## NAME

lstopo, lstopo-no-graphics, hwloc-ls − Show the topology of the system

## SYNOPSIS

**lstopo** [ *options* ]... [ *filename* ]

**lstopo-no-graphics** [ *options* ]... [ *filename* ]

**hwloc-ls** [ *options* ]... [ *filename* ]

Note that hwloc(7) provides a detailed explanation of the hwloc system; it should be read before reading this man page

## OPTIONS

**−−of** <format>, **−−output−format** <format>
Enforce the output in the given format. See the OUTPUT FORMATS section below.

**−i** <file>, **−−input** <file>
Read topology from XML file <file> (instead of discovering the topology on the local machine). If <file> is "−", the standard input is used. XML support must have been compiled in to hwloc for this option to be usable.

**−i** <directory>, **−−input** <directory>
Read topology from <directory> instead of discovering the topology of the local machine. On Linux, the directory may contain the topology files gathered from another machine topology with hwloc-gather-topology. On x86, the directory may contain a cpuid dump gathered with hwloc-gather-cpuid.

**−i** <specification>, **−−input** <specification>
Simulate a fake hierarchy (instead of discovering the topology on the local machine). If <specification> is "node:2 pu:3", the topology will contain two NUMA nodes with 3 processing units in each of them. The <specification> string must end with a number of PUs.

**−−if** <format>, **−−input−format** <format>
Enforce the input in the given format, among **xml**, **fsroot**, **cpuid** and **synthetic**.

**−−export−xml−flags** <flags>
Enforce flags when exporting to the XML format. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_topology_export_xml()*. Those names may be substrings of actual flag names as long as a single one matches. A value of **1** (or **v1**) reverts to the format of hwloc v1.x. The default is **0** (or **none**).

**−−export−synthetic−flags** <flags>
Enforce flags when exporting to the synthetic format. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_topology_export_synthetic()*. Those names may be substrings of actual flag names as long as a single one matches. A value of **2** (or **no_attr**) reverts to the format of hwloc v1.9. A value of **3** (or **no_ext,no_attr**) reverts to the original minimalistic format (before v1.9). The default is **0** (or **none**).

**−v −−verbose**
Include additional detail. The hwloc-info tool may be used to display even more information about specific objects.

**−s −−silent**
Reduce the amount of details to show.

**−−distances**
Only display distance matrices.

**−−distances-transform** <links|merge-switch-ports|transitive-closure>
Try applying a transformation to distances structures before displaying them. See hwloc_distances_transform() for details. More transformations may be applied using hwloc-annotate(1) (and it may save their output to XML).

**−−memattrs**
Only display memory attributes. All of them are displayed (while the default textual output selects memory attribute details depending on the verbosity level).

**−−cpukinds**
Only display CPU kinds. CPU kinds are displayed in order, starting from the most energy efficient ones up to the rather higher performance and power hungry ones.

**−−windows−processor−groups**
On Windows, only show information about processor groups. All of them are displayed, while the default verbose output only shows them if there are more than one.

**−f −−force**
If the destination file already exists, overwrite it.

**−l −−logical**
Display hwloc logical indexes of all objects, with prefix "L#". By default, both logical and physical/OS indexes are displayed for PUs and NUMA nodes, logical only for cores, dies and packages, and no index for other types.

**−p −−physical**
Display OS/physical indexes of all objects, with prefix "P#". By default, both logical and physical/OS indexes are displayed for PUs and NUMA nodes, logical only for cores, dies and packages, and no index for other types.

**−c −−cpuset**
Display the cpuset of each object.

**−C −−cpuset−only**
Only display the cpuset of each object; do not display anything else about the object.

**−−taskset**
Show CPU set strings in the format recognized by the taskset command-line program instead of hwloc-specific CPU set string format. This option should be combined with **−−cpuset** or **−−cpuset−only**, otherwise it will imply **−−cpuset**.

**−−only** <type>
Only show objects of the given type in the textual output.

**−−filter** <type>:<kind>, **−−filter** <type>
Filter objects of type <type>, or of any type if <type> is "all". "io", "cache" and "icache" are also supported.

