS indexes.

It may be consulted and modified with the bitmap API as any [hwloc\\_bitmap\\_t](#) (see [hwloc/bitmap.h](#)).

When binding memory on a system without any NUMA node (when the whole memory is considered as a single memory bank), the nodeset may be either empty (no memory selected) or full (whole system memory selected).

See also [Conversion between cpuset and nodeset](#).

## 13.4 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.*

### 13.4.1 Enumeration Type Documentation

#### 13.4.1.1 enum hwloc\_compare\_types\_e

##### Enumerator:

***HWLOC\_TYPE\_UNORDERED*** Value returned by [hwloc\\_compare\\_types](#) when types can not be compared.

### 13.4.1.2 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.

## 13.4.2 Function Documentation

### 13.4.2.1 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.

## 13.5 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.*
- struct `hwloc_obj_info_s`  
*Object info.*

### Typedefs

- typedef struct `hwloc_obj` \* `hwloc_obj_t`  
*Convenience typedef; a pointer to a struct `hwloc_obj`.*

### 13.5.1 Typedef Documentation

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

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

## 13.6 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.*

### 13.6.1 Function Documentation

#### 13.6.1.1 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

#### 13.6.1.2 HWLOC\_DECLSPEC void hwloc\_topology\_destroy ( hwloc\_topology\_t topology )

Terminate and free a topology context.

**Parameters**

*topology* is the topology to be freed

**13.6.1.3 HWLOC\_DECLSPEC int hwloc\_topology\_init ( hwloc\_topology\_t \*  
*topology* )**

Allocate a topology context.

**Parameters**

[out] *topology* is assigned a pointer to the new allocated context.

**Returns**

0 on success, -1 on error.

**13.6.1.4 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 [Configure Topology Detection](#) routines. 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](#)

## 13.7 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 PU binding support for this topology.*
- struct [hwloc\\_topology\\_membind\\_support](#)  
*Flags describing actual memory 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 int [hwloc\\_topology\\_set\\_xmlbuffer](#) ([hwloc\\_topology\\_t](#) \_\_hwloc\_restrict topology, const char \*\_\_hwloc\_restrict buffer, int size)

*Enable XML based topology using a memory buffer instead of a file.*

- HWLOC\_DECLSPEC struct [hwloc\\_topology\\_support](#) \* [hwloc\\_topology\\_get\\_support](#) ([hwloc\\_topology\\_t](#) \_\_hwloc\_restrict topology)

*Retrieve the topology support.*

### 13.7.1 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\(\)](#).

### 13.7.2 Enumeration Type Documentation

#### 13.7.2.1 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.

**13.7.3 Function Documentation**

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

Retrieve the topology support.

**13.7.3.2 `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.

**13.7.3.3 `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.



**13.7.3.4** `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.

**13.7.3.5** `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.

**13.7.3.6** `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.

**13.7.3.7** `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  
 -1 is returned and errno is set to ENOSYS on platforms that do not support this feature.

**13.7.3.8 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.

**13.7.3.9 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 from 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.

**13.7.3.10 HWLOC\_DECLSPEC int hwloc\_topology\_set\_xmlbuffer (**  
**hwloc\_topology\_t \_\_hwloc\_restrict topology, const char**  
**\*\_\_hwloc\_restrict buffer, int size )**

Enable XML based topology using a memory buffer instead of a file.

Gather topology information from the XML memory buffer given at `buffer` and of length `length`.

## 13.8 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 void hwloc_topology_export_xmlbuffer (hwloc_topology_t topology, char **xmlbuffer, int *buflen)`  
*Export the topology into a newly-allocated XML memory buffer.*
- `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.*

### 13.8.1 Function Documentation

#### 13.8.1.1 `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()`.

#### 13.8.1.2 `HWLOC_DECLSPEC void hwloc_topology_export_xmlbuffer (hwloc_topology_t topology, char ** xmlbuffer, int * buflen )`

Export the topology into a newly-allocated XML memory buffer.

`xmlbuffer` is allocated by the callee and should be freed with `xmlFree` later in the caller.

This memory buffer may be loaded later through `hwloc_topology_set_xmlbuffer()`.

### 13.8.1.3 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 bitmap cpuset.

cpuset and name will be copied.

#### Returns

the newly-created object

### 13.8.1.4 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

## 13.9 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  
\_\_hwloc\_restrict topology) \_\_hwloc\_attribute\_pure  
*Get the depth of the hierachical 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_nbobjs_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_nbobjs_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 _hwloc_restrict topology) __hwloc_attribute_pure`

*Does the topology context come from this system?*

## 13.9.1 Enumeration Type Documentation

### 13.9.1.1 `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.

## 13.9.2 Function Documentation

### 13.9.2.1 `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.

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

Returns the width of level at depth *depth*.

### 13.9.2.3 static \_\_hwloc\_inline int \_\_hwloc\_attribute\_pure hwloc\_get\_nbobjs\_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.

### 13.9.2.4 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\(\)](#).

### 13.9.2.5 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.

### 13.9.2.6 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).

## 13.10 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`.*

### 13.10.1 Function Documentation

#### 13.10.1.1 `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`.

#### 13.10.1.2 `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()`.

## 13.11 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.*

- `static __hwloc_inline char *__hwloc_attribute_pure hwloc_obj_get_info_by_name (hwloc_obj_t obj, const char *name)`

*Search the given key name in object infos and return the corresponding value.*

### 13.11.1 Function Documentation

**13.11.1.1** `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.

**13.11.1.2** `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`).

**13.11.1.3** `static __hwloc_inline char* __hwloc_attribute_pure  
hwloc_obj_get_info_by_name( hwloc_obj_t obj, const char * name  
) [static]`

Search the given key name in object infos and return the corresponding value.

**Returns**

NULL if no such key exists.

**13.11.1.4** `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.

**13.11.1.5** `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.

**13.11.1.6 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.

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

Return a stringified topology object type.

## 13.12 CPU binding

### Enumerations

- enum [hwloc\\_cpubind\\_flags\\_t](#) { [HWLOC\\_CPUBIND\\_PROCESS](#), [HWLOC\\_CPUBIND\\_THREAD](#), [HWLOC\\_CPUBIND\\_STRICT](#), [HWLOC\\_CPUBIND\\_NOMEMBIND](#) }

*Process/Thread binding flags.*

### Functions

- HWLOC\_DECLSPEC int [hwloc\\_set\\_cpubind](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_const\\_cpuset\\_t](#) set, int flags)

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

- HWLOC\_DECLSPEC int [hwloc\\_get\\_cpubind](#) ([hwloc\\_topology\\_t](#) topology, [hwloc\\_cpuset\\_t](#) set, int flags)

*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 flags)

*Bind a process *pid* on cpus given in bitmap set.*

- `HWLOC_DECLSPEC int hwloc_get_proc_cpubind (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_cpuset_t set, int flags)`  
*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 flags)`  
*Bind a thread `tid` on cpus given in bitmap `set`.*
- `HWLOC_DECLSPEC int hwloc_get_thread_cpubind (hwloc_topology_t topology, hwloc_thread_t tid, hwloc_cpuset_t set, int flags)`  
*Get the current binding of thread `tid`.*

### 13.12.1 Detailed Description

It is often useful to call `hwloc_bitmap_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. -1 is returned and `errno` is set to `ENOSYS` when it is not possible to bind the requested kind of object processes/threads. `errno` is set to `EXDEV` 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.

On some OSes, CPU binding may have effects on memory binding, see [HWLOC\\_CPUBIND\\_NOMEMBIND](#)

## 13.12.2 Enumeration Type Documentation

### 13.12.2.1 enum hwloc\_cpubind\_flags\_t

Process/Thread binding flags.

These flags can be used to refine the binding policy.

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

#### Note

Not all systems support all kinds of binding.

#### Enumerator:

**HWLOC\_CPUBIND\_PROCESS** Bind all threads of the current (possibly) multithreaded process.

**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.

**HWLOC\_CPUBIND\_NOMEMBIND** Avoid any effect on memory binding. On some OSes, some CPU binding function would also bind the memory on the corresponding NUMA node. It is often not a problem for the application, but if it is, setting this flag will make hwloc avoid using OS functions that would also bind memory. This will however reduce the support of CPU bindings, i.e. potentially return -1 with errno set to ENOSYS in some cases.

### 13.12.3 Function Documentation

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

Get current process or thread binding.

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

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 `flags`.

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

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 `flags`.

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

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

##### Returns

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

### 13.12.3.5 `HWLOC_DECLSPEC int hwloc_set_proc_cpubind ( hwloc_topology_t topology, hwloc_pid_t pid, hwloc_const_cpuset_t set, int flags )`

Bind a process `pid` on cpus given in bitmap `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 `flags`.

### 13.12.3.6 `HWLOC_DECLSPEC int hwloc_set_thread_cpubind ( hwloc_topology_t topology, hwloc_thread_t tid, hwloc_const_cpuset_t set, int flags )`

Bind a thread `tid` on cpus given in bitmap `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 `flags`.

## 13.13 Memory binding

### Enumerations

- enum `hwloc_membind_policy_t` {  
`HWLOC_MEMBIND_DEFAULT`, `HWLOC_MEMBIND_FIRSTTOUCH`,  
`HWLOC_MEMBIND_BIND`, `HWLOC_MEMBIND_INTERLEAVE`,  
`HWLOC_MEMBIND_REPLICATE`, `HWLOC_MEMBIND_NEXTTOUCH`  
}

*Memory binding policy.*

- enum `hwloc_membind_flags_t` {  
`HWLOC_MEMBIND_PROCESS`, `HWLOC_MEMBIND_THREAD`,  
`HWLOC_MEMBIND_STRICT`, `HWLOC_MEMBIND_MIGRATE`,  
`HWLOC_MEMBIND_NOCPUBIND` }

*Memory binding flags.*

## Functions

- `HWLOC_DECLSPEC int hwloc_set_membind_nodeset (hwloc_topology_t topology, hwloc_const_nodeset_t nodeset, hwloc_membind_policy_t policy, int flags)`  
*Bind current process memory on the given nodeset `nodeset`.*
- `HWLOC_DECLSPEC int hwloc_set_membind (hwloc_topology_t topology, hwloc_const_cpuset_t cpuset, hwloc_membind_policy_t policy, int flags)`  
*Bind current process memory on memory nodes near the given cpuset `cpuset`.*
- `HWLOC_DECLSPEC int hwloc_get_membind_nodeset (hwloc_topology_t topology, hwloc_nodeset_t nodeset, hwloc_membind_policy_t *policy, int flags)`  
*Get current process memory binding in nodeset `nodeset`.*
- `HWLOC_DECLSPEC int hwloc_get_membind (hwloc_topology_t topology, hwloc_cpuset_t cpuset, hwloc_membind_policy_t *policy, int flags)`  
*Get current process memory binding in cpuset `cpuset`.*
- `HWLOC_DECLSPEC int hwloc_set_proc_membind_nodeset (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_const_nodeset_t nodeset, hwloc_membind_policy_t policy, int flags)`  
*Bind given process memory on the given nodeset `nodeset`.*
- `HWLOC_DECLSPEC int hwloc_set_proc_membind (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_const_cpuset_t cpuset, hwloc_membind_policy_t policy, int flags)`  
*Bind given process memory on memory nodes near the given cpuset `cpuset`.*
- `HWLOC_DECLSPEC int hwloc_get_proc_membind_nodeset (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_nodeset_t nodeset, hwloc_membind_policy_t *policy, int flags)`  
*Get current process memory binding in nodeset `nodeset`.*
- `HWLOC_DECLSPEC int hwloc_get_proc_membind (hwloc_topology_t topology, hwloc_pid_t pid, hwloc_cpuset_t cpuset, hwloc_membind_policy_t *policy, int flags)`  
*Get current process memory binding in cpuset `cpuset`.*
- `HWLOC_DECLSPEC int hwloc_set_area_membind_nodeset (hwloc_topology_t topology, const void *addr, size_t len, hwloc_const_nodeset_t nodeset, hwloc_membind_policy_t policy, int flags)`  
*Bind some memory range on the given nodeset `nodeset`.*

- HWLOC\_DECLSPEC int [hwloc\\_set\\_area\\_mbind](#) ([hwloc\\_topology\\_t](#) topology, const void \*addr, size\_t len, [hwloc\\_const\\_cpuset\\_t](#) cpuset, [hwloc\\_mbind\\_policy\\_t](#) policy, int flags)

*Bind some memory range on memory nodes near the given cpuset cpuset.*

- HWLOC\_DECLSPEC int [hwloc\\_get\\_area\\_mbind\\_nodeset](#) ([hwloc\\_topology\\_t](#) topology, const void \*addr, size\_t len, [hwloc\\_nodeset\\_t](#) nodeset, [hwloc\\_mbind\\_policy\\_t](#) \*policy, int flags)

*Get some memory range memory binding in nodeset nodeset.*

- HWLOC\_DECLSPEC int [hwloc\\_get\\_area\\_mbind](#) ([hwloc\\_topology\\_t](#) topology, const void \*addr, size\_t len, [hwloc\\_cpuset\\_t](#) cpuset, [hwloc\\_mbind\\_policy\\_t](#) \*policy, int flags)

*Get some memory range memory binding in cpuset cpuset.*

- HWLOC\_DECLSPEC void \* [hwloc\\_alloc](#) ([hwloc\\_topology\\_t](#) topology, size\_t len)

*Allocate some memory.*

- HWLOC\_DECLSPEC void \* [hwloc\\_alloc\\_mbind\\_nodeset](#) ([hwloc\\_topology\\_t](#) topology, size\_t len, [hwloc\\_const\\_nodeset\\_t](#) nodeset, [hwloc\\_mbind\\_policy\\_t](#) policy, int flags) \_\_hwloc\_attribute\_malloc

*Allocate some memory on the given nodeset nodeset.*

- HWLOC\_DECLSPEC void \* [hwloc\\_alloc\\_mbind](#) ([hwloc\\_topology\\_t](#) topology, size\_t len, [hwloc\\_const\\_cpuset\\_t](#) cpuset, [hwloc\\_mbind\\_policy\\_t](#) policy, int flags) \_\_hwloc\_attribute\_malloc

*Allocate some memory on memory nodes near the given cpuset cpuset.*

- HWLOC\_DECLSPEC int [hwloc\\_free](#) ([hwloc\\_topology\\_t](#) topology, void \*addr, size\_t len)

*Free some memory allocated by [hwloc\\_alloc\(\)](#) or [hwloc\\_alloc\\_mbind\(\)](#).*

### 13.13.1 Detailed Description

#### Note

Not all OSes support all ways to bind existing allocated memory (migration), future memory allocation, explicit memory allocation, etc. and the corresponding binding functions may fail. -1 is returned and errno is set to ENOSYS when it is not possible to bind the requested kind of object processes/threads). errno is set to EXDEV when the requested cpuset can not be enforced (e.g. some systems only allow one NUMA node).



