libtopology

0.9

Generated by Doxygen 1.5.9

Fri Jul 3 19:00:23 2009

Contents

Chapter 1

Libtopology

Portable abstraction of hierarchical architectures for high-performance computing

1.1 Introduction

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

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

libtopology supports the following operating systems:

- Linux (including old kernels not having sysfs topology information, with knowledge of cpusets, offline cpus, and Kerrighed support)
- Solaris
- AIX
- Darwin
- OSF/1 (aka. Tru64)
- Windows
- For other OSes, only the number of processors is available for now.

For development and debugging purposes, libtopology 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

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

libtopology 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 Interface example interface below. It may also be used to binding applications onto certain cores or memory nodes. Several utility programs are also provided to ease command-line manipulation of topology objects, binding of processes, ...

1.2 Installation

libtopology (http://libtopology.gforge.inria.fr/) is available under the CeCILL-B license (BSD-like). It is hosted by the INRIA Gforge (http://gforge.inria.fr/projects/libtopology/). The current SVN snapshot can be fetched with:

- svn checkout svn://scm.gforge.inria.fr/svn/libtopology/trunk libtopology
- cd libtopology
- autoreconf -ifv

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

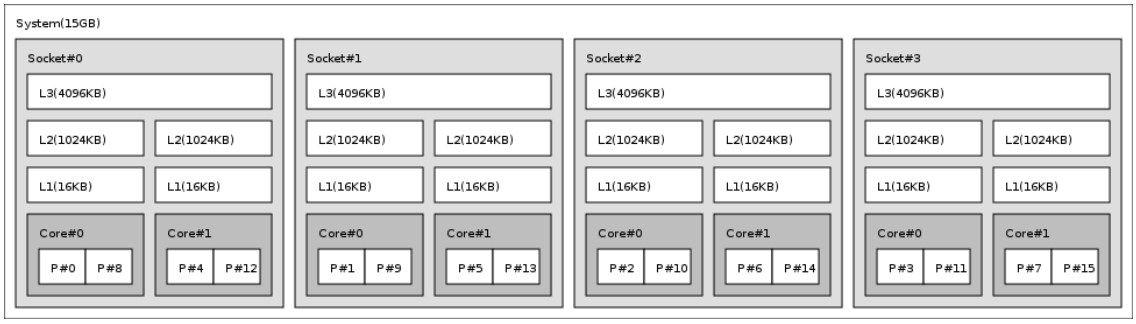
Installation by itself is as usual:

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

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

1.3 Examples

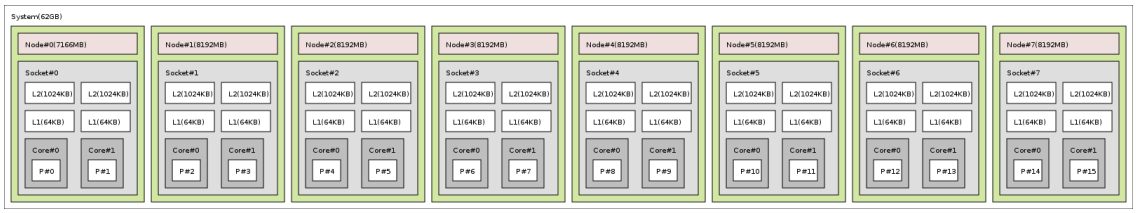
On a 4-socket 2-core machine with hyperthreading, the `lstopo` tool may show the following outputs:




```
System(15GB)
  Socket#0 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#0
      P#8
```

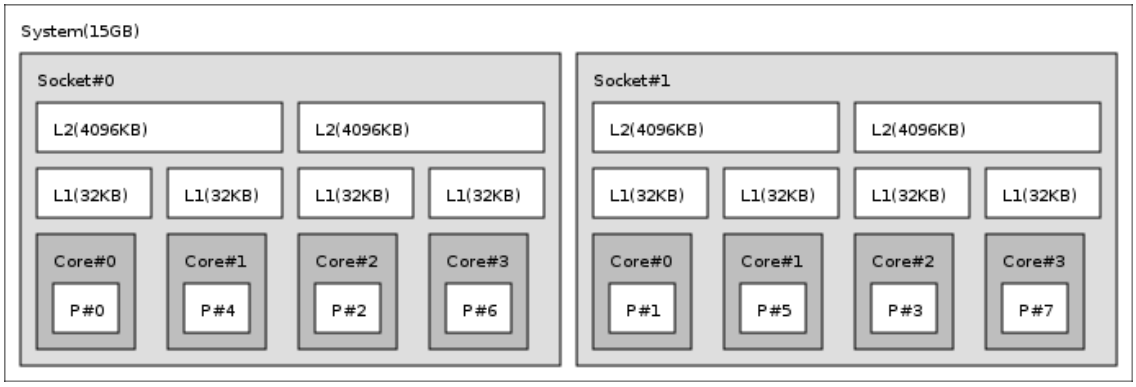
```
    L2(1024KB) + L1(16KB) + Core#1
      P#4
      P#12
  Socket#1 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#1
      P#9
    L2(1024KB) + L1(16KB) + Core#1
      P#5
      P#13
  Socket#2 + L3(4096KB)
    L2(1024KB) + L1(16KB) + Core#0
      P#2
      P#10
    L2(1024KB) + L1(16KB)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

On a 8-socket 2-core Opteron NUMA machine, the `lstopo` tool may show the following outputs:



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

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




```
System(15GB)
  Socket#0
    L2(4096KB)
      L1(32KB) + Core#0 + P#0
      L1(32KB) + Core#1 + P#4
    L2(4096KB)
      L1(32KB) + Core#2 + P#2
      L1(32KB) + Core#3 + P#6
  Socket#1
    L2(4096KB)
      L1(32KB) + Core#0 + P#1
      L1(32KB) + Core#1 + P#5
    L2(4096KB)
      L1(32KB) + Core#2 + P#3
      L1(32KB) + Core#3 + P#7
```

1.4 Interface example

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

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

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

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

static void print_children(topo_topology_t topology, topo_obj_t obj, int depth)
{
        char string[128];
        int i;

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

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

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

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

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


  /* Optionally, get some additional topology information
   * in case we need the topology depth later.
   */
  struct topo_topology_info topoinfo;
  topo_topology_get_info(topology, &topoinfo);


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

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


  /* Print the number of sockets.  */
  depth = topo_get_type_depth(topology, TOPO_OBJ_SOCKET);
  if (depth == TOPO_TYPE_DEPTH_UNKNOWN)
          printf("The number of sockets is unknown\n");
  else
          printf("%u socket(s)\n", topo_get_depth_nbobjs(topology, depth));



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

  /* Get last one.  */
  topo_obj_t obj = topo_get_obj_by_depth(topology, depth,
topo_get_depth_nbobjs(topology, depth) - 1);
  if (!obj)
          return 0;

  /* Get its cpuset.  */
  topo_cpuset_t cpuset = obj->cpuset;

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

  /* And try to bind ourself there.  */
  if (topo_set_cpubind(topology, &cpuset, 0)) {
          char s[TOPO_CPUSET_STRING_LENGTH + 1];
          topo_cpuset_snprintf(s, sizeof(s), &obj->cpuset);
```

```
                printf("Couldn't bind to cpuset %s\n", s);
        }


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

        return 0;
}
```

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

The basic interface is available in topology.h, a lot of traversal examples are available as inlines in topology/helper.h . On Linux, additional helpers are available in topology/linux-libnuma.h and on glibc-based systems, additional helpers are available in topology/glibc-sched.h

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

1.5 Questions and bugs

Questions should be sent to the devel mailing list (libtopology-devel@lists.gforge.inria.fr, http://lists.gforge.inria.fr/cgi-bin/mailman/listinfo/libtopology-devel). Bug reports should be reported in the tracker (http://gforge.inria.fr/tracker/?group_-id=1758).

1.6 Credits

libtopology is developed by the INRIA Runtime Team-Project (http://runtime.bordeaux.inria.fr/) (headed by Raymond Namyst http://dept-info.labri.fr/~namyst/).