<kind> specifies the filtering behavior. If "none" or not specified, all objects of the given type are removed. If "all", all objects are kept as usual. If "structure", objects are kept when they bring structure to the topology. If "important" (only applicable to I/O), only important objects are kept. See hwloc_topology_set_type_filter() for more details.

hwloc supports filtering any type except PUs and NUMA nodes. lstopo also offers PU and NUMA node filtering by hiding them in the graphical and textual outputs, but any object included in them (for instance Misc) will be hidden as well. Note that PUs and NUMA nodes may not be ignored in the XML output. Note also that the top-level object type cannot be ignored (usually Machine or System).

**−−ignore** <type>
This is the old way to specify **-−filter <type>:none**.

**−−no−smt**
Ignore PUs. This is identical to **-−filter PU:none**.

**−−no−caches**
Do not show caches. This is identical to **-−filter cache:none**.

**−−no−useless−caches**
This is identical to **-−filter cache:structure**.

**−−no−icaches**
This is identical to **-−filter icache:none**.

**−−disallowed**
Include objects disallowed by administrative limitations.

**−−allow** <all|local|0xff|nodeset=0xf0>
Include objects disallowed by administrative limitations (implies **−−disallowed**) and also change the set of allowed ones.

If **local** is given, only objects available to the current process are allowed (default behavior when loading from the native operating system backend). It may be useful if the topology was created by another process (with different administrative restrictions such as Linux Cgroups) and loaded here loaded from XML or synthetic. This case implies **−−thissystem**.

If **all**, all objects are allowed.

If a bitmap is given as a hexadecimal string, it is used as the set of allowed PUs.

If a bitmap is given after prefix **nodeset=**, it is the set of allowed NUMA nodes.

**−−flags** <flags>
Enforce topology flags. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_topology_set_flags()*. Those names may be substrings of actual flag names as long as a single one matches, for instance **disallowed,thissystem_allowed**. The default is **8** (or **import**).

**−−merge**
Do not show levels that do not have a hierarchical impact. This sets HWLOC_TYPE_FILTER_KEEP_STRUCTURE for all object types. This is identical to **−−filter all:structure**.

**−−no−factorize −−no−factorize**=<type>
Never factorize identical objects in the graphical output.

If an object type is given, only factorizing of these objects is disabled. This only applies to normal CPU-side objects, it is independent from PCI collapsing.

**−−factorize −−factorize**=[<type>,]<N>[,<L>[,<F>]
Factorize identical children in the graphical output (enabled by default).

If <N> is specified (4 by default), factorizing only occurs when there are strictly more than N identical children. If <L> and <F> are specified, they set the numbers of first and last children to keep after factorizing.

If an object type is given, only factorizing of these objects is configured. This only applies to normal CPU-side object, it is independent from PCI collapsing.

**−−no−collapse**
Do not collapse identical PCI devices. By default, identical sibling PCI devices (such as many virtual functions inside a single physical device) are collapsed.

**−−no−cpukinds**
Do not show different kinds of CPUs in the graphical output. By default, when supported, different types of lines, thickness and bold font may be used to display PU boxes of different kinds.

**−−restrict** <cpuset>
Restrict the topology to the given cpuset.

**−−restrict** nodeset=<nodeset>
Restrict the topology to the given nodeset, unless --restrict-flags specifies something different.

**−−restrict** binding
Restrict the topology to the current process binding. This option requires the use of the actual current machine topology (or any other topology with **−−thissystem** or with HWLOC_THISSYSTEM set to 1 in the environment).

**−−restrict−flags** <flags>
Enforce flags when restricting the topology. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_topology_restrict()*. Those names may be substrings of actual flag names as long as a single one matches, for instance **bynodeset,memless**. The default is **0** (or **none**).

**−−no−io**
Do not show any I/O device or bridge. This is identical to **−−filter io:none**. By default, common devices (GPUs, NICs, block devices, ...) and interesting bridges/switches are shown.

**−−no−bridges**
Do not show any I/O bridge except hostbridges. This is identical to **−−filter bridge:none**. By default, common devices (GPUs, NICs, block devices, ...) and interesting bridges/switches are shown.

**−−whole−io**
Show all I/O devices and bridges. This is identical to **−−filter io:all**. By default, only common devices (GPUs, NICs, block devices, ...) and interesting bridges/switches are shown.

**−−thissystem**
Assume that the selected backend provides the topology for the system on which we are running. This is useful when loading a custom topology such as an XML file and using **−−restrict binding** or **−−allow all**.