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

```
hwloc_alloc_membind_policy(topology, size, set, HWLOC_MEMBIND_DEFAULT, 0),
```

which will try to allocate new data bound to the given set, possibly by changing the current memory binding policy, or at worse allocate memory without binding it at all. Since `HWLOC_MEMBIND_STRICT` is not given, this will even not fail unless a mere `malloc()` itself would fail, i.e. `ENOMEM`.

Each binding is available with a CPU set argument or a NUMA memory node set argument. The name of the latter ends with `_nodeset`. It is also possible to convert between CPU set and node set using [hwloc\\_cpuset\\_to\\_nodeset](#) or [hwloc\\_cpuset\\_from\\_nodeset](#).

#### Note

On some OSes, memory binding may have effects on CPU binding, see [HWLOC\\_MEMBIND\\_NOCPUBIND](#)

### 13.13.2 Enumeration Type Documentation

#### 13.13.2.1 enum hwloc\_membind\_flags\_t

Memory binding flags.

These flags can be used to refine the binding policy.

#### Note

Not all systems support all kinds of binding.

#### Enumerator:

***HWLOC\_MEMBIND\_PROCESS*** Set policy for all threads of the current (possibly multithreaded) process.

***HWLOC\_MEMBIND\_THREAD*** Set policy for the current thread of the current process.

***HWLOC\_MEMBIND\_STRICT*** Request strict binding from the OS. The function will fail if the binding can not be completely enforced.

***HWLOC\_MEMBIND\_MIGRATE*** Migrate existing allocated memory. If memory can not be migrated and the `STRICT` flag is passed, an error will be returned.

***HWLOC\_MEMBIND\_NOCPUBIND*** Avoid any effect on CPU binding. On some OSes, some memory binding function would also bind the application on the corresponding CPUs. It is often not a problem for the application,

but if it is, setting this flag will make hwloc avoid using OS functions that would also bind on CPUs. This will however reduce the support of memory bindings, i.e. potentially return ENOSYS in some cases.

### 13.13.2.2 enum hwloc\_membind\_policy\_t

Memory binding policy.

These can be used to choose the binding policy.

Note that not all systems support all kinds of binding.

#### Enumerator:

**HWLOC\_MEMBIND\_DEFAULT** Reset the memory allocation policy to the system default.

**HWLOC\_MEMBIND\_FIRSTTOUCH** Allocate memory on the given nodes, but preferably on the node where the first accessor is running.

**HWLOC\_MEMBIND\_BIND** Allocate memory on the given nodes.

**HWLOC\_MEMBIND\_INTERLEAVE** Allocate memory on the given nodes in a round-robin manner.

**HWLOC\_MEMBIND\_REPLICATE** Replicate memory on the given nodes.

**HWLOC\_MEMBIND\_NEXTTOUCH** On next touch of existing allocated memory, migrate it to the node where the memory reference happened.

### 13.13.3 Function Documentation

#### 13.13.3.1 HWLOC\_DECLSPEC void\* hwloc\_alloc ( hwloc\_topology\_t topology, size\_t len )

Allocate some memory.

This is equivalent to malloc(), except it tries to allocated page-aligned memory from the OS.

#### Note

The allocated memory should be freed with [hwloc\\_free\(\)](#).

#### 13.13.3.2 HWLOC\_DECLSPEC void\* hwloc\_alloc\_membind ( hwloc\_topology\_t topology, size\_t len, hwloc\_const\_cpuset\_t cpuset, hwloc\_membind\_policy\_t policy, int flags )

Allocate some memory on memory nodes near the given cpuset cpuset.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported and `HWLOC_MEMBIND_STRICT` is given
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced and `HWLOC_MEMBIND_STRICT` is given

**Note**

The allocated memory should be freed with [hwloc\\_free\(\)](#).

**13.13.3.3** `HWLOC_DECLSPEC void* hwloc_alloc_membind_nodeuset ( hwloc_topology_t topology, size_t len, hwloc_const_nodeuset_t nodeuset, hwloc_membind_policy_t policy, int flags )`

Allocate some memory on the given nodeuset `nodeuset`.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported and `HWLOC_MEMBIND_STRICT` is given
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced and `HWLOC_MEMBIND_STRICT` is given

**Note**

The allocated memory should be freed with [hwloc\\_free\(\)](#).

**13.13.3.4** `HWLOC_DECLSPEC int hwloc_free ( hwloc_topology_t topology, void * addr, size_t len )`

Free some memory allocated by [hwloc\\_alloc\(\)](#) or [hwloc\\_alloc\\_membind\(\)](#).

**13.13.3.5** `HWLOC_DECLSPEC int hwloc_get_area_membind ( hwloc_topology_t topology, const void * addr, size_t len, hwloc_cpuset_t cpuset, hwloc_membind_policy_t * policy, int flags )`

Get some memory range memory binding in cpuset `cpuset`.

**13.13.3.6** `HWLOC_DECLSPEC int hwloc_get_area_membind_nodeset ( hwloc_topology_t topology, const void * addr, size_t len, hwloc_nodeset_t nodeset, hwloc_membind_policy_t * policy, int flags )`

Get some memory range memory binding in nodeset `nodeset`.

**13.13.3.7** `HWLOC_DECLSPEC int hwloc_get_membind ( hwloc_topology_t topology, hwloc_cpuset_t cpuset, hwloc_membind_policy_t * policy, int flags )`

Get current process memory binding in cpuset `cpuset`.

**13.13.3.8** `HWLOC_DECLSPEC int hwloc_get_membind_nodeset ( hwloc_topology_t topology, hwloc_nodeset_t nodeset, hwloc_membind_policy_t * policy, int flags )`

Get current process memory binding in nodeset `nodeset`.

**13.13.3.9** `HWLOC_DECLSPEC int hwloc_get_proc_membind ( hwloc_topology_t topology, hwloc_pid_t pid, hwloc_cpuset_t cpuset, hwloc_membind_policy_t * policy, int flags )`

Get current process memory binding in cpuset `cpuset`.

**13.13.3.10** `HWLOC_DECLSPEC int hwloc_get_proc_membind_nodeset ( hwloc_topology_t topology, hwloc_pid_t pid, hwloc_nodeset_t nodeset, hwloc_membind_policy_t * policy, int flags )`

Get current process memory binding in nodeset `nodeset`.

**13.13.3.11** `HWLOC_DECLSPEC int hwloc_set_area_membind ( hwloc_topology_t topology, const void * addr, size_t len, hwloc_const_cpuset_t cpuset, hwloc_membind_policy_t policy, int flags )`

Bind some memory range on memory nodes near the given cpuset `cpuset`.

### Returns

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

**13.13.3.12** **HWLOC\_DECLSPEC int hwloc\_set\_area\_membind\_nodeset**  
( **hwloc\_topology\_t topology**, **const void \* addr**, **size\_t len**,  
**hwloc\_const\_nodeset\_t nodeset**, **hwloc\_membind\_policy\_t policy**,  
**int flags** )

Bind some memory range on the given nodeset `nodeset`.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

**13.13.3.13** **HWLOC\_DECLSPEC int hwloc\_set\_membind** ( **hwloc\_topology\_t topology**, **hwloc\_const\_cpuset\_t cpuset**, **hwloc\_membind\_policy\_t policy**, **int flags** )

Bind current process memory on memory nodes near the given cpuset `cpuset`.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

**13.13.3.14** **HWLOC\_DECLSPEC int hwloc\_set\_membind\_nodeset** ( **hwloc\_topology\_t topology**, **hwloc\_const\_nodeset\_t nodeset**, **hwloc\_membind\_policy\_t policy**, **int flags** )

Bind current process memory on the given nodeset `nodeset`.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

**13.13.3.15** **HWLOC\_DECLSPEC int hwloc\_set\_proc\_membind** ( **hwloc\_topology\_t topology**, **hwloc\_pid\_t pid**, **hwloc\_const\_cpuset\_t cpuset**, **hwloc\_membind\_policy\_t policy**, **int flags** )

Bind given process memory on memory nodes near the given cpuset `cpuset`.

**Returns**

- 1 with `errno` set to `ENOSYS` if the action is not supported
- 1 with `errno` set to `EXDEV` if the binding cannot be enforced

**13.13.3.16 HWLOC\_DECLSPEC int hwloc\_set\_proc\_membind\_nodeset**  
 ( hwloc\_topology\_t *topology*, hwloc\_pid\_t *pid*,  
 hwloc\_const\_nodeset\_t *nodeset*, hwloc\_membind\_policy\_t *policy*,  
 int *flags* )

Bind given process memory on the given nodeset *nodeset*.

#### Returns

- 1 with errno set to ENOSYS if the action is not supported
- 1 with errno set to EXDEV if the binding cannot be enforced

## 13.14 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.*

### 13.14.1 Function Documentation

**13.14.1.1 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*.

**13.14.1.2 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*.

## 13.15 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*.

### 13.15.1 Function Documentation

**13.15.1.1** `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`.

**13.15.1.2** `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`.

**13.15.1.3** `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`.

**13.15.1.4** `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.

**13.15.1.5** `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`.



**13.15.1.6** `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\(\)](#).

**13.15.1.7** `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.

**13.15.1.8** `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()`.

**13.15.1.9** `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`.

## 13.16 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.*

### 13.16.1 Function Documentation

**13.16.1.1** `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.

**13.16.1.2** `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*.

**13.16.1.3** `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*.

**13.16.1.4** `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.

**13.16.1.5** `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`.

**13.16.1.6** `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\(\)](#).

**13.16.1.7** `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`.

**13.16.1.8** `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\(\)](#).

## 13.17 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`.*

### 13.17.1 Function Documentation

**13.17.1.1** 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.

**13.17.1.2** 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.

## 13.18 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`.*

### 13.18.1 Function Documentation

- 13.18.1.1** 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`.

- 13.18.1.2** 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.

## 13.19 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.*

### 13.19.1 Function Documentation

**13.19.1.1** `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

**13.19.1.2** `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

## 13.20 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.*

### 13.20.1 Function Documentation

**13.20.1.1** `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`.

**13.20.1.2** `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.

**13.20.1.3** `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.



## 13.21 Binding Helpers

### Functions

- static \_\_hwloc\_inline void `hwloc_distributev` (`hwloc_topology_t` topology, `hwloc_obj_t` \*root, unsigned `n_roots`, `hwloc_cpuset_t` \*cpuset, unsigned `n`, unsigned until)

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

- static \_\_hwloc\_inline void `hwloc_distribute` (`hwloc_topology_t` topology, `hwloc_obj_t` root, `hwloc_cpuset_t` \*cpuset, unsigned `n`, unsigned until)
- static \_\_hwloc\_inline void \* `hwloc_alloc_membind_policy_nodeset` (`hwloc_topology_t` topology, `size_t` len, `hwloc_const_nodeset_t` nodeset, `hwloc_membind_policy_t` policy, int flags)

*Allocate some memory on the given nodeset `nodeset`.*

- static \_\_hwloc\_inline void \* `hwloc_alloc_membind_policy` (`hwloc_topology_t` topology, `size_t` len, `hwloc_const_cpuset_t` cpuset, `hwloc_membind_policy_t` policy, int flags)

*Allocate some memory on the memory nodes near given cpuset `cpuset`.*

### 13.21.1 Function Documentation

**13.21.1.1** static \_\_hwloc\_inline void\* `hwloc_alloc_membind_policy` (`hwloc_topology_t` topology, `size_t` len, `hwloc_const_cpuset_t` cpuset, `hwloc_membind_policy_t` policy, int flags ) [static]

Allocate some memory on the memory nodes near given cpuset `cpuset`.

This is similar to `hwloc_alloc_membind_policy_nodeset`, but for a given cpuset.

**13.21.1.2** static \_\_hwloc\_inline void\* `hwloc_alloc_membind_policy_nodeset` (`hwloc_topology_t` topology, `size_t` len, `hwloc_const_nodeset_t` nodeset, `hwloc_membind_policy_t` policy, int flags ) [static]

Allocate some memory on the given nodeset `nodeset`.

This is similar to `hwloc_alloc_membind` except that it is allowed to change the current memory binding policy, thus providing more binding support, at the expense of changing the current state.

**13.21.1.3** `static __hwloc_inline void hwloc_distribute ( hwloc_topology_t topology, hwloc_obj_t root, hwloc_cpuset_t * cpuset, unsigned n, unsigned until ) [static]`

**13.21.1.4** `static __hwloc_inline void hwloc_distributev ( hwloc_topology_t topology, hwloc_obj_t * roots, unsigned n_roots, hwloc_cpuset_t * cpuset, unsigned n, unsigned until ) [static]`

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

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

Array `cpuset` will be filled with `n` cpusets recursively distributed linearly over the topology under `root`, down to depth `until` (which can be `MAX_INT` to distribute down to the finest level).

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\\_bitmap\\_singlify\(\)](#) before binding a thread so that it does not move at all.

This is the same as `hwloc_distribute`, but takes an array of roots instead of just one root.