Chapter 2

Glossary

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

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

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

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

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

Arity The number of children of an object

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

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

Cousin objects Objects of the same type as the current object

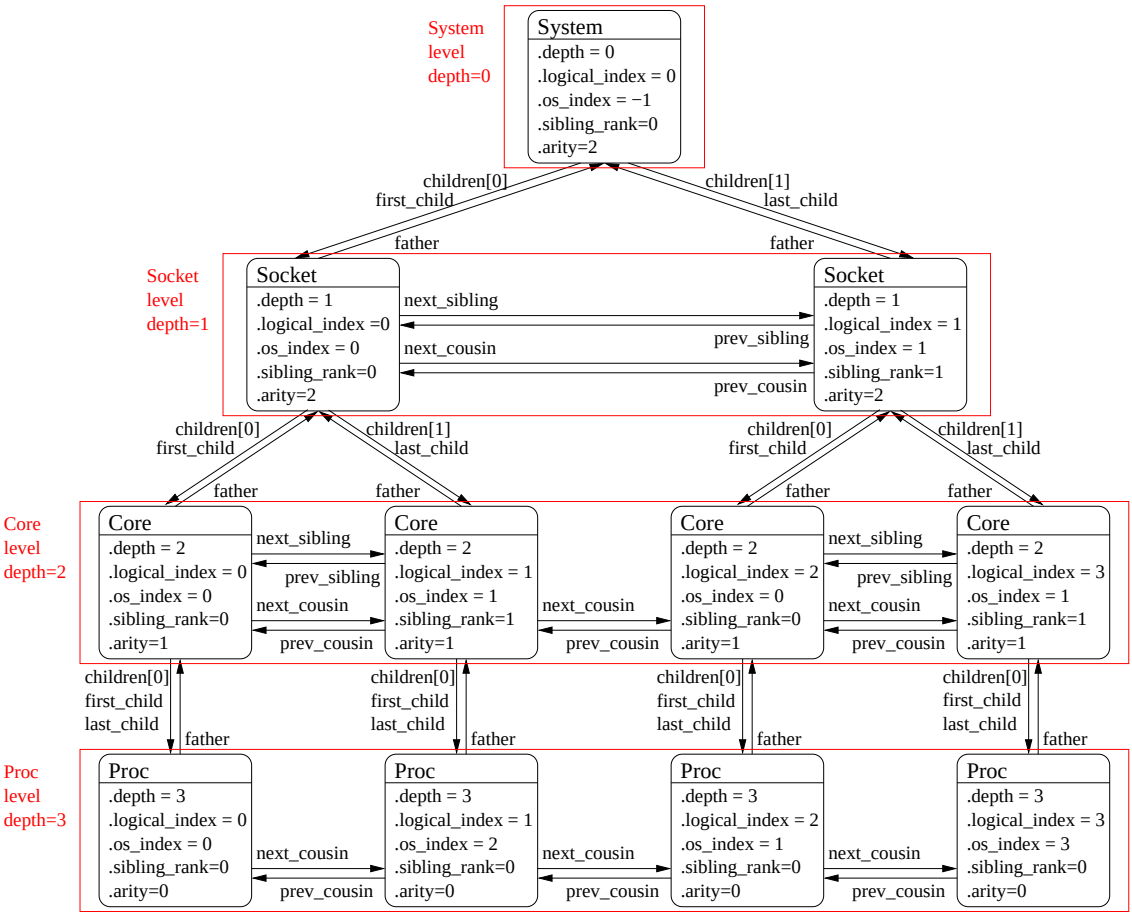
Level Set of objects of the same type

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

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

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

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



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

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Topology and Topology Info

Data Structures

- struct topo_topology_info

  Global information about a Topology context,.

Typedefs

- typedef struct topo_topology * topo_topology_t

  Topology context.

6.1.1 Typedef Documentation

6.1.1.1 typedef struct topo_topology* topo_topology_t

Topology context.

To be initialized with topo_topology_init() and built with topo_topology_load().

6.2 Topology Object Types

Defines

- #define TOPO_OBJ_TYPE_MAX (TOPO_OBJ_MISC+1)
  Maximal value of an Object Type.

Enumerations

- enum topo_obj_type_t {
  TOPO_OBJ_SYSTEM, TOPO_OBJ_MACHINE, TOPO_OBJ_NODE, TOPO_OBJ_SOCKET,
  TOPO_OBJ_CACHE, TOPO_OBJ_CORE, TOPO_OBJ_PROC, TOPO_OBJ_MISC }
  Type of topology Object.

Functions

- int topo_get_type_order (topo_obj_type_t type)
  Convert an object type into a number that permits to compare them.
- topo_obj_type_t topo_get_order_type (int order)
  Converse of topo_get_type_oder().

6.2.1 Define Documentation

6.2.1.1 #define TOPO_OBJ_TYPE_MAX (TOPO_OBJ_MISC+1)

Maximal value of an Object Type.

6.2.2 Enumeration Type Documentation

6.2.2.1 enum topo_obj_type_t

Type of topology Object.

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

Enumerator:

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

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

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

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

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

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

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

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

6.2.3 Function Documentation

6.2.3.1 topo_obj_type_t topo_get_order_type (int *order*)

Converse of topo_get_type_oder().

This is the converse of topo_get_type_order().

6.2.3.2 int topo_get_type_order (topo_obj_type_t *type*)

Convert an object type into a number that permits to compare them.

Types shouldn't be compared as they are, since newer ones may be added in the future. This function returns an integer value that can be used instead.

Note:

topo_get_type_order(TOPO_OBJ_SYSTEM) will always be the lowest value, and topo_get_type_order(TOPO_OBJ_PROC) will always be the highest value.

6.3 Topology Objects

Data Structures

- struct topo_obj

  Structure of a topology Object.

- union topo_obj_attr_u

  Object type-specific Attributes.

Typedefs

- typedef struct topo_obj * topo_obj_t

6.3.1 Typedef Documentation

6.3.1.1 typedef struct topo_obj* topo_obj_t

6.4 Create and Destroy Topologies

Functions

- int topo_topology_init (topo_topology_t *topologyp)

  Allocate a topology context.

- int topo_topology_load (topo_topology_t topology)

  Build the actual topology.

- void topo_topology_destroy (topo_topology_t topology)

  Terminate and free a topology context.

- void topo_topology_check (topo_topology_t topology)

  Run internal checks on a topology structure.

6.4.1 Function Documentation

6.4.1.1 void topo_topology_check (topo_topology_t *topology*)

Run internal checks on a topology structure.

Parameters:

topology is the topology to be checked

6.4.1.2 void topo_topology_destroy (topo_topology_t *topology*)

Terminate and free a topology context.

Parameters:

topology is the topology to be freed

6.4.1.3 int topo_topology_init (topo_topology_t * *topologyp*)

Allocate a topology context.

Parameters:

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

Returns:

0 on success, -1 on error.

6.4.1.4 int topo_topology_load (topo_topology_t *topology*)

Build the actual topology.

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

Parameters:

topology is the topology to be loaded with objects.

Returns:

0 on success, -1 on error.

See also:

Configure Topology Detection

6.5 Configure Topology Detection

Enumerations

- enum topo_flags_e { TOPO_FLAGS_WHOLE_SYSTEM = (1<<1) }

  Flags to be set onto a topology context before load.

Functions

- int topo_topology_ignore_type (topo_topology_t topology, topo_obj_type_t type)

  Ignore an object type.

- int topo_topology_ignore_type_keep_structure (topo_topology_t topology, topo_obj_type_t type)

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

- int topo_topology_ignore_all_keep_structure (topo_topology_t topology)

  Ignore all objects that do not bring any structure.

- int topo_topology_set_flags (topo_topology_t topology, unsigned long flags)

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

- int topo_topology_set_fsys_root (topo_topology_t __topo_restrict topology, const char *__topo_restrict fsys_root_path)

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

- int topo_topology_set_synthetic (topo_topology_t __topo_restrict topology, const char *__topo_restrict description)

  Enable synthetic topology.

- int topo_topology_set_xml (topo_topology_t __topo_restrict topology, const char *__topo_restrict xmlpath)

  Enable XML-file based topology.

6.5.1 Detailed Description

These functions can optionally be called between topology_init() and 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.

6.5.2 Enumeration Type Documentation

6.5.2.1 enum topo_flags_e

Flags to be set onto a topology context before load.

Flags should be given to topo_topology_set_flags().

Enumerator:

TOPO_FLAGS_WHOLE_SYSTEM

6.5.3 Function Documentation

6.5.3.1 int topo_topology_ignore_all_keep_structure (topo_topology_t *topology*)

Ignore all objects that do not bring any structure.

