

NAME

dtc - device tree compiler

SYNOPSIS

dtc [-@fhsv] [-b *boot_cpu_id*] [-d *dependency_file*] [-i *include_path*] [-E [*no-*]*checker_name*]
[-H *phandle_format*] [-I *input_format*] [-O *output_format*] [-o *output_file*] [-R *entries*] [-S *bytes*]
[-p *bytes*] [-V *blob_version*] [-W [*no-*]*checker_name*] [-P *predefined_properties*] *input_file*

DESCRIPTION

The **dtc** utility converts between flattened device tree (FDT) representations. It is most commonly used to generate device tree blobs (DTB), the binary representation of an FDT, from device tree sources (DTS), the ASCII text source representation.

The binary can be written in two formats, binary and assembly. The binary is identical to the in-memory representation and can be used directly by firmware, loaders, and so on. The assembly format, documented in *ASM FORMAT*, will produce the same binary format when assembled, but also includes some global variables that refer to parts of the table. This format is most commonly used to produce a kernel specific to a device, with the device tree blob compiled in.

The options are as follows:

-d *dependency_file*

Writes a dependency file understandable by make to the specified file. This file can be included in a Makefile and will ensure that the output file depends on the input file and any files that it includes. This argument is only useful when the input is DTS, as only the source format has a notion of inclusions.

-i *include_path*

Adds a path to search for include files.

-E [*no-*]*checker_name*

Enable or disable a specified checker. The argument is the name of the checker. The full list of checkers is given in *CHECKERS*.

-@ Emit a `__symbols__` node to allow plugins to be loaded.

-f Force the tool to attempt to generate the output, even if the input had errors.

-h Display the help text and exit.

-H *phandle_format*

Specifies the type of phandle nodes to generate in the output. Valid values are:

linux Generate the legacy linux,phandle nodes expected by older systems.

epapr Generate the phandle nodes, as described in the ePAPR specification. This is the most sensible option for device trees being used with FreeBSD.

both Generate both, for maximum compatibility.

-I *input_format*

Specifies the input format. Valid values are:

dtb Device tree blob. The binary representation of the FDT.

dts Device tree source. The ASCII representation of the FDT. This is the default if the input format is not explicitly stated.

-O *output_format*

Specifies the output format. Valid values are:

asm Assembler source for generating a device tree blob, as described in *ASM FORMAT*.

dtb Device tree blob. The binary representation of the FDT. This is the default if the output format is not explicitly stated.

dts Device tree source. The ASCII representation of the FDT.

-o *output_file*

The file to which to write the output.

-P *predefined_macro*

Defines a macro, in the form *name=value* or *name* to be used for device tree source files that contain conditional components. This tool supports two extensions to the standard to support conditional compilation of device trees. The first is an */include/if [property]/ file.dts* directive that is allowed at the start of a file and which will only include the specified file if the specified property is passed with this flag. The second is the *\$NAME* format for property values. These allow property value to be specified on the command line.

-R *entries*

The number of empty reservation table entries to pad the table with. This is useful if you are generating a device tree blob for bootloader or similar that needs to reserve some memory before passing control to the operating system.

-S *bytes*

The minimum size in bytes of the blob. The blob will be padded after the strings table to ensure that it is the correct size. This is useful for environments where the device tree blob must be modified in place.

-p *bytes*

The number of bytes of padding to add to the blob. The blob will be padded after the strings table to ensure that it is the correct size. This is useful for environments where the device tree blob must be modified in place.

-W [*no-]**checker_name*

Enable or disable a specified checker. This is an alias for **-E**.

-s Sorts the properties and nodes in the tree. This is mainly useful when using tools like `diff(1)` to compare two device tree sources.

-V *output_version*

The version of the format to output. This is only relevant for binary outputs, and only a value of 17 is currently supported.

-v Display the tool version and exit.

input_file

The source file.

ASM FORMAT

The assembly format defines several globals that can be referred to from other compilation units, in addition to any labels specified in the source. These are:

| | |
|-------------------------------|--|
| <code>dt_blob_start</code> | start of the device tree blob. |
| <code>dt_header</code> | start of the header, usually identical to the start of the blob. |
| <code>dt_reserve_map</code> | start of the reservation map. |
| <code>dt_struct_start</code> | start of the structure table. |
| <code>dt_struct_end</code> | end of the structure table. |
| <code>dt_strings_start</code> | start of the strings table. |
| <code>dt_strings_end</code> | end of the strings table. |
| <code>dt_blob_end</code> | end of the device tree blob. |

CHECKERS

The utility provides a number of semantic checks on the correctness of the tree. These can be disabled

with the **-W** flag. For example, **-W no-type-phandle** will disable the phandle type check. The supported checks are:

| | |
|------------------|--|
| type-compatible | Checks the type of the <i>compatible</i> property. |
| type-model | Checks the type of the <i>model</i> property. |
| type-compatible | Checks the type of the <i>compatible</i> property. |
| cells-attributes | Checks that all nodes with children have both <i>#address-cells</i> and <i>#size-cells</i> properties. |
| deleted-nodes | Checks that all <i>/delete-node/</i> statements refer to nodes that are merged. |

OVERLAYS

The utility provides support for generating overlays, also known as plugins. Overlays are a method of patching a base device tree that has been compiled with the **-@** flag, with some limited support for patching device trees that were not compiled with the **-@** flag.