## 13.22 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.*

### 13.22.1 Function Documentation

**13.22.1.1** `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.

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

**13.22.1.3** `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.

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

## 13.23 Nodeset Helpers

### Functions

- `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure hwloc_  
topology_get_complete_nodeset (hwloc_topology_t topology)`
- `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure hwloc_  
topology_get_topology_nodeset (hwloc_topology_t topology)`
- `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure hwloc_  
topology_get_allowed_nodeset (hwloc_topology_t topology)`

*Get allowed node set.*

### 13.23.1 Function Documentation

**13.23.1.1** `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure  
hwloc_topology_get_allowed_nodeset ( hwloc_topology_t topology )  
[static]`

Get allowed node set.

#### Returns

the node set of allowed memory of the system. If the topology is the result of a combination of several systems, NULL is returned.

#### Note

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

13.23.1.2 `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure  
hwloc_topology_get_complete_nodeset ( hwloc_topology_t topology  
) [static]`

13.23.1.3 `static __hwloc_inline hwloc_const_nodeset_t __hwloc_attribute_pure  
hwloc_topology_get_topology_nodeset ( hwloc_topology_t topology )  
[static]`

## 13.24 Conversion between cpuset and nodeset

### Functions

- `static __hwloc_inline void hwloc_cpuset_to_nodeset (hwloc_topology_t topology, hwloc_const_cpuset_t cpuset, hwloc_nodeset_t nodeset)`  
*Convert a CPU set into a NUMA node set and handle non-NUMA cases.*
- `static __hwloc_inline void hwloc_cpuset_to_nodeset_strict (struct hwloc_topology *topology, hwloc_const_cpuset_t cpuset, hwloc_nodeset_t nodeset)`  
*Convert a CPU set into a NUMA node set without handling non-NUMA cases.*
- `static __hwloc_inline void hwloc_cpuset_from_nodeset (hwloc_topology_t topology, hwloc_cpuset_t cpuset, hwloc_const_nodeset_t nodeset)`  
*Convert a NUMA node set into a CPU set and handle non-NUMA cases.*
- `static __hwloc_inline void hwloc_cpuset_from_nodeset_strict (struct hwloc_topology *topology, hwloc_cpuset_t cpuset, hwloc_const_nodeset_t nodeset)`  
*Convert a NUMA node set into a CPU set without handling non-NUMA cases.*

### 13.24.1 Detailed Description

There are two semantics for converting cpusets to nodesets depending on how non-NUMA machines are handled.

When manipulating nodesets for memory binding, non-NUMA machines should be considered as having a single NUMA node. The standard conversion routines below should be used so that marking the first bit of the nodeset means that memory should be bound to a non-NUMA whole machine.

When manipulating nodesets as an actual list of NUMA nodes without any need to handle memory binding on non-NUMA machines, the strict conversion routines may be used instead.

## 13.24.2 Function Documentation

**13.24.2.1** `static __hwloc_inline void hwloc_cpuset_from_nodeset  
( hwloc_topology_t topology, hwloc_cpuset_t cpuset,  
hwloc_const_nodeset_t nodeset ) [static]`

Convert a NUMA node set into a CPU set and handle non-NUMA cases.

If the topology contains no NUMA nodes, the machine is considered as a single memory node, and the following behavior is used: If `nodeset` is empty, `cpuset` will be emptied as well. Otherwise `cpuset` will be entirely filled. This is useful for manipulating memory binding sets.

**13.24.2.2** `static __hwloc_inline void hwloc_cpuset_from_nodeset_strict  
( struct hwloc_topology * topology, hwloc_cpuset_t cpuset,  
hwloc_const_nodeset_t nodeset ) [static]`

Convert a NUMA node set into a CPU set without handling non-NUMA cases.

This is the strict variant of [hwloc\\_cpuset\\_from\\_nodeset](#). It does not fix non-NUMA cases. If the topology contains some NUMA nodes, behave exactly the same. However, if the topology contains no NUMA nodes, return an empty `cpuset`.

**13.24.2.3** `static __hwloc_inline void hwloc_cpuset_to_nodeset (`  
`hwloc_topology_t topology, hwloc_const_cpuset_t cpuset,`  
`hwloc_nodeset_t nodeset ) [static]`

Convert a CPU set into a NUMA node set and handle non-NUMA cases.

If some NUMA nodes have no CPUs at all, this function never sets their indexes in the output node set, even if a full CPU set is given in input.

If the topology contains no NUMA nodes, the machine is considered as a single memory node, and the following behavior is used: If `cpuset` is empty, `nodeset` will be emptied as well. Otherwise `nodeset` will be entirely filled.

**13.24.2.4** `static __hwloc_inline void hwloc_cpuset_to_nodeset_strict ( struct`  
`hwloc_topology * topology, hwloc_const_cpuset_t cpuset,`  
`hwloc_nodeset_t nodeset ) [static]`

Convert a CPU set into a NUMA node set without handling non-NUMA cases.

This is the strict variant of [hwloc\\_cpuset\\_to\\_nodeset](#). It does not fix non-NUMA cases. If the topology contains some NUMA nodes, behave exactly the same. However, if the topology contains no NUMA nodes, return an empty `nodeset`.

## 13.25 The bitmap API

### Defines

- #define `hwloc_bitmap_foreach_begin(id, bitmap)`  
*Loop macro iterating on bitmap `bitmap`.*
- #define `hwloc_bitmap_foreach_end()`  
*End of loop. Needs a terminating `';`.*

### Typedefs

- typedef struct hwloc\_bitmap\_s \* `hwloc_bitmap_t`  
*Set of bits represented as an opaque pointer to an internal bitmap.*
- typedef struct hwloc\_bitmap\_s \* `hwloc_const_bitmap_t`

### Functions

- HWLOC\_DECLSPEC `hwloc_bitmap_t hwloc_bitmap_alloc (void) __hwloc_attribute_malloc`  
*Allocate a new empty bitmap.*
- HWLOC\_DECLSPEC `hwloc_bitmap_t hwloc_bitmap_alloc_full (void) __hwloc_attribute_malloc`  
*Allocate a new full bitmap.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_free (hwloc_bitmap_t bitmap)`  
*Free bitmap `bitmap`.*
- HWLOC\_DECLSPEC `hwloc_bitmap_t hwloc_bitmap_dup (hwloc_const_bitmap_t bitmap) __hwloc_attribute_malloc`  
*Duplicate bitmap `bitmap` by allocating a new bitmap and copying `bitmap` contents.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_copy (hwloc_bitmap_t dst, hwloc_const_bitmap_t src)`  
*Copy the contents of bitmap `src` into the already allocated bitmap `dst`.*
- HWLOC\_DECLSPEC int `hwloc_bitmap_snprintf (char *__hwloc_restrict buf, size_t buflen, hwloc_const_bitmap_t bitmap)`

*Stringify a bitmap.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_asprintf` (char \*\*strp, hwloc\_const\_bitmap\_t bitmap)

*Stringify a bitmap into a newly allocated string.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_sscanf` (hwloc\_bitmap\_t bitmap, const char \*\_\_hwloc\_restrict string)

*Parse a bitmap string and stores it in bitmap `bitmap`.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_taskset_snprintf` (char \*\_\_hwloc\_restrict buf, size\_t buflen, hwloc\_const\_bitmap\_t bitmap)

*Stringify a bitmap in the taskset-specific format.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_taskset_asprintf` (char \*\*strp, hwloc\_const\_bitmap\_t bitmap)

*Stringify a bitmap into a newly allocated taskset-specific string.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_taskset_sscanf` (hwloc\_bitmap\_t bitmap, const char \*\_\_hwloc\_restrict string)

*Parse a taskset-specific bitmap string and stores it in bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_zero` (hwloc\_bitmap\_t bitmap)

*Empty the bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_fill` (hwloc\_bitmap\_t bitmap)

*Fill bitmap `bitmap` with all possible indexes (even if those objects don't exist or are otherwise unavailable).*

- HWLOC\_DECLSPEC void `hwloc_bitmap_only` (hwloc\_bitmap\_t bitmap, unsigned id)

*Empty the bitmap `bitmap` and add bit `id`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_allbut` (hwloc\_bitmap\_t bitmap, unsigned id)

*Fill the bitmap and clear the index `id`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_from_ulong` (hwloc\_bitmap\_t bitmap, unsigned long mask)

*Setup bitmap `bitmap` from unsigned long `mask`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_from_ith_ulong` (hwloc\_bitmap\_t bitmap, unsigned i, unsigned long mask)



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

- HWLOC\_DECLSPEC void `hwloc_bitmap_set` (`hwloc_bitmap_t` bitmap, unsigned id)

*Add index `id` in bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_set_range` (`hwloc_bitmap_t` bitmap, unsigned begin, unsigned end)

*Add indexes from `begin` to `end` in bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_set_ith_ulong` (`hwloc_bitmap_t` bitmap, unsigned `i`, unsigned long mask)

*Replace `i`-th subset of bitmap `bitmap` with unsigned long mask.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_clr` (`hwloc_bitmap_t` bitmap, unsigned id)

*Remove index `id` from bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_clr_range` (`hwloc_bitmap_t` bitmap, unsigned begin, unsigned end)

*Remove index from `begin` to `end` in bitmap `bitmap`.*

- HWLOC\_DECLSPEC void `hwloc_bitmap_singlify` (`hwloc_bitmap_t` bitmap)

*Keep a single index among those set in bitmap `bitmap`.*

- HWLOC\_DECLSPEC unsigned long `hwloc_bitmap_to_ulong` (`hwloc_const_bitmap_t` bitmap) `__hwloc_attribute_pure`

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

- HWLOC\_DECLSPEC unsigned long `hwloc_bitmap_to_ith_ulong` (`hwloc_const_bitmap_t` bitmap, unsigned `i`) `__hwloc_attribute_pure`

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

- HWLOC\_DECLSPEC int `hwloc_bitmap_isset` (`hwloc_const_bitmap_t` bitmap, unsigned id) `__hwloc_attribute_pure`

*Test whether index `id` is part of bitmap `bitmap`.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_iszero` (`hwloc_const_bitmap_t` bitmap) `__hwloc_attribute_pure`

*Test whether bitmap `bitmap` is empty.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_isfull` (`hwloc_const_bitmap_t` bitmap) `__hwloc_attribute_pure`

*Test whether bitmap `bitmap` is completely full.*

- HWLOC\_DECLSPEC int `hwloc_bitmap_first` (`hwloc_const_bitmap_t` `bitmap`)  
\_\_hwloc\_attribute\_pure  
*Compute the first index (least significant bit) in bitmap `bitmap`.*
- HWLOC\_DECLSPEC int `hwloc_bitmap_next` (`hwloc_const_bitmap_t` `bitmap`,  
unsigned `prev`) \_\_hwloc\_attribute\_pure  
*Compute the next index in bitmap `bitmap` which is after index `prev`.*
- HWLOC\_DECLSPEC int `hwloc_bitmap_last` (`hwloc_const_bitmap_t` `bitmap`)  
\_\_hwloc\_attribute\_pure  
*Compute the last index (most significant bit) in bitmap `bitmap`.*
- HWLOC\_DECLSPEC int `hwloc_bitmap_weight` (`hwloc_const_bitmap_t`  
`bitmap`) \_\_hwloc\_attribute\_pure  
*Compute the "weight" of bitmap `bitmap` (i.e., number of indexes that are in the  
bitmap).*
- HWLOC\_DECLSPEC void `hwloc_bitmap_or` (`hwloc_bitmap_t` `res`, `hwloc_-`  
`const_bitmap_t` `bitmap1`, `hwloc_const_bitmap_t` `bitmap2`)  
*Or bitmaps `bitmap1` and `bitmap2` and store the result in bitmap `res`.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_and` (`hwloc_bitmap_t` `res`, `hwloc_-`  
`const_bitmap_t` `bitmap1`, `hwloc_const_bitmap_t` `bitmap2`)  
*And bitmaps `bitmap1` and `bitmap2` and store the result in bitmap `res`.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_andnot` (`hwloc_bitmap_t` `res`,  
`hwloc_const_bitmap_t` `bitmap1`, `hwloc_const_bitmap_t` `bitmap2`)  
*And bitmap `bitmap1` and the negation of `bitmap2` and store the result in bitmap  
`res`.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_xor` (`hwloc_bitmap_t` `res`, `hwloc_-`  
`const_bitmap_t` `bitmap1`, `hwloc_const_bitmap_t` `bitmap2`)  
*Xor bitmaps `bitmap1` and `bitmap2` and store the result in bitmap `res`.*
- HWLOC\_DECLSPEC void `hwloc_bitmap_not` (`hwloc_bitmap_t` `res`, `hwloc_-`  
`const_bitmap_t` `bitmap`)  
*Negate bitmap `bitmap` and store the result in bitmap `res`.*
- HWLOC\_DECLSPEC int `hwloc_bitmap_intersects` (`hwloc_const_bitmap_-`  
`t` `bitmap1`, `hwloc_const_bitmap_t` `bitmap2`) \_\_hwloc\_attribute\_pure  
*Test whether bitmaps `bitmap1` and `bitmap2` intersect.*

- `HWLOC_DECLSPEC int hwloc_bitmap_isincluded (hwloc_const_bitmap_t sub_bitmap, hwloc_const_bitmap_t super_bitmap) __hwloc_attribute_pure`  
*Test whether bitmap `sub_bitmap` is part of bitmap `super_bitmap`.*
- `HWLOC_DECLSPEC int hwloc_bitmap_isequal (hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2) __hwloc_attribute_pure`  
*Test whether bitmap `bitmap1` is equal to bitmap `bitmap2`.*
- `HWLOC_DECLSPEC int hwloc_bitmap_compare_first (hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2) __hwloc_attribute_pure`  
*Compare bitmaps `bitmap1` and `bitmap2` using their lowest index.*
- `HWLOC_DECLSPEC int hwloc_bitmap_compare (hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2) __hwloc_attribute_pure`  
*Compare bitmaps `bitmap1` and `bitmap2` using their highest index.*

### 13.25.1 Detailed Description

For use in hwloc itself, a `hwloc_bitmap_t` usually represents a set of objects, typically logical processors or memory nodes, indexed by OS physical number.