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

6.5.3.2 int topo_topology_ignore_type (topo_topology_t *topology*, topo_obj_type_t *type*)

Ignore an object type.

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

6.5.3.3 int topo_topology_ignore_type_keep_structure (topo_topology_t *topology*, topo_obj_type_t *type*)

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

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

6.5.3.4 int topo_topology_set_flags (topo_topology_t *topology*, unsigned long *flags*)

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

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

6.5.3.5 int topo_topology_set_fsys_root (topo_topology_t __topo_restrict *topology*, const char *__topo_restrict *fsys_root_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 `fsys_root_path` instead of the main file-system root. Not using the main file-system root causes the is_fake field of the topo_topology_info structure to be set.

6.5.3.6 int topo_topology_set_synthetic (topo_topology_t __topo_restrict *topology*, const char *__topo_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.

6.5.3.7 int topo_topology_set_xml (topo_topology_t __topo_restrict *topology*, const char *__topo_restrict *xmlpath*)

Enable XML-file based topology.

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

6.6 Get some Topology Information

Defines

- #define TOPO_TYPE_DEPTH_UNKNOWN -1

  No object of given type exists in the topology.

- #define TOPO_TYPE_DEPTH_MULTIPLE -2

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

Functions

- int topo_topology_get_info (topo_topology_t __topo_restrict topology, struct topo_topology_info *__topo_restrict info)

  Get additional global information about the topology.

- unsigned topo_get_type_depth (topo_topology_t topology, topo_obj_type_t type)

  Returns the depth of objects of type `type`.

- topo_obj_type_t topo_get_depth_type (topo_topology_t topology, unsigned depth)

  Returns the type of objects at depth `depth`.

- unsigned topo_get_depth_nbobjs (topo_topology_t topology, unsigned depth)

  Returns the width of level at depth `depth`.

6.6.1 Define Documentation

6.6.1.1 #define TOPO_TYPE_DEPTH_MULTIPLE -2

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

6.6.1.2 #define TOPO_TYPE_DEPTH_UNKNOWN -1

No object of given type exists in the topology.

6.6.2 Function Documentation

6.6.2.1 unsigned topo_get_depth_nbobjs (topo_topology_t *topology*, unsigned *depth*)

Returns the width of level at depth `depth`.

6.6.2.2 topo_obj_type_t topo_get_depth_type (topo_topology_t *topology*, unsigned *depth*)

Returns the type of objects at depth `depth`.

6.6.2.3 unsigned topo_get_type_depth (topo_topology_t *topology*, topo_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 TOPO_TYPE_DEPTH_UNKNOWN.

If type is absent but a similar type is acceptable, see also topo_get_type_or_below_depth() and topo_get_type_or_above_depth().

6.6.2.4 int topo_topology_get_info (topo_topology_t __topo_restrict *topology*, struct topo_topology_info *__topo_restrict *info*)

Get additional global information about the topology.

Retrieve additional global information about a loaded topology context. Might be useful if the whole topology depth is needed for instance.

6.7 Retrieve Objects

Functions

- topo_obj_t topo_get_obj_by_depth (topo_topology_t topology, unsigned depth, unsigned index)

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

6.7.1 Function Documentation

6.7.1.1 topo_obj_t topo_get_obj_by_depth (topo_topology_t *topology*, unsigned *depth*, unsigned *index*)

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

6.8 Object/String Conversion

Functions

- const char * topo_obj_type_string (topo_obj_type_t type)

  Return a stringified topology object type.

- topo_obj_type_t topo_obj_type_of_string (const char *string)

  Return an object type from the string.

- int topo_obj_snprintf (char *__topo_restrict string, size_t size, topo_topology_t topology, topo_obj_t obj, const char *__topo_restrict indexprefix, int verbose)

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

- int topo_obj_cpuset_snprintf (char *__topo_restrict str, size_t size, size_t nobj, const topo_obj_t *__topo_restrict objs)

  Stringify the cpuset containing a set of objects.

6.8.1 Function Documentation

6.8.1.1 int topo_obj_cpuset_snprintf (char *__topo_restrict *str*, size_t *size*, size_t *nobj*, const topo_obj_t *__topo_restrict *objs*)

Stringify the cpuset containing a set of objects.

Returns:

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

6.8.1.2 int topo_obj_snprintf (char *__topo_restrict *string*, size_t *size*, topo_topology_t *topology*, topo_obj_t *obj*, const char *__topo_restrict *indexprefix*, int *verbose*)

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

Returns:

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

6.8.1.3 topo_obj_type_t topo_obj_type_of_string (const char * *string*)

Return an object type from the string.

6.8.1.4 const char* topo_obj_type_string (topo_obj_type_t *type*)

Return a stringified topology object type.

6.9 Binding

Enumerations

- enum topo_cpubind_policy_t { TOPO_CPUBIND_PROCESS = (1<<0), TOPO_CPUBIND_THREAD = (1<<1), TOPO_CPUBIND_STRICT = (1<<2) }

  Process/Thread binding policy.

Functions

- int topo_set_cpubind (topo_topology_t topology, const topo_cpuset_t *set, int policy)

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

- int topo_set_proc_cpubind (topo_topology_t topology, topo_pid_t pid, const topo_cpuset_t *set, int policy)

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

- int topo_set_thread_cpubind (topo_topology_t topology, topo_thread_t tid, const topo_cpuset_t *set, int policy)

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

6.9.1 Detailed Description

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

Note:

Some OSes do not provide all ways to bind processes, threads, etc and the corresponding binding functions may fail. The most portable version that should be preferred over the others, whenever possible, is

```
topo_set_cpubind(topology, set, 0),
```