**−−pid** <pid>
Detect topology as seen by process <pid>, i.e. as if process <pid> did the discovery itself. Note that this can for instance change the set of allowed processors. Also show this process current CPU and Memory binding by marking the corresponding PUs and NUMA nodes (in Green in the graphical output, see the COLORS section below, or by appending *(binding)* to the verbose text output). If 0 is given as pid, the current binding for the lstopo process will be shown.

**−−ps −−top**
Show existing processes as misc objects in the output. To avoid uselessly cluttering the output, only processes that are restricted to some part of the machine are shown. On Linux, kernel threads are not shown. If many processes appear, the output may become hard to read anyway, making the hwloc-ps program more practical.

**−−children−order** <order>
Change the order of the different kinds of children with respect to their parent in the graphical output. <*order*> may be a comma-separated list of keywords among:

*memory:above* displays memory children above other children (and above the parent if it is a cache). PUs are therefore below their local NUMA nodes, like hwloc 1.x did.

*io:right* and *misc:right* place I/O or Misc children on the right of CPU children.

*io:below* and *misc:below* place I/O or Misc children below CPU children.

*plain* places everything not specified together with normal CPU children.

If only *plain* is specified, lstopo displays the topology in a basic manner that strictly matches the actual tree: Memory, I/O and Misc children are listed below their parent just like any other child. PUs are therefore on the side of their local NUMA nodes, below a common ancestor. This output may result in strange layouts since the size of Memory, CPU and I/O children may be very different, causing the placement algorithm to poorly arrange them in rows.

The default order is *memory:above,io:right,misc:right* which means Memory children are above CPU children while I/O and Misc are together on the right.

Up to hwloc 2.5, the default was rather to *memory:above,plain*.

Additionally, *io:right*, *io:below*, *misc:right* and *misc:below* may be suffixed with *:horiz*, *:vert* or *:rect* to force the horizontal, vertical or rectangular layout of children inside these sections.

See also the GRAPHICAL OUTPUT and LAYOUT sections below.

**−−fontsize** <size>
Set the size of text font in the graphical output.

The default is 10.

Boxes are scaled according to the text size. The *LSTOPO_TEXT_XSCALE* environment variable may be used to further scale the width of boxes (its default value is 1.0).

The **−−fontsize** option is ignored in the ASCII backend.

**−−gridsize** <size>
Set the margin between elements in the graphical output.

The default is 7. It was 10 prior to hwloc 2.1.

This option is ignored in the ASCII backend.

**−−linespacing** <size>
Set the spacing between lines of text in the graphical output.

The default is 4.

The option was included in **−−gridsize** prior to hwloc 2.1 (and its default was 10).

This option is ignored in the ASCII backend.

**−−thickness** <size>
Set the thickness of lines and boxes in the graphical output.

The default is 1.

This option is ignored in the ASCII backend.

**−−horiz**, **−−horiz**=<type1,...>
Force a horizontal graphical layout instead of nearly 4/3 ratio in the graphical output. If a comma-separated list of object types is given, the layout only applies to the corresponding *container* objects. Ignored for bridges since their children are always vertically aligned.

**−−vert**, **−−vert**=<type1,...>
Force a vertical graphical layout instead of nearly 4/3 ratio in the graphical output. If a comma-separated list of object types is given, the layout only applies to the corresponding *container* objects.

**−−rect**, **−−rect**=<type1,...>
Force a rectangular graphical layout with nearly 4/3 ratio in the graphical output. If a comma-separated list of object types is given, the layout only applies to the corresponding *container* objects. Ignored for bridges since their children are always vertically aligned.

**−−no−text**, **−−no−text**=<type1,...>
Do not display any text in boxes in the graphical output. If a comma-separated list of object types is given, text is disabled for the corresponding objects. This is mostly useful for removing text from Group objects.

**−−text**, **−−text**=<type1,...>
Display text in boxes in the graphical output (default). If a comma-separated list of object types is given, text is reenabled for the corresponding objects (if it was previously disabled with **−−no−text**).

**−−no−index**, **−−no−index**=<type1,...>
Do not show object indexes in the graphical output. If a comma-separated list of object types is given, indexes are disabled for the corresponding objects.

**−−index**, **−−index=**<type1,...>
Show object indexes in the graphical output (default). If a comma-separated list of object types is given, indexes are reenabled for the corresponding objects (if they were previously disabled with **−−no−index**).