To denote that a DTS is intended to be used as an overlay, */plugin/*; should be included in the header, following any applicable */dts-v1/*; tag.

Conventional overlays are crafted by creating *fragment* nodes in a root. Each fragment node must have either a *target* property set to a label reference, or a *target-path* string property set to a path. It must then have an `__overlay__` child node, whose properties and child nodes are merged into the base device tree when the overlay is applied.

Much simpler syntactic sugar was later invented to simplify generating overlays. Instead of creating targeted fragments manually, one can instead create a root node that targets a label in the base FDT using the *&label* syntax supported in conventional DTS. This will indicate that a fragment should be generated for the node, with the given *label* being the target, and the properties and child nodes will be used as the `__overlay__`.

Additionally, a path-based version of this syntactic sugar is supported. A root node may target a path in the base FDT using a name of the form *&{/path}*. A fragment will be generated for the node as it is in the *&label* case, except the *target-path* property will be set to */path* and no *target* will be set.

Both conventional overlays and the later-added syntactic sugar are supported.

Overlay blobs can be applied at boot time by setting *fdt_overlays* in loader.conf(5). Multiple overlays may be specified, and they will be applied in the order given.

NODE OMISSION

This utility supports the */omit-if-no-ref/* statement to mark nodes for omission if they are ultimately not

referenced elsewhere in the device tree. This may be used in more space-constrained environments to remove nodes that may not be applicable to the specific device the tree is being compiled for.

When the `-@` flag is used to write symbols, nodes with labels will be considered referenced and will not be removed from the tree.

EXAMPLES

The command:

```
dtc -o blob.S -O asm device.dts
```

will generate a *blob.S* file from the device tree source *device.dts* and print errors if any occur during parsing or property checking. The resulting file can be assembled and linked into a binary.

The command:

```
dtc -o - -O dts -I dtb device.dtb
```

will write the device tree source for the device tree blob *device.dtb* to the standard output. This is useful when debugging device trees.

The command:

```
dtc -@ -O dtb -I dts -o device.dtb device.dts
```

will generate a *device.dtb* file from the device tree source *device.dts* with a `__symbols__` node included so that overlays may be applied to it.

The command:

```
dtc -@ -O dtb -I dts -o device_overlay.dtbo device_overlay.dts
```

will generate a *device_overlay.dtbo* file, using the standard extension for a device tree overlay, from the device tree source *device_overlay.dts*. A `__symbols__` node will be included so that overlays may be applied to it. The presence of a `/plugin/;` directive in *device_overlay.dts* will indicate to the utility that it should also generate the underlying metadata required in overlays.

COMPATIBILITY

This utility is intended to be compatible with the device tree compiler provided by elinux.org.

Currently, it implements the subset of features required to build FreeBSD and others that have been

requested by FreeBSD developers.

The *fs* input format is not supported. This builds a tree from a Linux */proc/device-tree*, a file system hierarchy not found in FreeBSD, which instead exposes the DTB directly via a `sysctl`.

The warnings and errors supported by the `elinux.org` tool are not documented. This tool supports the warnings described in the *CHECKERS* section.

SEE ALSO

`fdt(4)`

STANDARDS

The device tree formats understood by this tool conform to the Power.org Standard for Embedded Power Architecture Platform Requirements (*ePAPR*), except as noted in the *BUGS* section and with the following exceptions for compatibility with the `elinux.org` tool:

- The target of cross references is defined to be a node name in the specification, but is in fact a label.

The `/include/` directive is not part of the standard, however it is implemented with the semantics compatible with the `elinux.org` tool. It must appear in the top level of a file, and imports a new root definition. If a file, plus all of its inclusions, contains multiple roots then they are merged. All nodes that are present in the second but not the first are imported. Any that appear in both are recursively merged, with properties from the second replacing those from the first and properties child nodes being recursively merged.

HISTORY

A `dtc` tool first appeared in FreeBSD 9.0. This version of the tool first appeared in FreeBSD 10.0.

AUTHORS

dtc was written by David T. Chisnall. Some features were added later by Kyle Evans.

Note: The fact that the tool and the author share the same initials is entirely coincidental.

BUGS

The device tree compiler does not yet support the following features:

- Labels in the middle of property values. This is only useful in the assembly output, and only vaguely useful there, so is unlikely to be added soon.
- Full paths, rather than labels, as the targets for phandles. This is not very hard to add, but will

probably not be added until something actually needs it.

The current version performs a very limited set of semantic checks on the tree. This will be improved in future versions.