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

```
topo_set_cpubind(topology, set, TOPO_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.

6.9.2 Enumeration Type Documentation

6.9.2.1 enum topo_cpubind_policy_t

Process/Thread binding policy.

These flags can be used to refine the binding policy.

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

Note:

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

Enumerator:

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

TOPO_CPUBIND_THREAD Bind current thread of current process.

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

6.9.3 Function Documentation

6.9.3.1 int topo_set_cpubind (topo_topology_t *topology*, const topo_cpuset_t * *set*, int *policy*)

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

6.9.3.2 int topo_set_proc_cpubind (topo_topology_t *topology*, topo_pid_t *pid*, const topo_cpuset_t * *set*, int *policy*)

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

Note:

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

6.9.3.3 int topo_set_thread_cpubind (topo_topology_t *topology*, topo_thread_t *tid*, const topo_cpuset_t * *set*, int *policy*)

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

Note:

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

6.10 Object Type Helpers

Functions

- static __inline__ unsigned topo_get_type_or_below_depth (topo_topology_t topology, topo_obj_type_t type)

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

- static __inline__ unsigned topo_get_type_or_above_depth (topo_topology_t topology, topo_obj_type_t type)

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

- static __inline__ int topo_get_type_nbobjs (topo_topology_t topology, topo_obj_type_t type)

  Returns the width of level type `type`.

6.10.1 Function Documentation

6.10.1.1 static __inline__ int topo_get_type_nbobjs (topo_topology_t *topology*, topo_obj_type_t *type*) [static]

Returns the width of level type `type`.

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

6.10.1.2 static __inline__ unsigned topo_get_type_or_above_depth (topo_topology_t *topology*, topo_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`.

6.10.1.3 static __inline__ unsigned topo_get_type_or_below_depth (topo_topology_t *topology*, topo_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`.

6.11 Basic Traversal Helpers

Functions

- static __inline__ topo_obj_t topo_get_system_obj (topo_topology_t topology)

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

- static __inline__ topo_obj_t topo_get_obj (topo_topology_t topology, topo_obj_type_t type, unsigned index)

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

- static __inline__ topo_obj_t topo_get_next_obj_by_depth (topo_topology_t topology, unsigned depth, topo_obj_t prev)

  Returns the next object at depth `depth`.

- static __inline__ topo_obj_t topo_get_next_obj (topo_topology_t topology, topo_obj_type_t type, topo_obj_t prev)

  Returns the next object of type `type`.

- static __inline__ topo_obj_t topo_get_next_child (topo_topology_t topology, topo_obj_t father, topo_obj_t prev)

  Return the next child.

- static __inline__ topo_obj_t topo_get_common_ancestor_obj (topo_obj_t obj1, topo_obj_t obj2)

  Returns the common father object to objects lvl1 and lvl2.

- static __inline__ int topo_obj_is_in_subtree (topo_obj_t obj, topo_obj_t subtree_root)

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

6.11.1 Function Documentation

6.11.1.1 static __inline__ topo_obj_t topo_get_common_ancestor_obj (topo_obj_t *obj1*, topo_obj_t *obj2*) [static]

Returns the common father object to objects lvl1 and lvl2.

6.11.1.2 static __inline__ topo_obj_t topo_get_next_child (topo_topology_t *topology*, topo_obj_t *father*, topo_obj_t *prev*) [static]

Return the next child.

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

6.11.1.3 static __inline__ topo_obj_t topo_get_next_obj (topo_topology_t *topology*, topo_obj_type_t *type*, topo_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 topo_get_next_obj_by_depth().

6.11.1.4 static __inline__ topo_obj_t topo_get_next_obj_by_depth (topo_topology_t *topology*, unsigned *depth*, topo_obj_t *prev*) [static]

Returns the next object at depth `depth`.

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

6.11.1.5 static __inline__ topo_obj_t topo_get_obj (topo_topology_t *topology*, topo_obj_type_t *type*, unsigned *index*) [static]

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

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

6.11.1.6 static __inline__ topo_obj_t topo_get_system_obj (topo_topology_t *topology*) [static]

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

6.11.1.7 static __inline__ int topo_obj_is_in_subtree (topo_obj_t *obj*, topo_obj_t *subtree_root*) [static]

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

6.12 Finding similar Objects Included in a CPU set

Functions

- static __inline__ topo_obj_t topo_get_next_obj_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_next_obj_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_obj_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, unsigned index)

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

- static __inline__ topo_obj_t topo_get_obj_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, unsigned index)

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

- static __inline__ unsigned topo_get_nbobjs_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth)

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

- static __inline__ int topo_get_nbobjs_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type)

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

6.12.1 Function Documentation

6.12.1.1 static __inline__ int topo_get_nbobjs_below_cpuset (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_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 below CPU set `set`, 0 is returned. If there are several levels with objects of that type below CPU set `set`, -1 is returned.

6.12.1.2 static __inline__ unsigned topo_get_nbobjs_below_cpuset_by_depth (topo_topology_t *topology*, const topo_cpuset_t * *set*, unsigned *depth*) [static]

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

6.12.1.3 static __inline__ topo_obj_t topo_get_next_obj_below_cpuset (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_obj_type_t *type*, topo_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 topo_get_next_obj_below_cpuset_by_depth().

6.12.1.4 static __inline__ topo_obj_t topo_get_next_obj_below_cpuset_by_depth (topo_topology_t *topology*, const topo_cpuset_t * *set*, unsigned *depth*, topo_obj_t *prev*) [static]

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

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

6.12.1.5 static __inline__ topo_obj_t topo_get_obj_below_cpuset (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_obj_type_t *type*, unsigned *index*) [static]

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

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

6.12.1.6 static __inline__ topo_obj_t topo_get_obj_below_cpuset_by_depth (topo_topology_t *topology*, const topo_cpuset_t * *set*, unsigned *depth*, unsigned *index*) [static]

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

6.13 Finding a single Object covering at least CPU set

Functions

- static topo_obj_t topo_get_cpuset_covering_child (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_t father)

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

- static topo_obj_t topo_get_cpuset_covering_obj (topo_topology_t topology, const topo_cpuset_t *set)

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

6.13.1 Function Documentation

6.13.1.1 static topo_obj_t topo_get_cpuset_covering_child (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_obj_t *father*) [inline, static]

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

Returns:

`NULL` if no child matches.

6.13.1.2 static topo_obj_t topo_get_cpuset_covering_obj (topo_topology_t *topology*, const topo_cpuset_t * *set*) [inline, static]

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

Returns:

`NULL` if no object matches.

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

Functions

- static __inline__ topo_obj_t topo_get_next_obj_above_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_next_obj_above_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, topo_obj_t prev)

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

6.14.1 Function Documentation

6.14.1.1 static __inline__ topo_obj_t topo_get_next_obj_above_cpuset (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_obj_type_t *type*, topo_obj_t *prev*) [static]

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

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

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

6.14.1.2 static __inline__ topo_obj_t topo_get_next_obj_above_cpuset_by_depth (topo_topology_t *topology*, const topo_cpuset_t * *set*, unsigned *depth*, topo_obj_t *prev*) [static]

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

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

6.15 Cache-specific Finding Helpers

Functions

- static __inline__ topo_obj_t topo_get_cpuset_covering_cache (topo_topology_t topology, const topo_cpuset_t *set)

  Get the first cache covering a cpuset `set`.

- static __inline__ topo_obj_t topo_get_shared_cache_above (topo_topology_t topology, topo_obj_t obj)

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

6.15.1 Function Documentation

6.15.1.1 static __inline__ topo_obj_t topo_get_cpuset_covering_cache (topo_topology_t *topology*, const topo_cpuset_t * *set*) [static]

Get the first cache covering a cpuset `set`.

Returns:

`NULL` if no cache matches

6.15.1.2 static __inline__ topo_obj_t topo_get_shared_cache_above (topo_topology_t *topology*, topo_obj_t *obj*) [static]

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

Returns:

`NULL` if no cache matches

6.16 Advanced Traversal Helpers

Functions

- int topo_get_cpuset_objs (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_t *__-topo_restrict objs, int max)

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

- int topo_get_closest_objs (topo_topology_t topology, topo_obj_t src, topo_obj_t *__topo_restrict objs, int max)

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

6.16.1 Function Documentation

6.16.1.1 int topo_get_closest_objs (topo_topology_t *topology*, topo_obj_t *src*, topo_obj_t *__topo_restrict *objs*, int *max*)

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

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

Returns:

the number of objects returned in `objs`.

6.16.1.2 int topo_get_cpuset_objs (topo_topology_t *topology*, const topo_cpuset_t * *set*, topo_obj_t *__topo_restrict *objs*, int *max*)

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

Returns:

the number of objects returned in `objs`.

6.17 Binding Helpers

Functions

- static __inline__ void topo_distribute (topo_topology_t topology, topo_obj_t root, topo_cpuset_t *cpuset, int n)

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

6.17.1 Function Documentation

6.17.1.1 static __inline__ void topo_distribute (topo_topology_t *topology*, topo_obj_t *root*, topo_cpuset_t * *cpuset*, int *n*) [static]

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

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

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

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

6.18 The Cpuset API

Data Structures

- struct topo_cpuset_t

  Set of CPUs represented as a bitmask.

Defines

- #define TOPO_CPUSET_ZERO (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_ZERO }

  Predefined cpuset with no CPU set.

- #define TOPO_CPUSET_FULL (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_FULL }

  Predefined cpuset with all CPUs set.

- #define TOPO_CPUSET_CPU(cpu) ({ topo_cpuset_t __set = TOPO_CPUSET_ZERO; TOPO_CPUSUBSET_CPUSUBSET(__set,cpu) = TOPO_CPUSUBSET_VAL(cpu); __set; })

  Predefined cpuset with CPU `cpu` *set.*

- #define TOPO_CPUSET_STRING_LENGTH (TOPO_CPUSET_SUBSTRING_COUNT∗(TOPO_CPUSET_SUBSTRING_LENGTH+1))

  Maximal required length of a string for printing a CPU set.

- #define TOPO_PRIxCPUSET "s"

  Printf format for printing a CPU set.

- #define topo_cpuset_foreach_begin(cpu, set)

  Loop macro iterating on CPU set `set`.

- #define topo_cpuset_foreach_end() }

  End of loop.

Functions

- static __inline__ int topo_cpuset_snprintf (char ∗__topo_restrict buf, size_t buflen, const topo_cpuset_t ∗__topo_restrict set)

  Stringify a cpuset.

- static __inline__ void topo_cpuset_from_string (const char ∗__topo_restrict string, topo_cpuset_t ∗__topo_restrict set)

  Parse a cpuset string.

- static __inline__ void topo_cpuset_zero (topo_cpuset_t ∗set)

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

- static __inline__ void topo_cpuset_fill (topo_cpuset_t *set)

  Fill CPU set `set`.

- static __inline__ void topo_cpuset_from_ulong (topo_cpuset_t *set, unsigned long mask)

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

- static __inline__ void topo_cpuset_from_ith_ulong (topo_cpuset_t *set, int i, unsigned long mask)

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

- static __inline__ unsigned long topo_cpuset_to_ulong (const topo_cpuset_t *set)

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

- static __inline__ unsigned long topo_cpuset_to_ith_ulong (const topo_cpuset_t *set, int i)

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

- static __inline__ void topo_cpuset_cpu (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_all_but_cpu (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_set (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_set_range (topo_cpuset_t *set, unsigned begincpu, unsigned endcpu)

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

- static __inline__ void topo_cpuset_clr (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ int topo_cpuset_isset (const topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ int topo_cpuset_iszero (const topo_cpuset_t *set)

  Test whether set `set` *is zero.*

- static __inline__ int topo_cpuset_isfull (const topo_cpuset_t *set)

  Test whether set `set` *is full.*

- static __inline__ int topo_cpuset_isequal (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_intersects (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_isincluded (const topo_cpuset_t *sub_set, const topo_cpuset_t *super_set)

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

- static __inline__ void topo_cpuset_orset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_andset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_clearset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_xorset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ int topo_cpuset_first (const topo_cpuset_t *cpuset)

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

- static __inline__ void topo_cpuset_singlify (topo_cpuset_t *set)

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

- static __inline__ int topo_cpuset_compar_first (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_compar (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_weight (const topo_cpuset_t *set)

  Compute the weight of CPU set `set`.

6.18.1 Detailed Description

For use in libtopology itself, a topo_cpuset_t represents a set of logical processors.

Note:

cpusets are indexed by OS logical processor number.

6.18.2 Define Documentation

6.18.2.1 #define TOPO_CPUSET_CPU(cpu) ({ topo_cpuset_t __set = TOPO_CPUSET_ZERO; TOPO_CPUSUBSET_CPUSUBSET(__set,cpu) = TOPO_CPUSUBSET_VAL(cpu); __set; })

Predefined cpuset with CPU `cpu` set.

6.18.2.2 #define topo_cpuset_foreach_begin(cpu, set)

Value:

```
for (cpu = 0; cpu < TOPO_NBMAXCPUS; cpu++) \
                if (topo_cpuset_isset(set, cpu)) {
```

Loop macro iterating on CPU set `set`.

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

6.18.2.3 #define topo_cpuset_foreach_end() }

End of loop.

See also:

topo_cpuset_foreach_begin

6.18.2.4 #define TOPO_CPUSET_FULL (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_FULL }

Predefined cpuset with all CPUs set.

6.18.2.5 #define TOPO_CPUSET_STRING_LENGTH (TOPO_CPUSET_SUBSTRING_COUNT∗(TOPO_CPUSET_SUBSTRING_LENGTH+1))

Maximal required length of a string for printing a CPU set.

Fewer characters may be needed if part of the CPU set is empty.

6.18.2.6 #define TOPO_CPUSET_ZERO (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_ZERO }

Predefined cpuset with no CPU set.

6.18.2.7 #define TOPO_PRIxCPUSET "s"

Printf format for printing a CPU set.

6.18.3 Function Documentation

6.18.3.1 static __inline__ void topo_cpuset_all_but_cpu (topo_cpuset_t ∗ *set*, unsigned *cpu*) [static]

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

6.18.3.2 static __inline__ void topo_cpuset_andset (topo_cpuset_t * *set*, const topo_cpuset_t * *modifier_set*) [static]

And set `modifier_set` into set `set`.

6.18.3.3 static __inline__ void topo_cpuset_clearset (topo_cpuset_t * *set*, const topo_cpuset_t * *modifier_set*) [static]

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

6.18.3.4 static __inline__ void topo_cpuset_clr (topo_cpuset_t * *set*, unsigned *cpu*) [static]

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

6.18.3.5 static __inline__ int topo_cpuset_compar (const topo_cpuset_t * *set1*, const topo_cpuset_t * *set2*) [static]

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

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

6.18.3.6 static __inline__ int topo_cpuset_compar_first (const topo_cpuset_t * *set1*, const topo_cpuset_t * *set2*) [static]

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

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

6.18.3.7 static __inline__ void topo_cpuset_cpu (topo_cpuset_t * *set*, unsigned *cpu*) [static]

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

6.18.3.8 static __inline__ void topo_cpuset_fill (topo_cpuset_t * *set*) [static]

Fill CPU set `set`.

6.18.3.9 static __inline__ int topo_cpuset_first (const topo_cpuset_t * *cpuset*) [static]

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

6.18.3.10 static __inline__ void topo_cpuset_from_ith_ulong (topo_cpuset_t * *set*, int *i*, unsigned long *mask*) [static]

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

6.18.3.11 static __inline__ void topo_cpuset_from_string (const char *__topo_restrict *string*, topo_cpuset_t *__topo_restrict *set*) [static]

Parse a cpuset string.

Must start and end with a digit.

6.18.3.12 static __inline__ void topo_cpuset_from_ulong (topo_cpuset_t * *set*, unsigned long *mask*) [static]

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

6.18.3.13 static __inline__ int topo_cpuset_intersects (const topo_cpuset_t * *set1*, const topo_cpuset_t * *set2*) [static]

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

6.18.3.14 static __inline__ int topo_cpuset_isequal (const topo_cpuset_t * *set1*, const topo_cpuset_t * *set2*) [static]

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

6.18.3.15 static __inline__ int topo_cpuset_isfull (const topo_cpuset_t * *set*) [static]

Test whether set `set` is full.

6.18.3.16 static __inline__ int topo_cpuset_isincluded (const topo_cpuset_t * *sub_set*, const topo_cpuset_t * *super_set*) [static]

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

6.18.3.17 static __inline__ int topo_cpuset_isset (const topo_cpuset_t * *set*, unsigned *cpu*) [static]

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

6.18.3.18 static __inline__ int topo_cpuset_iszero (const topo_cpuset_t * *set*) [static]

Test whether set `set` is zero.

6.18.3.19 static __inline__ void topo_cpuset_orset (topo_cpuset_t * *set*, const topo_cpuset_t * *modifier_set*) [static]

Or set `modifier_set` into set `set`.

6.18.3.20 static __inline__ void topo_cpuset_set (topo_cpuset_t * *set*, unsigned *cpu*) [static]

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

6.18.3.21 static __inline__ void topo_cpuset_set_range (topo_cpuset_t * *set*, unsigned *begincpu*, unsigned *endcpu*) [static]

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

6.18.3.22 static __inline__ void topo_cpuset_singlify (topo_cpuset_t * *set*) [static]

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

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

6.18.3.23 static __inline__ int topo_cpuset_snprintf (char *__topo_restrict *buf*, size_t *buflen*, const topo_cpuset_t *__topo_restrict *set*) [static]

Stringify a cpuset.

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

Returns:

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

6.18.3.24 static __inline__ unsigned long topo_cpuset_to_ith_ulong (const topo_cpuset_t * *set*, int *i*) [static]

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

6.18.3.25 static __inline__ unsigned long topo_cpuset_to_ulong (const topo_cpuset_t * *set*) [static]

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

6.18.3.26 static __inline__ int topo_cpuset_weight (const topo_cpuset_t * *set*) [static]

Compute the weight of CPU set `set`.

6.18.3.27 static __inline__ void topo_cpuset_xorset (topo_cpuset_t * *set*, const topo_cpuset_t * *modifier_set*) [static]

Xor set `set` with set `modifier_set`.

6.18.3.28 static __inline__ void topo_cpuset_zero (topo_cpuset_t * *set*) [static]

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

Empty CPU set `set`

6.19 Helpers for manipulating glibc sched affinity

Functions

- static __inline__ void topo_cpuset_to_glibc_sched_affinity (topo_topology_t topology, const topo_cpuset_t *toposet, cpu_set_t *schedset, size_t schedsetsize)

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

- static __inline__ void topo_cpuset_from_glibc_sched_affinity (topo_topology_t topology, topo_cpuset_t *toposet, const cpu_set_t *schedset, size_t schedsetsize)

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

6.19.1 Function Documentation

6.19.1.1 static __inline__ void topo_cpuset_from_glibc_sched_affinity (topo_topology_t *topology*, topo_cpuset_t * *toposet*, const cpu_set_t * *schedset*, size_t *schedsetsize*) [static]

Convert libtopology 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

6.19.1.2 static __inline__ void topo_cpuset_to_glibc_sched_affinity (topo_topology_t *topology*, const topo_cpuset_t * *toposet*, cpu_set_t * *schedset*, size_t *schedsetsize*) [static]

Convert libtopology 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

6.20 Helpers for manipulating Linux libnuma unsigned long masks

Functions

- static __inline__ void topo_cpuset_to_linux_libnuma_ulongs (topo_topology_t topology, const topo_cpuset_t *cpuset, unsigned long *mask, unsigned long *maxnode)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_ulongs (topo_topology_t topology, topo_cpuset_t *cpuset, const unsigned long *mask, unsigned long maxnode)

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

6.20.1 Function Documentation

6.20.1.1 static __inline__ void topo_cpuset_from_linux_libnuma_ulongs (topo_topology_t *topology*, topo_cpuset_t * *cpuset*, const unsigned long * *mask*, unsigned long *maxnode*) [static]

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

`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).

6.20.1.2 static __inline__ void topo_cpuset_to_linux_libnuma_ulongs (topo_topology_t *topology*, const topo_cpuset_t * *cpuset*, unsigned long * *mask*, unsigned long * *maxnode*) [static]

Convert libtopology 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.

6.21 Helpers for manipulating Linux libnuma bitmask

Functions

- static __inline__ struct bitmask * topo_cpuset_to_linux_libnuma_bitmask (topo_topology_t topology, const topo_cpuset_t *cpuset)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_bitmask (topo_topology_t topology, topo_cpuset_t *cpuset, const struct bitmask *bitmask)

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

6.21.1 Function Documentation

6.21.1.1 static __inline__ void topo_cpuset_from_linux_libnuma_bitmask (topo_topology_t *topology*, topo_cpuset_t * *cpuset*, const struct bitmask * *bitmask*) [static]

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

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

6.21.1.2 static __inline__ struct bitmask* topo_cpuset_to_linux_libnuma_bitmask (topo_topology_t *topology*, const topo_cpuset_t * *cpuset*) [static, read]

Convert libtopology 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.

6.22 Helpers for manipulating Linux libnuma nodemask_t

Functions

- static __inline__ void topo_cpuset_to_linux_libnuma_nodemask (topo_topology_t topology, const topo_cpuset_t *cpuset, nodemask_t *nodemask)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_nodemask (topo_topology_t topology, topo_cpuset_t *cpuset, const nodemask_t *nodemask)

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

6.22.1 Function Documentation

6.22.1.1 static __inline__ void topo_cpuset_from_linux_libnuma_nodemask (topo_topology_t *topology*, topo_cpuset_t * *cpuset*, const nodemask_t * *nodemask*) [static]

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

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

6.22.1.2 static __inline__ void topo_cpuset_to_linux_libnuma_nodemask (topo_topology_t *topology*, const topo_cpuset_t * *cpuset*, nodemask_t * *nodemask*) [static]

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

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

Chapter 7

Data Structure Documentation

7.1 topo_obj_attr_u::topo_cache_attr_u Struct Reference

Cache-specific Object Attributes.

`#include <topology.h>`

Data Fields

- unsigned long memory_kB
  Size of cache.
- unsigned depth
  Depth of cache.

7.1.1 Detailed Description

Cache-specific Object Attributes.

7.1.2 Field Documentation

7.1.2.1 unsigned topo_obj_attr_u::topo_cache_attr_u::depth

Depth of cache.

7.1.2.2 unsigned long topo_obj_attr_u::topo_cache_attr_u::memory_kB

Size of cache.

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

- topology.h

7.2 topo_cpuset_t Struct Reference

Set of CPUs represented as a bitmask.

`#include <cpuset.h>`

Data Fields

- unsigned long s [TOPO_CPUSUBSET_COUNT]

7.2.1 Detailed Description

Set of CPUs represented as a bitmask.

7.2.2 Field Documentation

7.2.2.1 unsigned long topo_cpuset_t::s[TOPO_CPUSUBSET_COUNT]

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

- cpuset.h

7.3 topo_obj_attr_u::topo_machine_attr_u Struct Reference

`#include <topology.h>`

Data Fields

- char * dmi_board_vendor
  DMI Board Vendor name.
- char * dmi_board_name
  DMI Board Model name.
- unsigned long memory_kB
  Size of memory node.
- unsigned long huge_page_free
  Number of available huge pages.
- unsigned long huge_page_size_kB
  Size of huge pages.

7.3.1 Field Documentation

7.3.1.1 char* topo_obj_attr_u::topo_machine_attr_u::dmi_board_name

DMI Board Model name.

7.3.1.2 char* topo_obj_attr_u::topo_machine_attr_u::dmi_board_vendor

DMI Board Vendor name.

7.3.1.3 unsigned long topo_obj_attr_u::topo_machine_attr_u::huge_page_free

Number of available huge pages.

7.3.1.4 unsigned long topo_obj_attr_u::topo_machine_attr_u::huge_page_size_kB

Size of huge pages.

7.3.1.5 unsigned long topo_obj_attr_u::topo_machine_attr_u::memory_kB

Size of memory node.

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

- topology.h

7.4 topo_obj_attr_u::topo_memory_attr_u Struct Reference

Node-specific Object Attributes.

`#include <topology.h>`

Data Fields

- unsigned long memory_kB
  Size of memory node.
- unsigned long huge_page_free
  Number of available huge pages.

7.4.1 Detailed Description

Node-specific Object Attributes.

7.4.2 Field Documentation

7.4.2.1 unsigned long topo_obj_attr_u::topo_memory_attr_u::huge_page_free

Number of available huge pages.

7.4.2.2 unsigned long topo_obj_attr_u::topo_memory_attr_u::memory_kB

Size of memory node.

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

- topology.h

7.5 topo_obj_attr_u::topo_misc_attr_u Struct Reference

Misc-specific Object Attributes.

`#include <topology.h>`

Data Fields

- unsigned depth

  Depth of misc object.

7.5.1 Detailed Description

Misc-specific Object Attributes.

7.5.2 Field Documentation

7.5.2.1 unsigned topo_obj_attr_u::topo_misc_attr_u::depth

Depth of misc object.

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

- topology.h

7.6 topo_obj Struct Reference

Structure of a topology Object.

`#include <topology.h>`

Data Fields

- topo_obj_type_t type

  Type of object.

- signed os_index

  OS-provided physical index number.

- union topo_obj_attr_u * attr

  Object type-specific Attributes.

- unsigned depth

  Vertical index in the hierarchy.

- unsigned logical_index

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

- struct topo_obj * next_cousin

  Next object of same type.

- struct topo_obj * prev_cousin

  Previous object of same type.

- struct topo_obj * father

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

- unsigned sibling_rank

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

- struct topo_obj * next_sibling

  Next object below the same father.

- struct topo_obj * prev_sibling

  Previous object below the same father.

- unsigned arity

  Number of children.

- struct topo_obj ** children

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

- struct topo_obj * first_child

  First child.

- struct topo_obj * last_child

  Last child.

- void * userdata

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

- topo_cpuset_t cpuset

  CPUs covered by this object.

7.6.1 Detailed Description

Structure of a topology Object.

Applications mustn't modify any field except userdata .

7.6.2 Field Documentation

7.6.2.1 unsigned topo_obj::arity

Number of children.

7.6.2.2 union topo_obj_attr_u* topo_obj::attr [write]

Object type-specific Attributes.

7.6.2.3 struct topo_obj** topo_obj::children [read]

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

7.6.2.4 topo_cpuset_t topo_obj::cpuset

CPUs covered by this object.

7.6.2.5 unsigned topo_obj::depth

Vertical index in the hierarchy.

7.6.2.6 struct topo_obj* topo_obj::father [read]

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

7.6.2.7 struct topo_obj* topo_obj::first_child [read]

First child.

7.6.2.8 struct topo_obj* topo_obj::last_child [read]

Last child.

7.6.2.9 unsigned topo_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.

7.6.2.10 struct topo_obj* topo_obj::next_cousin [read]

Next object of same type.

7.6.2.11 struct topo_obj* topo_obj::next_sibling [read]

Next object below the same father.

7.6.2.12 signed topo_obj::os_index

OS-provided physical index number.

7.6.2.13 struct topo_obj* topo_obj::prev_cousin [read]

Previous object of same type.

7.6.2.14 struct topo_obj* topo_obj::prev_sibling [read]

Previous object below the same father.

7.6.2.15 unsigned topo_obj::sibling_rank

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

7.6.2.16 topo_obj_type_t topo_obj::type

Type of object.

7.6.2.17 void* topo_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:

- topology.h

7.7 topo_obj_attr_u Union Reference

Object type-specific Attributes.

`#include <topology.h>`

Data Structures

- struct topo_cache_attr_u

  Cache-specific Object Attributes.

- struct topo_machine_attr_u
- struct topo_memory_attr_u

  Node-specific Object Attributes.

- struct topo_misc_attr_u

  Misc-specific Object Attributes.

Data Fields

- struct topo_obj_attr_u::topo_cache_attr_u cache

  Cache-specific Object Attributes.

- struct topo_obj_attr_u::topo_memory_attr_u node

  Node-specific Object Attributes.

- struct topo_obj_attr_u::topo_machine_attr_u machine

  System-specific Object Attributes.

- struct topo_machine_attr_u system
- struct topo_obj_attr_u::topo_misc_attr_u misc

  Misc-specific Object Attributes.

7.7.1 Detailed Description

Object type-specific Attributes.

7.7.2 Field Documentation

7.7.2.1 struct topo_obj_attr_u::topo_cache_attr_u topo_obj_attr_u::cache

Cache-specific Object Attributes.

7.7.2.2 struct topo_obj_attr_u::topo_machine_attr_u topo_obj_attr_u::machine

System-specific Object Attributes.

7.7.2.3 struct topo_obj_attr_u::topo_misc_attr_u topo_obj_attr_u::misc

Misc-specific Object Attributes.

7.7.2.4 struct topo_obj_attr_u::topo_memory_attr_u topo_obj_attr_u::node

Node-specific Object Attributes.

Machine-specific Object Attributes

7.7.2.5 struct topo_machine_attr_u topo_obj_attr_u::system [read]

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

- topology.h

7.8 topo_topology_info Struct Reference

Global information about a Topology context,.

`#include <topology.h>`

Data Fields

- unsigned depth
  topology size
- int is_fake
  set if the topology is different from the actual underlying machine

7.8.1 Detailed Description

Global information about a Topology context,.

To be filled with topo_topology_get_info().

7.8.2 Field Documentation

7.8.2.1 unsigned topo_topology_info::depth

topology size

7.8.2.2 int topo_topology_info::is_fake

set if the topology is different from the actual underlying machine

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

- topology.h

Chapter 8

File Documentation

8.1 cpuset.h File Reference

The Cpuset API, for use in libtopology itself.

`#include <topology/config.h>`

`#include <topology/cpuset-bits.h>`

Data Structures

- struct topo_cpuset_t

  Set of CPUs represented as a bitmask.

Defines

- #define TOPO_CPUSET_ZERO (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_ZERO }

  Predefined cpuset with no CPU set.

- #define TOPO_CPUSET_FULL (topo_cpuset_t){ .s[0 ... TOPO_CPUSUBSET_COUNT-1] = TOPO_CPUSUBSET_FULL }

  Predefined cpuset with all CPUs set.

- #define TOPO_CPUSET_CPU(cpu) ({ topo_cpuset_t __set = TOPO_CPUSET_ZERO; TOPO_CPUSUBSET_CPUSUBSET(__set,cpu) = TOPO_CPUSUBSET_VAL(cpu); __set; })

  Predefined cpuset with CPU `cpu` *set.*

- #define TOPO_CPUSET_STRING_LENGTH (TOPO_CPUSET_SUBSTRING_COUNT∗(TOPO_CPUSET_SUBSTRING_LENGTH+1))

  Maximal required length of a string for printing a CPU set.

- #define TOPO_PRIxCPUSET "s"

  Printf format for printing a CPU set.

- #define topo_cpuset_foreach_begin(cpu, set)

  Loop macro iterating on CPU set `set`.

- #define topo_cpuset_foreach_end() }

  End of loop.

Functions

- static __inline__ int topo_cpuset_snprintf (char *__topo_restrict buf, size_t buflen, const topo_cpuset_t *__topo_restrict set)

  Stringify a cpuset.

- static __inline__ void topo_cpuset_from_string (const char *__topo_restrict string, topo_cpuset_t *__topo_restrict set)

  Parse a cpuset string.

- static __inline__ void topo_cpuset_zero (topo_cpuset_t *set)

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

- static __inline__ void topo_cpuset_fill (topo_cpuset_t *set)

  Fill CPU set `set`.

- static __inline__ void topo_cpuset_from_ulong (topo_cpuset_t *set, unsigned long mask)

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

- static __inline__ void topo_cpuset_from_ith_ulong (topo_cpuset_t *set, int i, unsigned long mask)

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

- static __inline__ unsigned long topo_cpuset_to_ulong (const topo_cpuset_t *set)

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

- static __inline__ unsigned long topo_cpuset_to_ith_ulong (const topo_cpuset_t *set, int i)

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

- static __inline__ void topo_cpuset_cpu (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_all_but_cpu (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_set (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ void topo_cpuset_set_range (topo_cpuset_t *set, unsigned begincpu, unsigned endcpu)

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

- static __inline__ void topo_cpuset_clr (topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ int topo_cpuset_isset (const topo_cpuset_t *set, unsigned cpu)

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

- static __inline__ int topo_cpuset_iszero (const topo_cpuset_t *set)

  Test whether set `set` *is zero.*

- static __inline__ int topo_cpuset_isfull (const topo_cpuset_t *set)

  Test whether set `set` *is full.*

- static __inline__ int topo_cpuset_isequal (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_intersects (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_isincluded (const topo_cpuset_t *sub_set, const topo_cpuset_t *super_set)

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

- static __inline__ void topo_cpuset_orset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_andset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_clearset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ void topo_cpuset_xorset (topo_cpuset_t *set, const topo_cpuset_t *modifier_set)

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

- static __inline__ int topo_cpuset_first (const topo_cpuset_t *cpuset)

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

- static __inline__ void topo_cpuset_singlify (topo_cpuset_t *set)

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

- static __inline__ int topo_cpuset_compar_first (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_compar (const topo_cpuset_t *set1, const topo_cpuset_t *set2)

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

- static __inline__ int topo_cpuset_weight (const topo_cpuset_t *set)

  Compute the weight of CPU set `set`.

8.1.1 Detailed Description

The Cpuset API, for use in libtopology itself.

8.2 glibc-sched.h File Reference

Macros to help interaction between libtopolopy and glibc scheduling routines.