**−−no−attrs**, **−−no−attrs**=<type1,...>
Do not show object attributes (such as memory size, cache size, PCI bus ID, PCI link speed, etc.) in the graphical output. If a comma-separated list of object types is given, attributes are disabled for the corresponding objects.

**−−attrs**, **−−attrs=**<type1,...>
Show object attributes (such as memory size, cache size, PCI bus ID, PCI link speed, etc.) in the graphical output (default). If a comma-separated list of object types is given, attributes are reenabled for the corresponding objects (if they were previously disabled with **−−no−attrs**).

**−−no−legend**
Remove all text legend lines at the bottom of the graphical output.

**−−no−default−legend**
Remove default text legend lines at the bottom of the graphical output. User-added legend lines with **−−append−legend or the "lstopoLegend" info are still displayed if any.**

**−−append−legend** <line>
Append the line of text to the bottom of the legend in the graphical output. If adding multiple lines, each line should be given separately by passing this option multiple times. Additional legend lines may also be specified inside the topology using the "lstopoLegend" info attributes on the topology root object.

**−−grey**, **−−greyscale**
Use greyscale instead of colors in the graphical output.

**−−palette** <grey|greyscale|defaut|colors|white|none>
Change the color palette. Passing *grey* or *greyscale* is identical to passing **−−grey** or **−−greyscale**. Passing *white* or *none* uses white instead of colors for all box backgrounds. Passing *default* or *colors* reverts back to the default color palette.

**−−palette** type=#rrggbb
Replace the color of the given box type with the given 3x8bit hexadecimal RGB combination (e.g. *#ff0000* is red). Existing types are *machine*, *group*, *package*, *group_in_package*, *die*, *core*, *pu*, *numanode*, *memories* (box containing multiple memory children), *cache*, *pcidev*, *osdev*, *bridge*, and *misc*.

See also CUSTOM COLOR below for customizing individual objects.

**−−binding−color** <none|#rrggbb>
Do not colorize PUs and NUMA nodes according to the binding in the graphical output. Or change the color to the given 3x8bit hexadecimal RGB combination (e.g. *#ff0000* is red).

**−−disallowed−color** <none|#rrggbb>
Do not colorize disallowed PUs and NUMA nodes in the graphical output. Or change the color to the given 3x8bit hexadecimal RGB combination (e.g. *#00ff00* is green).

**−−top−color** <none|#rrggbb>
Do not colorize task objects in the graphical output when −−top is given. Or change the color to the given 3x8bit hexadecimal RGB combination (e.g. *#0000ff* is blue).

**−−version**
Report version and exit.

**−h −−help**
Display help message and exit.

## DESCRIPTION

lstopo and lstopo-no-graphics are capable of displaying a topological map of the system in a variety of different output formats. The only difference between lstopo and lstopo-no-graphics is that graphical outputs are only supported by lstopo, to reduce dependencies on external libraries. hwloc-ls is identical to lstopo-no-graphics.

The filename specified directly implies the output format that will be used; see the OUTPUT FORMATS section, below. Output formats that support color will indicate specific characteristics about individual CPUs by their color; see the COLORS section, below.

## OUTPUT FORMATS

By default, if no output filename is specific, the output is sent to a graphical window if possible in the current environment (DISPLAY environment variable set on Unix, etc.). Otherwise, a text summary is displayed in the console.

The filename on the command line usually determines the format of the output. There are a few filenames that indicate specific output formats and devices (e.g., a filename of "-" will output a text summary to stdout), but most filenames indicate the desired output format by their suffix (e.g., "topo.png" will output a PNG-format file).

The format of the output may also be changed with "−−of". For instance, "−−of pdf" will generate a PDF-format file on the standard output, while "−−of fig toto" will output a Xfig-format file named "toto".

The list of currently supported formats is given below. Any of them may be used with "−−of" or as a filename suffix.

**default** Send the output to a window or to the console depending on the environment.

**console**
Send a text summary to stdout. Binding or unallowed processors are only annotated in this mode if verbose; see the COLORS section, below.

**ascii** Output an ASCII art representation of the map (formerly called **txt**). If outputting to stdout and if colors are supported on the terminal, the output will be colorized.

**tikz** or **tex**
Output a LaTeX tikzpicture representation of the map that can be compiled with a LaTeX compiler.

**fig** Output a representation of the map that can be loaded in Xfig.

**svg** Output a SVG representation of the map, using Cairo (by default, if supported) or a native SVG backend (fallback, always supported). See **cairosvg** and **nativesvg** below.