A bitmap may be infinite.

### 13.25.2 Define Documentation

#### 13.25.2.1 `#define hwloc_bitmap_foreach_begin( id, bitmap )`

Loop macro iterating on bitmap `bitmap`.

`index` is the loop variable; it should be an unsigned int. The first iteration will set `index` to the lowest index in the bitmap. Successive iterations will iterate through, in order, all remaining indexes that in the bitmap. To be specific: each iteration will return a value for `index` such that `hwloc_bitmap_isset(bitmap, index)` is true.

The assert prevents the loop from being infinite if the bitmap is infinite.

#### 13.25.2.2 `#define hwloc_bitmap_foreach_end( )`

End of loop. Needs a terminating `;`.

See also

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

### 13.25.3 Typedef Documentation

#### 13.25.3.1 `typedef struct hwloc_bitmap_s* hwloc_bitmap_t`

Set of bits represented as an opaque pointer to an internal bitmap.

#### 13.25.3.2 `typedef struct hwloc_bitmap_s* hwloc_const_bitmap_t`

### 13.25.4 Function Documentation

#### 13.25.4.1 `HWLOC_DECLSPEC void hwloc_bitmap_allbut ( hwloc_bitmap_t bitmap, unsigned id )`

Fill the `bitmap` and clear the index `id`.

#### 13.25.4.2 `HWLOC_DECLSPEC hwloc_bitmap_t hwloc_bitmap_alloc ( void )`

Allocate a new empty bitmap.

##### Returns

A valid bitmap or NULL.

The bitmap should be freed by a corresponding call to [hwloc\\_bitmap\\_free\(\)](#).

#### 13.25.4.3 `HWLOC_DECLSPEC hwloc_bitmap_t hwloc_bitmap_alloc_full ( void )`

Allocate a new full bitmap.

#### 13.25.4.4 `HWLOC_DECLSPEC void hwloc_bitmap_and ( hwloc_bitmap_t res, hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2 )`

And bitmaps `bitmap1` and `bitmap2` and store the result in bitmap `res`.

#### 13.25.4.5 `HWLOC_DECLSPEC void hwloc_bitmap_andnot ( hwloc_bitmap_t res, hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2 )`

And bitmap `bitmap1` and the negation of `bitmap2` and store the result in bitmap `res`.

**13.25.4.6 HWLOC\_DECLSPEC int hwloc\_bitmap\_asprintf ( char \*\* *strp*,  
hwloc\_const\_bitmap\_t *bitmap* )**

Stringify a bitmap into a newly allocated string.

**Returns**

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

**13.25.4.7 HWLOC\_DECLSPEC void hwloc\_bitmap\_clr ( hwloc\_bitmap\_t  
*bitmap*, unsigned *id* )**

Remove index `id` from bitmap `bitmap`.

**13.25.4.8 HWLOC\_DECLSPEC void hwloc\_bitmap\_clr\_range (  
hwloc\_bitmap\_t *bitmap*, unsigned *begin*, unsigned *end* )**

Remove index from `begin` to `end` in bitmap `bitmap`.

**13.25.4.9 HWLOC\_DECLSPEC int hwloc\_bitmap\_compare (  
hwloc\_const\_bitmap\_t *bitmap1*, hwloc\_const\_bitmap\_t *bitmap2* )**

Compare bitmaps `bitmap1` and `bitmap2` using their highest index.

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

**13.25.4.10 HWLOC\_DECLSPEC int hwloc\_bitmap\_compare\_first (  
hwloc\_const\_bitmap\_t *bitmap1*, hwloc\_const\_bitmap\_t *bitmap2* )**

Compare bitmaps `bitmap1` and `bitmap2` using their lowest index.

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

**13.25.4.11 HWLOC\_DECLSPEC void hwloc\_bitmap\_copy ( hwloc\_bitmap\_t  
*dst*, hwloc\_const\_bitmap\_t *src* )**

Copy the contents of bitmap `src` into the already allocated bitmap `dst`.

**13.25.4.12 HWLOC\_DECLSPEC hwloc\_bitmap\_t hwloc\_bitmap\_dup ( hwloc\_const\_bitmap\_t *bitmap* )**

Duplicate bitmap *bitmap* by allocating a new bitmap and copying *bitmap* contents.

**13.25.4.13 HWLOC\_DECLSPEC void hwloc\_bitmap\_fill ( hwloc\_bitmap\_t *bitmap* )**

Fill bitmap *bitmap* with all possible indexes (even if those objects don't exist or are otherwise unavailable).

**13.25.4.14 HWLOC\_DECLSPEC int hwloc\_bitmap\_first ( hwloc\_const\_bitmap\_t *bitmap* )**

Compute the first index (least significant bit) in bitmap *bitmap*.

**Returns**

-1 if no index is set.

**13.25.4.15 HWLOC\_DECLSPEC void hwloc\_bitmap\_free ( hwloc\_bitmap\_t *bitmap* )**

Free bitmap *bitmap*.

**13.25.4.16 HWLOC\_DECLSPEC void hwloc\_bitmap\_from\_ith\_ulong ( hwloc\_bitmap\_t *bitmap*, unsigned *i*, unsigned long *mask* )**

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

**13.25.4.17 HWLOC\_DECLSPEC void hwloc\_bitmap\_from\_ulong ( hwloc\_bitmap\_t *bitmap*, unsigned long *mask* )**

Setup bitmap *bitmap* from unsigned long *mask*.

**13.25.4.18 HWLOC\_DECLSPEC int hwloc\_bitmap\_intersects ( hwloc\_const\_bitmap\_t *bitmap1*, hwloc\_const\_bitmap\_t *bitmap2* )**

Test whether bitmaps *bitmap1* and *bitmap2* intersects.

**13.25.4.19** `HWLOC_DECLSPEC int hwloc_bitmap_isequal (`  
    `hwloc_const_bitmap_t bitmap1, hwloc_const_bitmap_t bitmap2 )`

Test whether bitmap `bitmap1` is equal to bitmap `bitmap2`.

**13.25.4.20** `HWLOC_DECLSPEC int hwloc_bitmap_isfull (`  
    `hwloc_const_bitmap_t bitmap )`

Test whether bitmap `bitmap` is completely full.

**13.25.4.21** `HWLOC_DECLSPEC int hwloc_bitmap_isincluded (`  
    `hwloc_const_bitmap_t sub_bitmap, hwloc_const_bitmap_t`  
    `super_bitmap )`

Test whether bitmap `sub_bitmap` is part of bitmap `super_bitmap`.

**13.25.4.22** `HWLOC_DECLSPEC int hwloc_bitmap_isset (`  
    `hwloc_const_bitmap_t bitmap, unsigned id )`

Test whether index `id` is part of bitmap `bitmap`.

**13.25.4.23** `HWLOC_DECLSPEC int hwloc_bitmap_iszero (`  
    `hwloc_const_bitmap_t bitmap )`

Test whether bitmap `bitmap` is empty.

**13.25.4.24** `HWLOC_DECLSPEC int hwloc_bitmap_last (`  
    `hwloc_const_bitmap_t bitmap )`

Compute the last index (most significant bit) in bitmap `bitmap`.

#### Returns

-1 if no index is bitmap, or if the index bitmap is infinite.

**13.25.4.25** `HWLOC_DECLSPEC int hwloc_bitmap_next (`  
    `hwloc_const_bitmap_t bitmap, unsigned prev )`

Compute the next index in bitmap `bitmap` which is after index `prev`.

**Returns**

-1 if no index with higher index is bitmap.

**13.25.4.26** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_not ( hwloc\_bitmap\_t *res*, hwloc\_const\_bitmap\_t *bitmap* )

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

**13.25.4.27** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_only ( hwloc\_bitmap\_t *bitmap*, unsigned *id* )

Empty the bitmap *bitmap* and add bit *id*.

**13.25.4.28** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_or ( hwloc\_bitmap\_t *res*, hwloc\_const\_bitmap\_t *bitmap1*, hwloc\_const\_bitmap\_t *bitmap2* )

Or bitmaps *bitmap1* and *bitmap2* and store the result in bitmap *res*.

**13.25.4.29** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_set ( hwloc\_bitmap\_t *bitmap*, unsigned *id* )

Add index *id* in bitmap *bitmap*.

**13.25.4.30** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_set\_ith\_ulong ( hwloc\_bitmap\_t *bitmap*, unsigned *i*, unsigned long *mask* )

Replace *i* -th subset of bitmap *bitmap* with unsigned long *mask*.

**13.25.4.31** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_set\_range ( hwloc\_bitmap\_t *bitmap*, unsigned *begin*, unsigned *end* )

Add indexes from *begin* to *end* in bitmap *bitmap*.

**13.25.4.32** **HWLOC\_DECLSPEC** void hwloc\_bitmap\_singlify ( hwloc\_bitmap\_t *bitmap* )

Keep a single index among those set in bitmap *bitmap*.

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



**13.25.4.33** `HWLOC_DECLSPEC int hwloc_bitmap_snprintf ( char  
*__hwloc_restrict buf, size_t buflen, hwloc_const_bitmap_t bitmap  
)`

Stringify a bitmap.

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`).

**13.25.4.34** `HWLOC_DECLSPEC int hwloc_bitmap_sscanf ( hwloc_bitmap_t  
bitmap, const char *__hwloc_restrict string )`

Parse a bitmap string and stores it in bitmap `bitmap`.

Must start and end with a digit.

**13.25.4.35** `HWLOC_DECLSPEC int hwloc_bitmap_taskset_asprintf ( char **  
strp, hwloc_const_bitmap_t bitmap )`

Stringify a bitmap into a newly allocated taskset-specific string.

**13.25.4.36** `HWLOC_DECLSPEC int hwloc_bitmap_taskset_snprintf ( char  
*__hwloc_restrict buf, size_t buflen, hwloc_const_bitmap_t bitmap  
)`

Stringify a bitmap in the taskset-specific format.

The taskset command manipulates bitmap strings that contain a single (possible very long) hexadecimal number starting with `0x`.

**13.25.4.37** `HWLOC_DECLSPEC int hwloc_bitmap_taskset_sscanf (  
hwloc_bitmap_t bitmap, const char *__hwloc_restrict string )`

Parse a taskset-specific bitmap string and stores it in bitmap `bitmap`.

**13.25.4.38** `HWLOC_DECLSPEC unsigned long hwloc_bitmap_to_ith_ulong (  
hwloc_const_bitmap_t bitmap, unsigned i )`

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

#### 13.25.4.39 HWLOC\_DECLSPEC unsigned long hwloc\_bitmap\_to\_ulong ( hwloc\_const\_bitmap\_t *bitmap* )

Convert the beginning part of bitmap *bitmap* into unsigned long *mask*.

#### 13.25.4.40 HWLOC\_DECLSPEC int hwloc\_bitmap\_weight ( hwloc\_const\_bitmap\_t *bitmap* )

Compute the "weight" of bitmap *bitmap* (i.e., number of indexes that are in the bitmap).

##### Returns

the number of indexes that are in the bitmap.

#### 13.25.4.41 HWLOC\_DECLSPEC void hwloc\_bitmap\_xor ( hwloc\_bitmap\_t *res*, hwloc\_const\_bitmap\_t *bitmap1*, hwloc\_const\_bitmap\_t *bitmap2* )

Xor bitmaps *bitmap1* and *bitmap2* and store the result in bitmap *res*.

#### 13.25.4.42 HWLOC\_DECLSPEC void hwloc\_bitmap\_zero ( hwloc\_bitmap\_t *bitmap* )

Empty the bitmap *bitmap*.

## 13.26 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.*

### 13.26.1 Function Documentation

**13.26.1.1** `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`

**13.26.1.2** `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`

## 13.27 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`.*

### 13.27.1 Detailed Description

This includes helpers for manipulating linux kernel cpumap files, and hwloc equivalents of the Linux sched\_setaffinity and sched\_getaffinity system calls.

### 13.27.2 Function Documentation

#### 13.27.2.1 HWLOC\_DECLSPEC int hwloc\_linux\_get\_tid\_cpupind ( 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.

#### 13.27.2.2 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.

#### 13.27.2.3 HWLOC\_DECLSPEC int hwloc\_linux\_set\_tid\_cpupind ( 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.

## 13.28 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_nodeset_to_linux_libnuma_ulongs` (`hwloc_topology_t` topology, `hwloc_const_nodeset_t` nodeset, unsigned long \*mask, unsigned long \*maxnode)

*Convert hwloc NUMA node set nodeset 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.*

- static \_\_hwloc\_inline int `hwloc_nodeset_from_linux_libnuma_ulongs` (`hwloc_topology_t` topology, `hwloc_nodeset_t` nodeset, const unsigned long \*mask, unsigned long maxnode)

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

### 13.28.1 Function Documentation

**13.28.1.1** 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).

**13.28.1.2** 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.

**13.28.1.3** `static __hwloc_inline int hwloc_nodeset_from_linux_libnuma_ulongs  
( hwloc_topology_t topology, hwloc_nodeset_t nodeset, const  
unsigned long * mask, unsigned long maxnode ) [static]`

Convert the array of unsigned long `mask` into hwloc NUMA node 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).

**13.28.1.4** `static __hwloc_inline int hwloc_nodeset_to_linux_libnuma_ulongs  
( hwloc_topology_t topology, hwloc_const_nodeset_t nodeset,  
unsigned long * mask, unsigned long * maxnode ) [static]`

Convert hwloc NUMA node set `nodeset` 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.