```
#include <topology.h>
#include <topology/helper.h>
```

Functions

- static __inline__ void topo_cpuset_to_glibc_sched_affinity (topo_topology_t topology, const topo_cpuset_t *toposet, cpu_set_t *schedset, size_t schedsetsize)

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

- static __inline__ void topo_cpuset_from_glibc_sched_affinity (topo_topology_t topology, topo_cpuset_t *toposet, const cpu_set_t *schedset, size_t schedsetsize)

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

8.2.1 Detailed Description

Macros to help interaction between libtopolopy and glibc scheduling routines.

Applications that use both libtopology and glibc scheduling routines such as sched_getaffinity may want to include this file so as to ease conversion between their respective types.

8.3 helper.h File Reference

High-level libtopology traversal helpers.

Functions

- static __inline__ unsigned topo_get_type_or_below_depth (topo_topology_t topology, topo_obj_type_t type)

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

- static __inline__ unsigned topo_get_type_or_above_depth (topo_topology_t topology, topo_obj_type_t type)

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

- static __inline__ int topo_get_type_nbobjs (topo_topology_t topology, topo_obj_type_t type)

  Returns the width of level type `type`.

- static __inline__ topo_obj_t topo_get_system_obj (topo_topology_t topology)

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

- static __inline__ topo_obj_t topo_get_obj (topo_topology_t topology, topo_obj_type_t type, unsigned index)

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

- static __inline__ topo_obj_t topo_get_next_obj_by_depth (topo_topology_t topology, unsigned depth, topo_obj_t prev)

  Returns the next object at depth `depth`.

- static __inline__ topo_obj_t topo_get_next_obj (topo_topology_t topology, topo_obj_type_t type, topo_obj_t prev)

  Returns the next object of type `type`.

- static __inline__ topo_obj_t topo_get_next_child (topo_topology_t topology, topo_obj_t father, topo_obj_t prev)

  Return the next child.

- static __inline__ topo_obj_t topo_get_common_ancestor_obj (topo_obj_t obj1, topo_obj_t obj2)

  Returns the common father object to objects lvl1 and lvl2.

- static __inline__ int topo_obj_is_in_subtree (topo_obj_t obj, topo_obj_t subtree_root)

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

- static __inline__ topo_obj_t topo_get_next_obj_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_next_obj_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_obj_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, unsigned index)

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

- static __inline__ topo_obj_t topo_get_obj_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, unsigned index)

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

- static __inline__ unsigned topo_get_nbobjs_below_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth)

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

- static __inline__ int topo_get_nbobjs_below_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type)

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

- static topo_obj_t topo_get_cpuset_covering_child (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_t father)

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

- static topo_obj_t topo_get_cpuset_covering_obj (topo_topology_t topology, const topo_cpuset_t *set)

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

- static __inline__ topo_obj_t topo_get_next_obj_above_cpuset_by_depth (topo_topology_t topology, const topo_cpuset_t *set, unsigned depth, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_next_obj_above_cpuset (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_type_t type, topo_obj_t prev)

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

- static __inline__ topo_obj_t topo_get_cpuset_covering_cache (topo_topology_t topology, const topo_cpuset_t *set)

  Get the first cache covering a cpuset `set`.

- static __inline__ topo_obj_t topo_get_shared_cache_above (topo_topology_t topology, topo_obj_t obj)

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

- int topo_get_cpuset_objs (topo_topology_t topology, const topo_cpuset_t *set, topo_obj_t *__topo_restrict objs, int max)

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

- int topo_get_closest_objs (topo_topology_t topology, topo_obj_t src, topo_obj_t *__topo_restrict objs, int max)

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

- static __inline__ void topo_distribute (topo_topology_t topology, topo_obj_t root, topo_cpuset_t *cpuset, int n)

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

8.3.1 Detailed Description

High-level libtopology traversal helpers.

8.4 linux-libnuma.h File Reference

Macros to help interaction between libtopology and Linux libnuma.