**cairosvg** or **svg(cairo)**
If lstopo was compiled with the proper support, output a SVG representation of the map using Cairo.

**nativesvg** or **svg(native)**
Output a SVG representation of the map using the native SVG backend. It may be less pretty than the Cairo output, but it is always supported, and SVG objects have attributes for identifying and manipulating them. See dynamic_SVG_example.html for an example.

**pdf** If lstopo was compiled with the proper support, lstopo outputs a PDF representation of the map.

**ps** If lstopo was compiled with the proper support, lstopo outputs a Postscript representation of the map.

**png** If lstopo was compiled with the proper support, lstopo outputs a PNG representation of the map.

**synthetic**
If the topology is symmetric (which requires that the root object has its symmetric_subtree field set), lstopo outputs a synthetic description string. This output may be reused as an input synthetic topology description later. See also the Synthetic topologies section in the documentation. Note that Misc and I/O devices are ignored during this export.

**xml** If lstopo was compiled with the proper support, lstopo outputs an XML representation of the map. It may be reused later, even on another machine, with lstopo −−input, the HWLOC_XMLFILE environment variable, or the hwloc_topology_set_xml() function.

The following special names may be used:

**−** Send a text summary to stdout.

**/dev/stdout**
Send a text summary to stdout. It is effectively the same as specifying "−".

**−.<format>**
If the entire filename is "−.<format>", lstopo behaves as if "−−of <format> -" was given, which means a file of the given format is sent to the standard output.

See the output of "lstopo −−help" for a specific list of what graphical output formats are supported in your hwloc installation.

## GRAPHICAL OUTPUT

The graphical output is made of nested boxes representing the inclusion of objects in the hierarchy of resources. Usually a Machine box contains one or several Package boxes, that contain multiple Core boxes, with one or several PUs each.

### Caches

Caches are displayed in a slightly different manner because they do not actually include computing resources such as cores. For instance, a L2 Cache shared by a pair of Cores is drawn as a Cache box on top of two Core boxes (instead of having Core boxes inside the Cache box).

### NUMA nodes and Memory-side Caches

By default, NUMA nodes boxes are drawn on top of their local computing resources. For instance, a processor Package containing one NUMA node and four Cores is displayed as a Package box containing the NUMA node box above four Core boxes. If a NUMA node is local to the L3 Cache, the NUMA node is displayed above that Cache box. All this specific drawing strategy for memory objects may be disabled by passing command-line option **−−children−order plain**.

If multiple NUMA nodes are attached to the same parent object, they are displayed inside an additional unnamed memory box.

If some Memory-side Caches exist in front of some NUMA nodes, they are drawn as boxes immediately above them.

### PCI bridges, PCI devices and OS devices

The PCI hierarchy is not drawn as a set of included boxes but rather as a tree of bridges (that may actually be switches) with links between them. The tree starts with a small square on the left for the hostbridge or root complex. It ends with PCI device boxes on the right. Intermediate PCI bridges/switches may appear as additional small squares in the middle.

PCI devices on the right of the tree are boxes containing their PCI bus ID (such as 00:02.3). They may also contain sub-boxes for OS device objects such as a network interface *eth0* or a CUDA GPU *cuda0*.

The datarate of a PCI link may be written (in GB/s) right below its drawn line (if the operating system and/or libraries are able to report that information). This datarate is the currently configured PCI datarate. It may change during execution since some devices are able to slow their PCI links down when idle.

When there is a single link (horizontal line) on the right of a PCI bridge, it means that a single device or bridge is connected on the secondary PCI bus behind that bridge. When there is a vertical line, it means that multiple devices and/or bridges are connected to the same secondary PCI bus.

## LAYOUT

In its graphical output, lstopo uses simple rectangular heuristics to try to achieve a 4/3 ratio between width and height. Although the hierarchy of resources is properly reflected, the exact physical organization (NUMA distances, rings, complete graphs, etc.) is currently ignored.

The layout of a level may be changed with **−−vert**, **−−horiz**, and **−−rect** to force a parent object to arrange its children in vertical, horizontal or rectangular manners respectively.