## 13.29 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 struct bitmask * __hwloc_attribute_malloc hwloc_nodeset_to_linux_libnuma_bitmask (hwloc_topology_t topology, hwloc_const_nodeset_t nodeset)`

*Convert hwloc NUMA node set `nodeset` 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`.*

- `static __hwloc_inline int hwloc_nodeset_from_linux_libnuma_bitmask (hwloc_topology_t topology, hwloc_nodeset_t nodeset, const struct bitmask *bitmask)`

*Convert libnuma bitmask `bitmask` into hwloc NUMA node set `nodeset`.*

### 13.29.1 Function Documentation

**13.29.1.1** `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.

**13.29.1.2** `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`.

**13.29.1.3** `static __hwloc_inline int hwloc_nodeset_from_linux_libnuma_bitmask ( hwloc_topology_t topology, hwloc_nodeset_t nodeset, const struct bitmask * bitmask ) [static]`

Convert libnuma bitmask `bitmask` into hwloc NUMA node set `nodeset`.

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

**13.29.1.4** `static __hwloc_inline struct bitmask* __hwloc_attribute_malloc  
hwloc_nodeset_to_linux_libnuma_bitmask ( hwloc_topology_t  
topology, hwloc_const_nodeset_t nodeset ) [static, read]`

Convert hwloc NUMA node set `nodeset` 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.

## 13.30 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_nodeset_to_linux_libnuma_nodemask (hwloc_topology_t topology, hwloc_const_nodeset_t nodeset, nodemask_t *nodemask)`

*Convert hwloc NUMA node set `nodeset` 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`.*

- `static __hwloc_inline int hwloc_nodeset_from_linux_libnuma_nodemask (hwloc_topology_t topology, hwloc_nodeset_t nodeset, const nodemask_t *nodemask)`

*Convert libnuma nodemask `nodemask` into hwloc NUMA node set `nodeset`.*



### 13.30.1 Function Documentation

**13.30.1.1** `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.

**13.30.1.2** `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.

**13.30.1.3** `static __hwloc_inline int hwloc_nodeset_from_linux_libnuma_nodemask ( hwloc_topology_t topology, hwloc_nodeset_t nodeset, const nodemask_t * nodemask ) [static]`

Convert libnuma nodemask `nodemask` into hwloc NUMA node set `nodeset`.

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

**13.30.1.4** `static __hwloc_inline int hwloc_nodeset_to_linux_libnuma_nodemask ( hwloc_topology_t topology, hwloc_const_nodeset_t nodeset, nodemask_t * nodemask ) [static]`

Convert hwloc NUMA node set `nodeset` into libnuma nodemask `nodemask`.

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

## 13.31 CUDA Driver API Specific Functions

### Functions

- `static __hwloc_inline int hwloc_cuda_get_device_cpuset (hwloc_topology_t topology __hwloc_attribute_unused, CUdevice cuddevice, hwloc_cpuset_t set)`

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

### 13.31.1 Function Documentation

**13.31.1.1** `static __hwloc_inline int hwloc_cuda_get_device_cpuset ( hwloc_topology_t topology __hwloc_attribute_unused, CUdevice cudevice, hwloc_cpuset_t set ) [static]`

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

For the given CUDA Driver API device `cudevice`, 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.

## 13.32 CUDA Runtime API Specific Functions

### Functions

- `static __hwloc_inline int hwloc_cudart_get_device_cpuset (hwloc_topology_t topology __hwloc_attribute_unused, int device, hwloc_cpuset_t set)`

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

### 13.32.1 Function Documentation

**13.32.1.1** `static __hwloc_inline int hwloc_cudart_get_device_cpuset ( hwloc_topology_t topology __hwloc_attribute_unused, int device, hwloc_cpuset_t set ) [static]`

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

For the given CUDA Runtime API device `cudevice`, 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.

## 13.33 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.*

### 13.33.1 Function Documentation

**13.33.1.1** 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.

## 13.34 Myrinet Express-Specific Functions

### Functions

- static \_\_hwloc\_inline int `hwloc_mx_board_get_device_cpuset` (`hwloc_topology_t` topology, unsigned id, `hwloc_cpuset_t` set)

*Get the CPU set of logical processors that are physically close the MX board id.*

- static \_\_hwloc\_inline int `hwloc_mx_endpoint_get_device_cpuset` (`hwloc_topology_t` topology, `mx_endpoint_t` endpoint, `hwloc_cpuset_t` set)

*Get the CPU set of logical processors that are physically close to endpoint endpoint.*

### 13.34.1 Function Documentation

**13.34.1.1** `static __hwloc_inline int hwloc_mx_board_get_device_cpuset (`  
    `hwloc_topology_t topology, unsigned id, hwloc_cpuset_t set )`  
    `[static]`

Get the CPU set of logical processors that are physically close the MX board `id`.

For the given Myrinet Express board index `id`, read the OS-provided NUMA node and return the corresponding CPU set.

**13.34.1.2** `static __hwloc_inline int hwloc_mx_endpoint_get_device_cpuset`  
    `( hwloc_topology_t topology, mx_endpoint_t endpoint,`  
    `hwloc_cpuset_t set ) [static]`

Get the CPU set of logical processors that are physically close to endpoint `endpoint`.

For the given Myrinet Express endpoint `endpoint`, read the OS-provided NUMA node and return the corresponding CPU set.

## Chapter 14

# Data Structure Documentation

### 14.1 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.*
- unsigned [linesize](#)  
*Cache-line size in bytes.*

#### 14.1.1 Detailed Description

Cache-specific Object Attributes.

### 14.1.2 Field Documentation

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

Depth of cache.

#### 14.1.2.2 unsigned hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s::linesize

Cache-line size in bytes.

#### 14.1.2.3 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

## 14.2 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.*

### 14.2.1 Detailed Description

Group-specific Object Attributes.

### 14.2.2 Field Documentation

#### 14.2.2.1 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

## 14.3 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, may be NULL if no attribute value was found.*
- 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_nodeset_t nodeset`  
*NUMA nodes covered by this object or containing this object.*



- [hwloc\\_nodeset\\_t complete\\_nodeset](#)  
*The complete NUMA node set of this object.*
- [hwloc\\_nodeset\\_t allowed\\_nodeset](#)  
*The set of allowed NUMA memory nodes.*
- struct [hwloc\\_obj\\_info\\_s](#) \* [infos](#)  
*Array of stringified info type=name.*
- unsigned [infos\\_count](#)  
*Size of infos array.*

### 14.3.1 Detailed Description

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

### 14.3.2 Field Documentation

#### 14.3.2.1 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_bitmap_dup` must be used instead.

#### 14.3.2.2 hwloc\_nodeset\_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.

If there are no NUMA nodes in the machine, all the memory is close to this object, so `allowed_nodeset` is full.

**Note**

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

**14.3.2.3 unsigned hwloc\_obj::arity**

Number of children.

**14.3.2.4 union hwloc\_obj\_attr\_u\* hwloc\_obj::attr**

Object type-specific Attributes, may be `NULL` if no attribute value was found.

**14.3.2.5 struct hwloc\_obj\*\* hwloc\_obj::children**

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

**14.3.2.6 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_bitmap_dup` must be used instead.

**14.3.2.7 hwloc\_node\_t hwloc\_obj::complete\_node**

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.

If there are no NUMA nodes in the machine, all the memory is close to this object, so `complete_node` is full.

**Note**

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

**14.3.2.8 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\_bitmap\_dup must be used instead.

**14.3.2.9 unsigned hwloc\_obj::depth**

Vertical index in the hierarchy.

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

First child.

**14.3.2.11 struct hwloc\_obj\_info\_s\* hwloc\_obj::infos**

Array of stringified info type=name.

**14.3.2.12 unsigned hwloc\_obj::infos\_count**

Size of infos array.

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

Last child.

**14.3.2.14 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.

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

Memory attributes.

**14.3.2.16 char\* hwloc\_obj::name**

Object description if any.

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

Next object of same type.

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

Next object below the same parent.

**14.3.2.19 hwloc\_nodese\_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).

In the end, these nodes are those that are close to the current object.

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

If there are no NUMA nodes in the machine, all the memory is close to this object, so `nodeset` is full.

**Note**

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

**14.3.2.20 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_bitmap_dup` must be used instead.

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

OS-provided physical index number.

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

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

**14.3.2.23 struct hwloc\_obj\* hwloc\_obj::parent**

Parent, NULL if root (system object).

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

Previous object of same type.

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

Previous object below the same parent.

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

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

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

Type of object.

#### 14.3.2.28 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

## 14.4 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.*

### Data Fields

- struct [hwloc\\_obj\\_attr\\_u::hwloc\\_cache\\_attr\\_s](#) cache  
*Cache-specific Object Attributes.*
- struct [hwloc\\_obj\\_attr\\_u::hwloc\\_group\\_attr\\_s](#) group  
*Group-specific Object Attributes.*

#### 14.4.1 Detailed Description

Object type-specific Attributes.

#### 14.4.2 Field Documentation

##### 14.4.2.1 struct hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s hwloc\_obj\_attr\_u::cache

Cache-specific Object Attributes.

#### 14.4.2.2 struct hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s hwloc\_obj\_attr\_u::group

Group-specific Object Attributes.

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

- hwloc.h

## 14.5 hwloc\_obj\_info\_s Struct Reference

Object info.

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

### Data Fields

- char \* [name](#)  
*Info name.*
- char \* [value](#)  
*Info value.*

### 14.5.1 Detailed Description

Object info.

### 14.5.2 Field Documentation

#### 14.5.2.1 char\* hwloc\_obj\_info\_s::name

Info name.

#### 14.5.2.2 char\* hwloc\_obj\_info\_s::value

Info value.

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

- hwloc.h

## 14.6 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.*

### 14.6.1 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.

### 14.6.2 Field Documentation

#### 14.6.2.1 `uint64_t hwloc_obj_memory_s::hwloc_obj_memory_page_type_s::count`

Number of pages of this size.

#### 14.6.2.2 `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`

## 14.7 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.*

### 14.7.1 Detailed Description

Object memory.

### 14.7.2 Field Documentation

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

Local memory (in bytes).

#### 14.7.2.2 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.

#### 14.7.2.3 unsigned hwloc\_obj\_memory\_s::page\_types\_len

Size of array `page_types`.

#### 14.7.2.4 `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`

## 14.8 `hwloc_topology_cpupbind_support` Struct Reference

Flags describing actual PU 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](#)

#### 14.8.1 Detailed Description

Flags describing actual PU binding support for this topology.

#### 14.8.2 Field Documentation

##### 14.8.2.1 `unsigned char hwloc_topology_cpupbind_support::get_proc_cpupbind`

Getting the binding of a whole given process is supported.

##### 14.8.2.2 `unsigned char hwloc_topology_cpupbind_support::get_thisproc_cpupbind`

Getting the binding of the whole current process is supported.

**14.8.2.3 unsigned char hwloc\_topology\_cpubind\_support::get\_thisthread\_ - cpubind**

Getting the binding of the current thread only is supported.

**14.8.2.4 unsigned char hwloc\_topology\_cpubind\_support::get\_thread\_cpubind**

Getting the binding of a given thread only is supported.

**14.8.2.5 unsigned char hwloc\_topology\_cpubind\_support::set\_proc\_cpubind**

Binding a whole given process is supported.

**14.8.2.6 unsigned char hwloc\_topology\_cpubind\_support::set\_thisproc\_ - cpubind**

Binding the whole current process is supported.

**14.8.2.7 unsigned char hwloc\_topology\_cpubind\_support::set\_thisthread\_ - cpubind**

Binding the current thread only is supported.

**14.8.2.8 unsigned char hwloc\_topology\_cpubind\_support::set\_thread\_cpubind**

Binding a given thread only is supported.

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

- hwloc.h

## 14.9 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.*

### 14.9.1 Detailed Description

Flags describing actual discovery support for this topology.

### 14.9.2 Field Documentation

#### 14.9.2.1 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

## 14.10 hwloc\_topology\_membind\_support Struct Reference

Flags describing actual memory binding support for this topology.

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

## Data Fields

- unsigned char [set\\_thisproc\\_membind](#)
- unsigned char [get\\_thisproc\\_membind](#)
- unsigned char [set\\_proc\\_membind](#)
- unsigned char [get\\_proc\\_membind](#)
- unsigned char [set\\_thisthread\\_membind](#)
- unsigned char [get\\_thisthread\\_membind](#)
- unsigned char [set\\_area\\_membind](#)
- unsigned char [get\\_area\\_membind](#)
- unsigned char [alloc\\_membind](#)
- unsigned char [firsttouch\\_membind](#)
- unsigned char [bind\\_membind](#)
- unsigned char [interleave\\_membind](#)

- unsigned char [replicate\\_membind](#)
- unsigned char [nexttouch\\_membind](#)
- unsigned char [migrate\\_membind](#)

### 14.10.1 Detailed Description

Flags describing actual memory binding support for this topology.

### 14.10.2 Field Documentation

#### 14.10.2.1 unsigned char hwloc\_topology\_membind\_support::alloc\_membind

Allocating a bound memory area is supported.

#### 14.10.2.2 unsigned char hwloc\_topology\_membind\_support::bind\_membind

Bind policy is supported.

#### 14.10.2.3 unsigned char hwloc\_topology\_membind\_support::firsttouch\_membind

First-touch policy is supported.

#### 14.10.2.4 unsigned char hwloc\_topology\_membind\_support::get\_area\_membind

Getting the binding of a given memory area is supported.

#### 14.10.2.5 unsigned char hwloc\_topology\_membind\_support::get\_proc\_membind

Getting the binding of a whole given process is supported.

#### 14.10.2.6 unsigned char hwloc\_topology\_membind\_support::get\_thisproc\_membind

Getting the binding of the whole current process is supported.

**14.10.2.7 unsigned char hwloc\_topology\_membind\_support::get\_thisthread\_membind**

Getting the binding of the current thread only is supported.

**14.10.2.8 unsigned char hwloc\_topology\_membind\_support::interleave\_membind**

Interleave policy is supported.

**14.10.2.9 unsigned char hwloc\_topology\_membind\_support::migrate\_membind**

Migration flags is supported.