```
#include <topology.h>
#include <numa.h>
#include <assert.h>
```

Functions

- static __inline__ void topo_cpuset_to_linux_libnuma_ulongs (topo_topology_t topology, const topo_cpuset_t *cpuset, unsigned long *mask, unsigned long *maxnode)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_ulongs (topo_topology_t topology, topo_cpuset_t *cpuset, const unsigned long *mask, unsigned long maxnode)

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

- static __inline__ struct bitmask * topo_cpuset_to_linux_libnuma_bitmask (topo_topology_t topology, const topo_cpuset_t *cpuset)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_bitmask (topo_topology_t topology, topo_cpuset_t *cpuset, const struct bitmask *bitmask)

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

- static __inline__ void topo_cpuset_to_linux_libnuma_nodemask (topo_topology_t topology, const topo_cpuset_t *cpuset, nodemask_t *nodemask)

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

- static __inline__ void topo_cpuset_from_linux_libnuma_nodemask (topo_topology_t topology, topo_cpuset_t *cpuset, const nodemask_t *nodemask)

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

8.4.1 Detailed Description

Macros to help interaction between libtopology and Linux libnuma.

Applications that use both Linux libnuma and libtopology may want to include this file so as to ease conversion between their respective types.

8.5 topology.doxy File Reference

8.6 topology.h File Reference

The libtopology API.