The position of Memory, I/O and Misc children with respect to other children objects may be changed using **−−children−order**. This effectivement divides children into multiple sections. The layout of children is first computed inside each section, before sections are placed inside (or below) the parent box.

The vertical/horizontal/rectangular layout of these additional sections may also be configured through **−−children−order**.

## COLORS

Individual CPUs and NUMA nodes are colored in the graphical output formats to indicate different characteristics:

| | |
|---|---|
| Green | The topology is reported as seen by a specific process (see **−−pid**), and the given CPU or NUMA node is in this process CPU or Memory binding mask. |
| White | The CPU or NUMA node is in the allowed set (see below). If the topology is reported as seen by a specific process (see **−−pid**), the object is also not in this process binding mask. |
| Red | The CPU or NUMA node is not in the allowed set (see below). |

The "allowed set" is the set of CPUs or NUMA nodes to which the current process is allowed to bind. The allowed set is usually either inherited from the parent process or set by administrative qpolicies on the system. Linux cpusets are one example of limiting the allowed set for a process and its children to be less than the full set of CPUs or NUMA nodes on the system.

Different processes may therefore have different CPUs or NUMA nodes in the allowed set. Hence, invoking lstopo in different contexts and/or as different users may display different colors for the same individual CPUs (e.g., running lstopo in one context may show a specific CPU as red, but running lstopo in a different context may show the same CPU as white).

Some lstopo output modes, e.g. the console mode (default non-graphical output), do not support colors at all. The console mode displays the above characteristics by appending text to each PU line if verbose messages are enabled.

## CUSTOM COLORS

The colors of different kinds of boxes may be configured with **−−palette**.

The color of each object in the graphical output may also be enforced by specifying a "lstopoStyle" info attribute in that object. Its value should be a semi-colon separated list of "<attribute>=#rrggbb" where rr, gg and bb are the RGB components of a color, each between 0 and 255, in hexadecimal (00 to ff). <attribute> may be

**Background**
Sets the background color of the main object box.

**Text** Sets the color of the text showing the object name, type, index, etc.

**Text2** Sets the color of the additional text near the object, for instance the link speed behind a PCI bridge.

The "lstopoStyle" info may be added to a temporarily-saved XML topologies with hwloc-annotate, or with hwloc_obj_add_info(). For instance, to display all core objects in blue (with white names):

```
lstopo save.xml
hwloc-annotate save.xml save.xml core:all info lstopoStyle "Background=#0000ff;Text=#ffffff"
lstopo -i save.xml
```

## EXAMPLES

To display the machine topology in textual mode:

```
lstopo-no-graphics
```

To display the machine topology in ascii-art mode:

```
lstopo-no-graphics -.ascii
```

To display in graphical mode (assuming that the DISPLAY environment variable is set to a relevant value):

```
lstopo
```

To export the topology to a PNG file:

```
lstopo file.png
```

To export an XML file on a machine and later display the corresponding graphical output on another machine:

```
machine1$ lstopo file.xml
<transfer file.xml from machine1 to machine2>
machine2$ lstopo --input file.xml
```

To save the current machine topology to XML and later reload it faster while still considering it as the current machine:

```
$ lstopo file.xml
<...>
$ lstopo --input file.xml --thissystem
```

To restrict an XML topology to only physical processors 0, 1, 4 and 5:

```
lstopo --input file.xml --restrict 0x33 newfile.xml
```

To restrict an XML topology to only numa node whose logical index is 1:

```
lstopo --input file.xml --restrict $(hwloc-calc --input file.xml node:1) newfile.xml
```

To display a summary of the topology:

```
lstopo -s
```

To get more details about the topology:

```
lstopo -v
```

To only show cores:

```
lstopo --only core
```

To show cpusets:

```
lstopo --cpuset
```

To only show the cpusets of package:

```
lstopo --only package --cpuset-only
```

Simulate a fake hierarchy; this example shows with 2 NUMA nodes of 2 processor units:

```
lstopo --input "node:2 2"
```

To count the number of logical processors in the system

```
lstopo --only pu | wc -l
```

To append the kernel release and version to the graphical legend:

```
lstopo --append-legend "Kernel release: $(uname -r)" --append-legend "Kernel version: $(uname -v)"
```

## SEE ALSO

hwloc(7), hwloc-info(1), hwloc-bind(1), hwloc-annotate(1), hwloc-ps(1), hwloc-gather-topology(1), hwloc-gather-cpuid(1)