**14.10.2.10 unsigned char hwloc\_topology\_membind\_support::nexttouch\_membind**

Next-touch migration policy is supported.

**14.10.2.11 unsigned char hwloc\_topology\_membind\_support::replicate\_membind**

Replication policy is supported.

**14.10.2.12 unsigned char hwloc\_topology\_membind\_support::set\_area\_membind**

Binding a given memory area is supported.

**14.10.2.13 unsigned char hwloc\_topology\_membind\_support::set\_proc\_membind**

Binding a whole given process is supported.

**14.10.2.14 unsigned char hwloc\_topology\_membind\_support::set\_thisproc\_membind**

Binding the whole current process is supported.

#### 14.10.2.15 unsigned char hwloc\_topology\_membind\_support::set\_thisthread\_membind

Binding the current thread only is supported.

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

- hwloc.h

## 14.11 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](#)
- struct [hwloc\\_topology\\_membind\\_support](#) \* [membind](#)

#### 14.11.1 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.

#### 14.11.2 Field Documentation

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

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

**14.11.2.3** struct [hwloc\\_topology\\_membind\\_support](#)\*  
[hwloc\\_topology\\_support::membind](#)

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

- hwloc.h

# Index

- Advanced Traversal Helpers, [87](#)
- alloc\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- allowed\_cpuset
  - hwloc\_obj, [121](#)
- allowed\_nodeset
  - hwloc\_obj, [121](#)
- API version, [47](#)
- arity
  - hwloc\_obj, [122](#)
- attr
  - hwloc\_obj, [122](#)
- Basic Traversal Helpers, [79](#)
- bind\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- Binding Helpers, [89](#)
- cache
  - hwloc\_obj\_attr\_u, [126](#)
- Cache-specific Finding Helpers, [86](#)
- children
  - hwloc\_obj, [122](#)
- complete\_cpuset
  - hwloc\_obj, [122](#)
- complete\_nodeset
  - hwloc\_obj, [122](#)
- Configure Topology Detection, [53](#)
- Conversion between cpuset and nodeset, [93](#)
- count
  - hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s, [128](#)
- CPU binding, [66](#)
- cpubind
  - hwloc\_topology\_support, [135](#)
- cpuset
  - hwloc\_obj, [123](#)
- Cpuset Helpers, [90](#)
- Create and Destroy Topologies, [52](#)
- CUDA Driver API Specific Functions, [113](#)
- CUDA Runtime API Specific Functions, [114](#)
- depth
  - hwloc\_obj, [123](#)
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [118](#)
  - hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s, [118](#)
- discovery
  - hwloc\_topology\_support, [135](#)
- Finding a set of similar Objects covering at least a CPU set, [85](#)
- Finding a single Object covering at least CPU set, [84](#)
- Finding Objects Inside a CPU set, [82](#)
- first\_child
  - hwloc\_obj, [123](#)
- firsttouch\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- Get some Topology Information, [60](#)
- get\_area\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- get\_proc\_cpubind
  - hwloc\_topology\_cpubind\_support, [130](#)



- get\_proc\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- get\_thisproc\_cpubind
  - hwloc\_topology\_cpubind\_support, [130](#)
- get\_thisproc\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- get\_thisthread\_cpubind
  - hwloc\_topology\_cpubind\_support, [130](#)
- get\_thisthread\_membind
  - hwloc\_topology\_membind\_support, [133](#)
- get\_thread\_cpubind
  - hwloc\_topology\_cpubind\_support, [131](#)
- group
  - hwloc\_obj\_attr\_u, [126](#)
- Helpers for manipulating glibc sched affinity, [106](#)
- Helpers for manipulating Linux libnuma bitmask, [110](#)
- Helpers for manipulating Linux libnuma nodemask\_t, [112](#)
- Helpers for manipulating Linux libnuma unsigned long masks, [108](#)
- HWLOC\_CPUBIND\_NOMEMBIND
  - hwlocality\_cpubinding, [68](#)
- HWLOC\_CPUBIND\_PROCESS
  - hwlocality\_cpubinding, [68](#)
- HWLOC\_CPUBIND\_STRICT
  - hwlocality\_cpubinding, [68](#)
- HWLOC\_CPUBIND\_THREAD
  - hwlocality\_cpubinding, [68](#)
- HWLOC\_MEMBIND\_BIND
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_DEFAULT
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_FIRSTTOUCH
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_INTERLEAVE
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_MIGRATE
  - hwlocality\_membinding, [73](#)
- HWLOC\_MEMBIND\_NEXTTOUCH
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_NOCPUBIND
  - hwlocality\_membinding, [73](#)
- HWLOC\_MEMBIND\_PROCESS
  - hwlocality\_membinding, [73](#)
- HWLOC\_MEMBIND\_REPLICATE
  - hwlocality\_membinding, [74](#)
- HWLOC\_MEMBIND\_STRICT
  - hwlocality\_membinding, [73](#)
- HWLOC\_MEMBIND\_THREAD
  - hwlocality\_membinding, [73](#)
- HWLOC\_OBJ\_CACHE
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_CORE
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_GROUP
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_MACHINE
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_MISC
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_NODE
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_PU
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_SOCKET
  - hwlocality\_types, [50](#)
- HWLOC\_OBJ\_SYSTEM
  - hwlocality\_types, [50](#)
- HWLOC\_TOPOLOGY\_FLAG\_IS\_THISSYSTEM
  - hwlocality\_configuration, [56](#)
- HWLOC\_TOPOLOGY\_FLAG\_WHOLE\_SYSTEM
  - hwlocality\_configuration, [55](#)
- HWLOC\_TYPE\_DEPTH\_MULTIPLE
  - hwlocality\_information, [61](#)
- HWLOC\_TYPE\_DEPTH\_UNKNOWN
  - hwlocality\_information, [61](#)
- HWLOC\_TYPE\_UNORDERED
  - hwlocality\_types, [49](#)
- hwloc\_alloc
  - hwlocality\_membinding, [74](#)
- hwloc\_alloc\_membind

- hwlocality\_membinding, 74
- hwloc\_alloc\_membind\_nodeset
  - hwlocality\_membinding, 75
- hwloc\_alloc\_membind\_policy
  - hwlocality\_helper\_binding, 89
- hwloc\_alloc\_membind\_policy\_nodeset
  - hwlocality\_helper\_binding, 89
- HWLOC\_API\_VERSION
  - hwlocality\_api\_version, 47
- hwloc\_bitmap\_allbut
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_alloc
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_alloc\_full
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_and
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_andnot
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_asprintf
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_clr
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_clr\_range
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_compare
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_compare\_first
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_copy
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_dup
  - hwlocality\_bitmap, 101
- hwloc\_bitmap\_fill
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_first
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_foreach\_begin
  - hwlocality\_bitmap, 99
- hwloc\_bitmap\_foreach\_end
  - hwlocality\_bitmap, 99
- hwloc\_bitmap\_free
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_from\_ith\_ulong
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_from\_ulong
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_intersects
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_isequal
  - hwlocality\_bitmap, 102
- hwloc\_bitmap\_isfull
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_isincluded
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_isset
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_iszero
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_last
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_next
  - hwlocality\_bitmap, 103
- hwloc\_bitmap\_not
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_only
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_or
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_set
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_set\_ith\_ulong
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_set\_range
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_singlify
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_snprintf
  - hwlocality\_bitmap, 104
- hwloc\_bitmap\_sscanf
  - hwlocality\_bitmap, 105
- hwloc\_bitmap\_t
  - hwlocality\_bitmap, 100
- hwloc\_bitmap\_taskset\_asprintf
  - hwlocality\_bitmap, 105
- hwloc\_bitmap\_taskset\_snprintf
  - hwlocality\_bitmap, 105
- hwloc\_bitmap\_taskset\_sscanf
  - hwlocality\_bitmap, 105
- hwloc\_bitmap\_to\_ith\_ulong
  - hwlocality\_bitmap, 105
- hwloc\_bitmap\_to\_ulong

- hwlocality\_bitmap, 105
- hwloc\_bitmap\_weight
  - hwlocality\_bitmap, 106
- hwloc\_bitmap\_xor
  - hwlocality\_bitmap, 106
- hwloc\_bitmap\_zero
  - hwlocality\_bitmap, 106
- hwloc\_compare\_types
  - hwlocality\_types, 50
- hwloc\_compare\_types\_e
  - hwlocality\_types, 49
- hwloc\_const\_bitmap\_t
  - hwlocality\_bitmap, 100
- hwloc\_const\_cpuset\_t
  - hwlocality\_sets, 48
- hwloc\_const\_nodeset\_t
  - hwlocality\_sets, 48
- hwloc\_cpubind\_flags\_t
  - hwlocality\_cpubinding, 68
- hwloc\_cpuset\_from\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, 107
- hwloc\_cpuset\_from\_linux\_libnuma\_  
bitmask
  - hwlocality\_linux\_libnuma\_bitmask,  
111
- hwloc\_cpuset\_from\_linux\_libnuma\_  
nodemask
  - hwlocality\_linux\_libnuma\_  
nodemask, 113
- hwloc\_cpuset\_from\_linux\_libnuma\_  
ulongs
  - hwlocality\_linux\_libnuma\_ulongs,  
109
- hwloc\_cpuset\_from\_nodeset
  - hwlocality\_helper\_nodeset\_convert,  
94
- hwloc\_cpuset\_from\_nodeset\_strict
  - hwlocality\_helper\_nodeset\_convert,  
94
- hwloc\_cpuset\_t
  - hwlocality\_sets, 48
- hwloc\_cpuset\_to\_glibc\_sched\_affinity
  - hwlocality\_glibc\_sched, 107
- hwloc\_cpuset\_to\_linux\_libnuma\_  
bitmask
  - hwlocality\_linux\_libnuma\_bitmask,  
111
- hwloc\_cpuset\_to\_linux\_libnuma\_  
nodemask
  - hwlocality\_linux\_libnuma\_  
nodemask, 113
- hwloc\_cpuset\_to\_linux\_libnuma\_ulongs
  - hwlocality\_linux\_libnuma\_ulongs,  
109
- hwloc\_cpuset\_to\_nodeset
  - hwlocality\_helper\_nodeset\_convert,  
94
- hwloc\_cpuset\_to\_nodeset\_strict
  - hwlocality\_helper\_nodeset\_convert,  
94
- hwloc\_cuda\_get\_device\_cpuset
  - hwlocality\_cuda, 114
- hwloc\_cudart\_get\_device\_cpuset
  - hwlocality\_cudart, 114
- hwloc\_distribute
  - hwlocality\_helper\_binding, 89
- hwloc\_distributev
  - hwlocality\_helper\_binding, 90
- hwloc\_free
  - hwlocality\_membinding, 75
- hwloc\_get\_ancestor\_obj\_by\_depth
  - hwlocality\_helper\_traversal\_basic,  
80
- hwloc\_get\_ancestor\_obj\_by\_type
  - hwlocality\_helper\_traversal\_basic,  
80
- hwloc\_get\_area\_membind
  - hwlocality\_membinding, 75
- hwloc\_get\_area\_membind\_nodeset
  - hwlocality\_membinding, 75
- hwloc\_get\_cache\_covering\_cpuset
  - hwlocality\_helper\_find\_cache, 87
- hwloc\_get\_child\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 85
- hwloc\_get\_closest\_objs
  - hwlocality\_helper\_traversal, 88
- hwloc\_get\_common\_ancestor\_obj
  - hwlocality\_helper\_traversal\_basic,  
80
- hwloc\_get\_cpubind
  - hwlocality\_cpubinding, 69

- hwloc\_get\_depth\_type
  - hwlocality\_information, 61
- hwloc\_get\_first\_largest\_obj\_inside\_-cpuset
  - hwlocality\_helper\_find\_inside, 83
- hwloc\_get\_largest\_objs\_inside\_cpuset
  - hwlocality\_helper\_find\_inside, 83
- hwloc\_get\_membind
  - hwlocality\_membinding, 76
- hwloc\_get\_membind\_nodeset
  - hwlocality\_membinding, 76
- hwloc\_get\_nbobjs\_by\_depth
  - hwlocality\_information, 61
- hwloc\_get\_nbobjs\_by\_type
  - hwlocality\_information, 62
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_-depth
  - hwlocality\_helper\_find\_inside, 83
- hwloc\_get\_nbobjs\_inside\_cpuset\_by\_-type
  - hwlocality\_helper\_find\_inside, 83
- hwloc\_get\_next\_child
  - hwlocality\_helper\_traversal\_basic, 80
- hwloc\_get\_next\_obj\_by\_depth
  - hwlocality\_helper\_traversal\_basic, 80
- hwloc\_get\_next\_obj\_by\_type
  - hwlocality\_helper\_traversal\_basic, 80
- hwloc\_get\_next\_obj\_covering\_cpuset\_-by\_depth
  - hwlocality\_helper\_find\_coverings, 86
- hwloc\_get\_next\_obj\_covering\_cpuset\_-by\_type
  - hwlocality\_helper\_find\_coverings, 86
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_-depth
  - hwlocality\_helper\_find\_inside, 83
- hwloc\_get\_next\_obj\_inside\_cpuset\_by\_-type
  - hwlocality\_helper\_find\_inside, 84
- hwloc\_get\_obj\_below\_array\_by\_type
  - hwlocality\_helper\_traversal, 88
- hwloc\_get\_obj\_below\_by\_type
  - hwlocality\_helper\_traversal, 88
- hwloc\_get\_obj\_by\_depth
  - hwlocality\_traversal, 63
- hwloc\_get\_obj\_by\_type
  - hwlocality\_traversal, 63
- hwloc\_get\_obj\_covering\_cpuset
  - hwlocality\_helper\_find\_covering, 85
- hwloc\_get\_obj\_inside\_cpuset\_by\_depth
  - hwlocality\_helper\_find\_inside, 84
- hwloc\_get\_obj\_inside\_cpuset\_by\_type
  - hwlocality\_helper\_find\_inside, 84
- hwloc\_get\_proc\_cpupbind
  - hwlocality\_cpupbinding, 69
- hwloc\_get\_proc\_membind
  - hwlocality\_membinding, 76
- hwloc\_get\_proc\_membind\_nodeset
  - hwlocality\_membinding, 76
- hwloc\_get\_pu\_obj\_by\_os\_index
  - hwlocality\_helper\_traversal\_basic, 81
- hwloc\_get\_root\_obj
  - hwlocality\_helper\_traversal\_basic, 81
- hwloc\_get\_shared\_cache\_covering\_obj
  - hwlocality\_helper\_find\_cache, 87
- hwloc\_get\_thread\_cpupbind
  - hwlocality\_cpupbinding, 69
- hwloc\_get\_type\_depth
  - hwlocality\_information, 62
- hwloc\_get\_type\_depth\_e
  - hwlocality\_information, 61
- hwloc\_get\_type\_or\_above\_depth
  - hwlocality\_helper\_types, 78
- hwloc\_get\_type\_or\_below\_depth
  - hwlocality\_helper\_types, 78
- hwloc\_ibv\_get\_device\_cpuset
  - hwlocality\_openfabrics, 115
- hwloc\_linux\_get\_tid\_cpupbind
  - hwlocality\_linux, 108
- hwloc\_linux\_parse\_cpumap\_file
  - hwlocality\_linux, 108
- hwloc\_linux\_set\_tid\_cpupbind
  - hwlocality\_linux, 108
- hwloc\_membind\_flags\_t
  - hwlocality\_membinding, 73