```
#include <sys/types.h>
#include <stdio.h>
#include <topology/cpuset.h>
#include <topology/helper.h>
```

Data Structures

- struct topo_topology_info
  Global information about a Topology context,.
- struct topo_obj
  Structure of a topology Object.
- union topo_obj_attr_u
  Object type-specific Attributes.
- struct topo_obj_attr_u::topo_cache_attr_u
  Cache-specific Object Attributes.
- struct topo_obj_attr_u::topo_memory_attr_u
  Node-specific Object Attributes.
- struct topo_obj_attr_u::topo_machine_attr_u
- struct topo_obj_attr_u::topo_misc_attr_u
  Misc-specific Object Attributes.

Defines

- #define TOPO_OBJ_TYPE_MAX (TOPO_OBJ_MISC+1)
  Maximal value of an Object Type.
- #define TOPO_TYPE_DEPTH_UNKNOWN -1
  No object of given type exists in the topology.
- #define TOPO_TYPE_DEPTH_MULTIPLE -2
  Objects of given type exist at different depth in the topology.

Typedefs

- typedef struct topo_topology * topo_topology_t
  Topology context.
- typedef struct topo_obj * topo_obj_t

Enumerations

- enum topo_obj_type_t {
  TOPO_OBJ_SYSTEM, TOPO_OBJ_MACHINE, TOPO_OBJ_NODE, TOPO_OBJ_SOCKET,
  TOPO_OBJ_CACHE, TOPO_OBJ_CORE, TOPO_OBJ_PROC, TOPO_OBJ_MISC }
  Type of topology Object.

- enum topo_flags_e { TOPO_FLAGS_WHOLE_SYSTEM = (1<<1) }
  Flags to be set onto a topology context before load.

- enum topo_cpubind_policy_t { TOPO_CPUBIND_PROCESS = (1<<0), TOPO_CPUBIND_-THREAD = (1<<1), TOPO_CPUBIND_STRICT = (1<<2) }
  Process/Thread binding policy.

Functions

- int topo_get_type_order (topo_obj_type_t type)
  Convert an object type into a number that permits to compare them.

- topo_obj_type_t topo_get_order_type (int order)
  Converse of topo_get_type_oder().

- int topo_topology_init (topo_topology_t *topologyp)
  Allocate a topology context.

- int topo_topology_load (topo_topology_t topology)
  Build the actual topology.

- void topo_topology_destroy (topo_topology_t topology)
  Terminate and free a topology context.

- void topo_topology_check (topo_topology_t topology)
  Run internal checks on a topology structure.

- int topo_topology_ignore_type (topo_topology_t topology, topo_obj_type_t type)
  Ignore an object type.

- int topo_topology_ignore_type_keep_structure (topo_topology_t topology, topo_obj_type_t type)
  Ignore an object type if it does not bring any structure.

- int topo_topology_ignore_all_keep_structure (topo_topology_t topology)
  Ignore all objects that do not bring any structure.

- int topo_topology_set_flags (topo_topology_t topology, unsigned long flags)
  Set OR'ed flags to non-yet-loaded topology.

- int topo_topology_set_fsys_root (topo_topology_t __topo_restrict topology, const char *__topo_-restrict fsys_root_path)

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

- int topo_topology_set_synthetic (topo_topology_t __topo_restrict topology, const char *__topo_restrict description)

  Enable synthetic topology.

- int topo_topology_set_xml (topo_topology_t __topo_restrict topology, const char *__topo_restrict xmlpath)

  Enable XML-file based topology.

- int topo_topology_get_info (topo_topology_t __topo_restrict topology, struct topo_topology_info *__topo_restrict info)

  Get additional global information about the topology.

- unsigned topo_get_type_depth (topo_topology_t topology, topo_obj_type_t type)

  Returns the depth of objects of type `type`.

- topo_obj_type_t topo_get_depth_type (topo_topology_t topology, unsigned depth)

  Returns the type of objects at depth `depth`.

- unsigned topo_get_depth_nbobjs (topo_topology_t topology, unsigned depth)

  Returns the width of level at depth `depth`.

- topo_obj_t topo_get_obj_by_depth (topo_topology_t topology, unsigned depth, unsigned index)

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

- const char * topo_obj_type_string (topo_obj_type_t type)

  Return a stringified topology object type.

- topo_obj_type_t topo_obj_type_of_string (const char *string)

  Return an object type from the string.

- int topo_obj_snprintf (char *__topo_restrict string, size_t size, topo_topology_t topology, topo_obj_t obj, const char *__topo_restrict indexprefix, int verbose)

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

- int topo_obj_cpuset_snprintf (char *__topo_restrict str, size_t size, size_t nobj, const topo_obj_t *__topo_restrict objs)

  Stringify the cpuset containing a set of objects.

- int topo_set_cpubind (topo_topology_t topology, const topo_cpuset_t *set, int policy)

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

- int topo_set_proc_cpubind (topo_topology_t topology, topo_pid_t pid, const topo_cpuset_t *set, int policy)

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

- int topo_set_thread_cpubind (topo_topology_t topology, topo_thread_t tid, const topo_cpuset_t *set, int policy)

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

8.6.1 Detailed Description

The libtopology API.

See topology/cpuset.h for CPU set specific macros See topology/helper.h for high-level topology traversal helpers

Index