- hwloc\_membind\_policy\_t
  - hwlocality\_membinding, 74
- hwloc\_mx\_board\_get\_device\_cpuset
  - hwlocality\_myriexpress, 116
- hwloc\_mx\_endpoint\_get\_device\_cpuset
  - hwlocality\_myriexpress, 116
- hwloc\_nodeset\_from\_linux\_libnuma\_-bitmask
  - hwlocality\_linux\_libnuma\_bitmask, 111
- hwloc\_nodeset\_from\_linux\_libnuma\_-nodemask
  - hwlocality\_linux\_libnuma\_-nodemask, 113
- hwloc\_nodeset\_from\_linux\_libnuma\_-ulongs
  - hwlocality\_linux\_libnuma\_ulongs, 109
- hwloc\_nodeset\_t
  - hwlocality\_sets, 48
- hwloc\_nodeset\_to\_linux\_libnuma\_-bitmask
  - hwlocality\_linux\_libnuma\_bitmask, 111
- hwloc\_nodeset\_to\_linux\_libnuma\_-nodemask
  - hwlocality\_linux\_libnuma\_-nodemask, 113
- hwloc\_nodeset\_to\_linux\_libnuma\_-ulongs
  - hwlocality\_linux\_libnuma\_ulongs, 110
- hwloc\_obj, 119
  - allowed\_cpuset, 121
  - allowed\_nodeset, 121
  - arity, 122
  - attr, 122
  - children, 122
  - complete\_cpuset, 122
  - complete\_nodeset, 122
  - cpuset, 123
  - depth, 123
  - first\_child, 123
  - infos, 123
  - infos\_count, 123
  - last\_child, 123
  - logical\_index, 123
  - memory, 124
  - name, 124
  - next\_cousin, 124
  - next\_sibling, 124
  - nodeset, 124
  - online\_cpuset, 124
  - os\_index, 125
  - os\_level, 125
  - parent, 125
  - prev\_cousin, 125
  - prev\_sibling, 125
  - sibling\_rank, 125
  - type, 125
  - userdata, 125
- hwloc\_obj\_attr\_snprintf
  - hwlocality\_conversion, 64
- hwloc\_obj\_attr\_u, 126
  - cache, 126
  - group, 126
- hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, 117
  - depth, 118
  - linesize, 118
  - size, 118
- hwloc\_obj\_attr\_u::hwloc\_group\_attr\_s, 118
  - depth, 118
- hwloc\_obj\_cpuset\_snprintf
  - hwlocality\_conversion, 64
- hwloc\_obj\_get\_info\_by\_name
  - hwlocality\_conversion, 65
- hwloc\_obj\_info\_s, 127
  - name, 127
  - value, 127
- hwloc\_obj\_is\_in\_subtree
  - hwlocality\_helper\_traversal\_basic, 81
- hwloc\_obj\_memory\_s, 128
  - local\_memory, 129
  - page\_types, 129
  - page\_types\_len, 129
  - total\_memory, 129
- hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s, 128
  - count, 128

- size, [128](#)
- hwloc\_obj\_snprintf
  - hwlocality\_conversion, [65](#)
- hwloc\_obj\_t
  - hwlocality\_objects, [52](#)
- hwloc\_obj\_type\_of\_string
  - hwlocality\_conversion, [65](#)
- hwloc\_obj\_type\_snprintf
  - hwlocality\_conversion, [65](#)
- hwloc\_obj\_type\_string
  - hwlocality\_conversion, [66](#)
- hwloc\_obj\_type\_t
  - hwlocality\_types, [49](#)
- hwloc\_set\_area\_membind
  - hwlocality\_membinding, [76](#)
- hwloc\_set\_area\_membind\_nodeuset
  - hwlocality\_membinding, [76](#)
- hwloc\_set\_cpubind
  - hwlocality\_cpubinding, [69](#)
- hwloc\_set\_membind
  - hwlocality\_membinding, [77](#)
- hwloc\_set\_membind\_nodeuset
  - hwlocality\_membinding, [77](#)
- hwloc\_set\_proc\_cpubind
  - hwlocality\_cpubinding, [69](#)
- hwloc\_set\_proc\_membind
  - hwlocality\_membinding, [77](#)
- hwloc\_set\_proc\_membind\_nodeuset
  - hwlocality\_membinding, [77](#)
- hwloc\_set\_thread\_cpubind
  - hwlocality\_cpubinding, [70](#)
- hwloc\_topology\_check
  - hwlocality\_creation, [52](#)
- hwloc\_topology\_cpubind\_support, [130](#)
  - get\_proc\_cpubind, [130](#)
  - get\_thisproc\_cpubind, [130](#)
  - get\_thisthread\_cpubind, [130](#)
  - get\_thread\_cpubind, [131](#)
  - set\_proc\_cpubind, [131](#)
  - set\_thisproc\_cpubind, [131](#)
  - set\_thisthread\_cpubind, [131](#)
  - set\_thread\_cpubind, [131](#)
- hwloc\_topology\_destroy
  - hwlocality\_creation, [52](#)
- hwloc\_topology\_discovery\_support, [131](#)
  - pu, [132](#)
- hwloc\_topology\_export\_xml
  - hwlocality\_tinker, [59](#)
- hwloc\_topology\_export\_xmlbuffer
  - hwlocality\_tinker, [59](#)
- hwloc\_topology\_flags\_e
  - hwlocality\_configuration, [55](#)
- hwloc\_topology\_get\_allowed\_cpuset
  - hwlocality\_helper\_cpuset, [91](#)
- hwloc\_topology\_get\_allowed\_nodeuset
  - hwlocality\_helper\_nodeuset, [92](#)
- hwloc\_topology\_get\_complete\_cpuset
  - hwlocality\_helper\_cpuset, [91](#)
- hwloc\_topology\_get\_complete\_nodeuset
  - hwlocality\_helper\_nodeuset, [92](#)
- hwloc\_topology\_get\_depth
  - hwlocality\_information, [62](#)
- hwloc\_topology\_get\_online\_cpuset
  - hwlocality\_helper\_cpuset, [91](#)
- hwloc\_topology\_get\_support
  - hwlocality\_configuration, [56](#)
- hwloc\_topology\_get\_topology\_cpuset
  - hwlocality\_helper\_cpuset, [91](#)
- hwloc\_topology\_get\_topology\_nodeuset
  - hwlocality\_helper\_nodeuset, [93](#)
- hwloc\_topology\_ignore\_all\_keep\_ -
  - structure
  - hwlocality\_configuration, [56](#)
- hwloc\_topology\_ignore\_type
  - hwlocality\_configuration, [56](#)
- hwloc\_topology\_ignore\_type\_keep\_ -
  - structure
  - hwlocality\_configuration, [56](#)
- hwloc\_topology\_init
  - hwlocality\_creation, [53](#)
- hwloc\_topology\_insert\_misc\_object\_ -
  - by\_cpuset
  - hwlocality\_tinker, [59](#)
- hwloc\_topology\_insert\_misc\_object\_ -
  - by\_parent
  - hwlocality\_tinker, [60](#)
- hwloc\_topology\_is\_thissystem
  - hwlocality\_information, [62](#)
- hwloc\_topology\_load
  - hwlocality\_creation, [53](#)
- hwloc\_topology\_membind\_support, [132](#)
  - alloc\_membind, [133](#)

- bind\_membind, 133
- firsttouch\_membind, 133
- get\_area\_membind, 133
- get\_proc\_membind, 133
- get\_thisproc\_membind, 133
- get\_thisthread\_membind, 133
- interleave\_membind, 134
- migrate\_membind, 134
- nexttouch\_membind, 134
- replicate\_membind, 134
- set\_area\_membind, 134
- set\_proc\_membind, 134
- set\_thisproc\_membind, 134
- set\_thisthread\_membind, 134
- hwloc\_topology\_set\_flags
  - hwlocality\_configuration, 57
- hwloc\_topology\_set\_fsroot
  - hwlocality\_configuration, 57
- hwloc\_topology\_set\_pid
  - hwlocality\_configuration, 57
- hwloc\_topology\_set\_synthetic
  - hwlocality\_configuration, 58
- hwloc\_topology\_set\_xml
  - hwlocality\_configuration, 58
- hwloc\_topology\_set\_xmlbuffer
  - hwlocality\_configuration, 58
- hwloc\_topology\_support, 135
  - cpubind, 135
  - discovery, 135
  - membind, 135
- hwloc\_topology\_t
  - hwlocality\_topology, 48
- hwlocality\_configuration
  - HWLOC\_TOPOLOGY\_FLAG\_-  
IS\_THISSYSTEM, 56
  - HWLOC\_TOPOLOGY\_FLAG\_-  
WHOLE\_SYSTEM, 55
- hwlocality\_cpubinding
  - HWLOC\_CPUBIND\_-  
NOMEMBIND, 68
  - HWLOC\_CPUBIND\_PROCESS,  
68
  - HWLOC\_CPUBIND\_STRICT, 68
  - HWLOC\_CPUBIND\_THREAD, 68
- hwlocality\_information
  - HWLOC\_TYPE\_DEPTH\_-  
MULTIPLE, 61
  - HWLOC\_TYPE\_DEPTH\_-  
UNKNOWN, 61
- hwlocality\_membinding
  - HWLOC\_MEMBIND\_BIND, 74
  - HWLOC\_MEMBIND\_DEFAULT,  
74
  - HWLOC\_MEMBIND\_-  
FIRSTTOUCH, 74
  - HWLOC\_MEMBIND\_-  
INTERLEAVE, 74
  - HWLOC\_MEMBIND\_MIGRATE,  
73
  - HWLOC\_MEMBIND\_-  
NEXTTOUCH, 74
  - HWLOC\_MEMBIND\_-  
NOCPUBIND, 73
  - HWLOC\_MEMBIND\_PROCESS,  
73
  - HWLOC\_MEMBIND\_-  
REPLICATE, 74
  - HWLOC\_MEMBIND\_STRICT, 73
  - HWLOC\_MEMBIND\_THREAD,  
73
- hwlocality\_types
  - HWLOC\_OBJ\_CACHE, 50
  - HWLOC\_OBJ\_CORE, 50
  - HWLOC\_OBJ\_GROUP, 50
  - HWLOC\_OBJ\_MACHINE, 50
  - HWLOC\_OBJ\_MISC, 50
  - HWLOC\_OBJ\_NODE, 50
  - HWLOC\_OBJ\_PU, 50
  - HWLOC\_OBJ\_SOCKET, 50
  - HWLOC\_OBJ\_SYSTEM, 50
  - HWLOC\_TYPE\_UNORDERED,  
49
- hwlocality\_api\_version
  - HWLOC\_API\_VERSION, 47
- hwlocality\_bitmap
  - hwloc\_bitmap\_allbut, 100
  - hwloc\_bitmap\_alloc, 100
  - hwloc\_bitmap\_alloc\_full, 100
  - hwloc\_bitmap\_and, 100
  - hwloc\_bitmap\_andnot, 100
  - hwloc\_bitmap\_asprintf, 100

- hwloc\_bitmap\_clr, 101
- hwloc\_bitmap\_clr\_range, 101
- hwloc\_bitmap\_compare, 101
- hwloc\_bitmap\_compare\_first, 101
- hwloc\_bitmap\_copy, 101
- hwloc\_bitmap\_dup, 101
- hwloc\_bitmap\_fill, 102
- hwloc\_bitmap\_first, 102
- hwloc\_bitmap\_foreach\_begin, 99
- hwloc\_bitmap\_foreach\_end, 99
- hwloc\_bitmap\_free, 102
- hwloc\_bitmap\_from\_ith\_ulong, 102
- hwloc\_bitmap\_from\_ulong, 102
- hwloc\_bitmap\_intersects, 102
- hwloc\_bitmap\_isequal, 102
- hwloc\_bitmap\_isfull, 103
- hwloc\_bitmap\_isincluded, 103
- hwloc\_bitmap\_isset, 103
- hwloc\_bitmap\_iszero, 103
- hwloc\_bitmap\_last, 103
- hwloc\_bitmap\_next, 103
- hwloc\_bitmap\_not, 104
- hwloc\_bitmap\_only, 104
- hwloc\_bitmap\_or, 104
- hwloc\_bitmap\_set, 104
- hwloc\_bitmap\_set\_ith\_ulong, 104
- hwloc\_bitmap\_set\_range, 104
- hwloc\_bitmap\_singlify, 104
- hwloc\_bitmap\_snprintf, 104
- hwloc\_bitmap\_sscanf, 105
- hwloc\_bitmap\_t, 100
- hwloc\_bitmap\_taskset\_asprintf, 105
- hwloc\_bitmap\_taskset\_snprintf, 105
- hwloc\_bitmap\_taskset\_sscanf, 105
- hwloc\_bitmap\_to\_ith\_ulong, 105
- hwloc\_bitmap\_to\_ulong, 105
- hwloc\_bitmap\_weight, 106
- hwloc\_bitmap\_xor, 106
- hwloc\_bitmap\_zero, 106
- hwloc\_const\_bitmap\_t, 100
- hwlocality\_configuration
  - hwloc\_topology\_flags\_e, 55
  - hwloc\_topology\_get\_support, 56
  - hwloc\_topology\_ignore\_all\_keep\_-  
structure, 56
  - hwloc\_topology\_ignore\_type, 56
  - hwloc\_topology\_ignore\_type\_-  
keep\_structure, 56
  - hwloc\_topology\_set\_flags, 57
  - hwloc\_topology\_set\_fsroot, 57
  - hwloc\_topology\_set\_pid, 57
  - hwloc\_topology\_set\_synthetic, 58
  - hwloc\_topology\_set\_xml, 58
  - hwloc\_topology\_set\_xmlbuffer, 58
- hwlocality\_conversion
  - hwloc\_obj\_attr\_snprintf, 64
  - hwloc\_obj\_cpuset\_snprintf, 64
  - hwloc\_obj\_get\_info\_by\_name, 65
  - hwloc\_obj\_snprintf, 65
  - hwloc\_obj\_type\_of\_string, 65
  - hwloc\_obj\_type\_snprintf, 65
  - hwloc\_obj\_type\_string, 66
- hwlocality\_cpubinding
  - hwloc\_cpusbind\_flags\_t, 68
  - hwloc\_get\_cpusbind, 69
  - hwloc\_get\_proc\_cpusbind, 69
  - hwloc\_get\_thread\_cpusbind, 69
  - hwloc\_set\_cpusbind, 69
  - hwloc\_set\_proc\_cpusbind, 69
  - hwloc\_set\_thread\_cpusbind, 70
- hwlocality\_creation
  - hwloc\_topology\_check, 52
  - hwloc\_topology\_destroy, 52
  - hwloc\_topology\_init, 53
  - hwloc\_topology\_load, 53
- hwlocality\_cuda
  - hwloc\_cuda\_get\_device\_cpuset, 114
- hwlocality\_cudart
  - hwloc\_cudart\_get\_device\_cpuset,  
114
- hwlocality\_glibc\_sched
  - hwloc\_cpuset\_from\_glibc\_sched\_-  
affinity, 107
  - hwloc\_cpuset\_to\_glibc\_sched\_-  
affinity, 107
- hwlocality\_helper\_binding
  - hwloc\_alloc\_membind\_policy, 89
  - hwloc\_alloc\_membind\_policy\_-  
nodeset, 89
  - hwloc\_distribute, 89
  - hwloc\_distributev, 90
- hwlocality\_helper\_cpuset



- hwloc\_topology\_get\_allowed\_-cpuset, [91](#)
- hwloc\_topology\_get\_complete\_-cpuset, [91](#)
- hwloc\_topology\_get\_online\_cpuset, [91](#)
- hwloc\_topology\_get\_topology\_-cpuset, [91](#)
- hwlocality\_helper\_find\_cache
  - hwloc\_get\_cache\_covering\_cpuset, [87](#)
  - hwloc\_get\_shared\_cache\_-covering\_obj, [87](#)
- hwlocality\_helper\_find\_covering
  - hwloc\_get\_child\_covering\_cpuset, [85](#)
  - hwloc\_get\_obj\_covering\_cpuset, [85](#)
- hwlocality\_helper\_find\_coverings
  - hwloc\_get\_next\_obj\_covering\_-cpuset\_by\_depth, [86](#)
  - hwloc\_get\_next\_obj\_covering\_-cpuset\_by\_type, [86](#)
- hwlocality\_helper\_find\_inside
  - hwloc\_get\_first\_largest\_obj\_-inside\_cpuset, [83](#)
  - hwloc\_get\_largest\_objs\_inside\_-cpuset, [83](#)
  - hwloc\_get\_nbobjs\_inside\_cpuset\_-by\_depth, [83](#)
  - hwloc\_get\_nbobjs\_inside\_cpuset\_-by\_type, [83](#)
  - hwloc\_get\_next\_obj\_inside\_-cpuset\_by\_depth, [83](#)
  - hwloc\_get\_next\_obj\_inside\_-cpuset\_by\_type, [84](#)
  - hwloc\_get\_obj\_inside\_cpuset\_by\_-depth, [84](#)
  - hwloc\_get\_obj\_inside\_cpuset\_by\_-type, [84](#)
- hwlocality\_helper\_nodeset
  - hwloc\_topology\_get\_allowed\_-nodeset, [92](#)
  - hwloc\_topology\_get\_complete\_-nodeset, [92](#)
  - hwloc\_topology\_get\_topology\_-nodeset, [93](#)
- hwlocality\_helper\_nodeset\_convert
  - hwloc\_cpuset\_from\_nodeset, [94](#)
  - hwloc\_cpuset\_from\_nodeset\_strict, [94](#)
  - hwloc\_cpuset\_to\_nodeset, [94](#)
  - hwloc\_cpuset\_to\_nodeset\_strict, [94](#)
- hwlocality\_helper\_traversal
  - hwloc\_get\_closest\_objs, [88](#)
  - hwloc\_get\_obj\_below\_array\_by\_-type, [88](#)
  - hwloc\_get\_obj\_below\_by\_type, [88](#)
- hwlocality\_helper\_traversal\_basic
  - hwloc\_get\_ancestor\_obj\_by\_depth, [80](#)
  - hwloc\_get\_ancestor\_obj\_by\_type, [80](#)
  - hwloc\_get\_common\_ancestor\_obj, [80](#)
  - hwloc\_get\_next\_child, [80](#)
  - hwloc\_get\_next\_obj\_by\_depth, [80](#)
  - hwloc\_get\_next\_obj\_by\_type, [80](#)
  - hwloc\_get\_pu\_obj\_by\_os\_index, [81](#)
  - hwloc\_get\_root\_obj, [81](#)
  - hwloc\_obj\_is\_in\_subtree, [81](#)
- hwlocality\_helper\_types
  - hwloc\_get\_type\_or\_above\_depth, [78](#)
  - hwloc\_get\_type\_or\_below\_depth, [78](#)
- hwlocality\_information
  - hwloc\_get\_depth\_type, [61](#)
  - hwloc\_get\_nbobjs\_by\_depth, [61](#)
  - hwloc\_get\_nbobjs\_by\_type, [62](#)
  - hwloc\_get\_type\_depth, [62](#)
  - hwloc\_get\_type\_depth\_e, [61](#)
  - hwloc\_topology\_get\_depth, [62](#)
  - hwloc\_topology\_is\_thissystem, [62](#)
- hwlocality\_linux
  - hwloc\_linux\_get\_tid\_cpupbind, [108](#)
  - hwloc\_linux\_parse\_cpumap\_file, [108](#)
  - hwloc\_linux\_set\_tid\_cpupbind, [108](#)
- hwlocality\_linux\_libnuma\_bitmask
  - hwloc\_cpuset\_from\_linux\_-libnuma\_bitmask, [111](#)

- hwloc\_cpuset\_to\_linux\_libnuma\_-  
bitmask, [111](#)
- hwloc\_nodeset\_from\_linux\_-  
libnuma\_bitmask, [111](#)
- hwloc\_nodeset\_to\_linux\_libnuma\_-  
bitmask, [111](#)
- hwlocality\_linux\_libnuma\_nodemask
  - hwloc\_cpuset\_from\_linux\_-  
libnuma\_nodemask, [113](#)
  - hwloc\_cpuset\_to\_linux\_libnuma\_-  
nodemask, [113](#)
  - hwloc\_nodeset\_from\_linux\_-  
libnuma\_nodemask, [113](#)
  - hwloc\_nodeset\_to\_linux\_libnuma\_-  
nodemask, [113](#)
- hwlocality\_linux\_libnuma\_ulong
  - hwloc\_cpuset\_from\_linux\_-  
libnuma\_ulong, [109](#)
  - hwloc\_cpuset\_to\_linux\_libnuma\_-  
ulong, [109](#)
  - hwloc\_nodeset\_from\_linux\_-  
libnuma\_ulong, [109](#)
  - hwloc\_nodeset\_to\_linux\_libnuma\_-  
ulong, [110](#)
- hwlocality\_membinding
  - hwloc\_alloc, [74](#)
  - hwloc\_alloc\_membind, [74](#)
  - hwloc\_alloc\_membind\_nodeset, [75](#)
  - hwloc\_free, [75](#)
  - hwloc\_get\_area\_membind, [75](#)
  - hwloc\_get\_area\_membind\_nodeset,  
[75](#)
  - hwloc\_get\_membind, [76](#)
  - hwloc\_get\_membind\_nodeset, [76](#)
  - hwloc\_get\_proc\_membind, [76](#)
  - hwloc\_get\_proc\_membind\_nodeset,  
[76](#)
  - hwloc\_membind\_flags\_t, [73](#)
  - hwloc\_membind\_policy\_t, [74](#)
  - hwloc\_set\_area\_membind, [76](#)
  - hwloc\_set\_area\_membind\_nodeset,  
[76](#)
  - hwloc\_set\_membind, [77](#)
  - hwloc\_set\_membind\_nodeset, [77](#)
  - hwloc\_set\_proc\_membind, [77](#)
- hwloc\_set\_proc\_membind\_nodeset,  
[77](#)
- hwlocality\_myriexpress
  - hwloc\_mx\_board\_get\_device\_-  
cpuset, [116](#)
  - hwloc\_mx\_endpoint\_get\_device\_-  
cpuset, [116](#)
- hwlocality\_objects
  - hwloc\_obj\_t, [52](#)
- hwlocality\_openfabrics
  - hwloc\_ibv\_get\_device\_cpuset, [115](#)
- hwlocality\_sets
  - hwloc\_const\_cpuset\_t, [48](#)
  - hwloc\_const\_nodeset\_t, [48](#)
  - hwloc\_cpuset\_t, [48](#)
  - hwloc\_nodeset\_t, [48](#)
- hwlocality\_tinker
  - hwloc\_topology\_export\_xml, [59](#)
  - hwloc\_topology\_export\_xmlbuffer,  
[59](#)
  - hwloc\_topology\_insert\_misc\_-  
object\_by\_cpuset, [59](#)
  - hwloc\_topology\_insert\_misc\_-  
object\_by\_parent, [60](#)
- hwlocality\_topology
  - hwloc\_topology\_t, [48](#)
- hwlocality\_traversal
  - hwloc\_get\_obj\_by\_depth, [63](#)
  - hwloc\_get\_obj\_by\_type, [63](#)
- hwlocality\_types
  - hwloc\_compare\_types, [50](#)
  - hwloc\_compare\_types\_e, [49](#)
  - hwloc\_obj\_type\_t, [49](#)
- infos
  - hwloc\_obj, [123](#)
- infos\_count
  - hwloc\_obj, [123](#)
- interleave\_membind
  - hwloc\_topology\_membind\_support,  
[134](#)
- last\_child
  - hwloc\_obj, [123](#)
- linesize

- hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [118](#)
- Linux-only helpers, [107](#)
- local\_memory
  - hwloc\_obj\_memory\_s, [129](#)
- logical\_index
  - hwloc\_obj, [123](#)
- membind
  - hwloc\_topology\_support, [135](#)
- memory
  - hwloc\_obj, [124](#)
- Memory binding, [70](#)
- migrate\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- Myrinet Express-Specific Functions, [115](#)
- name
  - hwloc\_obj, [124](#)
  - hwloc\_obj\_info\_s, [127](#)
- next\_cousin
  - hwloc\_obj, [124](#)
- next\_sibling
  - hwloc\_obj, [124](#)
- nexttouch\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- nodeset
  - hwloc\_obj, [124](#)
- Nodeset Helpers, [92](#)
- Object sets, [48](#)
- Object Type Helpers, [78](#)
- Object/String Conversion, [63](#)
- online\_cpuset
  - hwloc\_obj, [124](#)
- OpenFabrics-Specific Functions, [115](#)
- os\_index
  - hwloc\_obj, [125](#)
- os\_level
  - hwloc\_obj, [125](#)
- page\_types
  - hwloc\_obj\_memory\_s, [129](#)
- page\_types\_len
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [118](#)
- hwloc\_obj\_memory\_s, [129](#)
- parent
  - hwloc\_obj, [125](#)
- prev\_cousin
  - hwloc\_obj, [125](#)
- prev\_sibling
  - hwloc\_obj, [125](#)
- pu
  - hwloc\_topology\_discovery\_support, [132](#)
- replicate\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- Retrieve Objects, [63](#)
- set\_area\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- set\_proc\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [131](#)
- set\_proc\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- set\_thisproc\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [131](#)
- set\_thisproc\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- set\_thisthread\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [131](#)
- set\_thisthread\_membind
  - hwloc\_topology\_membind\_support, [134](#)
- set\_thread\_cpupbind
  - hwloc\_topology\_cpupbind\_support, [131](#)
- sibling\_rank
  - hwloc\_obj, [125](#)
- size
  - hwloc\_obj\_attr\_u::hwloc\_cache\_attr\_s, [118](#)

hwloc\_obj\_memory\_s::hwloc\_obj\_memory\_page\_type\_s, [128](#)

The bitmap API, [95](#)

Tinker with topologies., [59](#)

Topology context, [47](#)

Topology Object Types, [49](#)

Topology Objects, [51](#)

total\_memory

hwloc\_obj\_memory\_s, [129](#)

type

hwloc\_obj, [125](#)

userdata

hwloc\_obj, [125](#)

value

hwloc\_obj\_info\_s, [127](#)