Devicetree

FDT Format

Plumbers 2018
Vancouver, Canada

Frank Rowand, Sony

November 14, 2018
181114_0049

Metadata

Motivation:

- size reduction of FDT and kernel data
- remove metadata from tree name space

side-effects:

- update of FDT format required
- additional features possible, eg
  * phandle as property value, format in decompile
  * delete node
  * delete property
  * validation features

FDT Overlay Metadata

How should the metadata required by overlays be encoded in the FDT?

Discussion was in progress on devicetree-compiler list

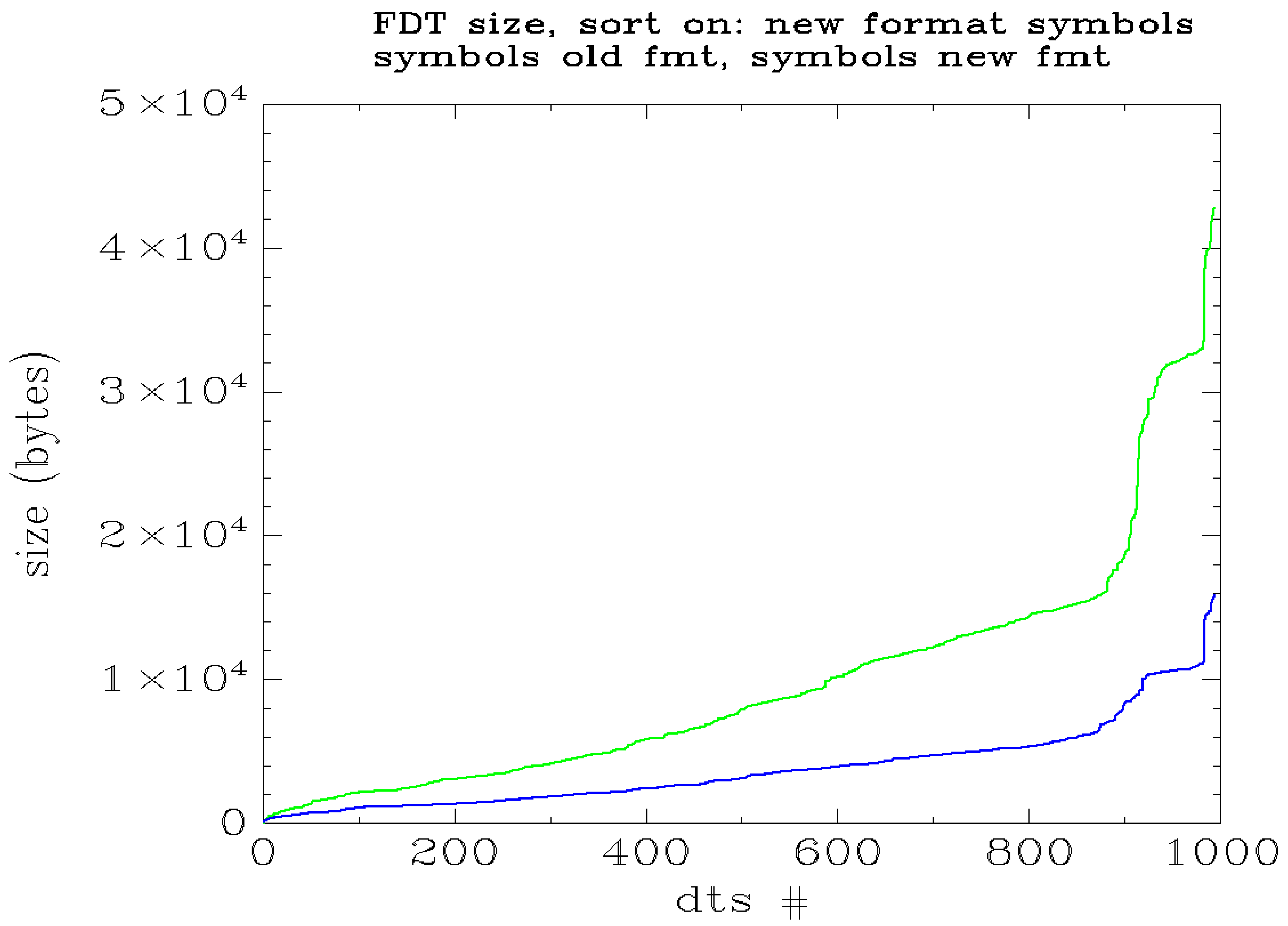
Subject: [RFC] devicetree: new FDT format version
Message-ID: <b96829f9-2e8b-fdc5-5090-58591e2260cf@gmail.com>
Date: Mon, 22 Jan 2018 00:09:18 -0800

Metadata - base FDT overhead

Takeaway:

prototyping showed that the size of base metadata to enable overlay apply can be reduced significantly

(see size slides)


FDT size, sort on: new format symbols
symbols old fmt, symbols new fmt
size (bytes)
dts #
0
1×10^4
2×10^4
3×10^4
4×10^4
5×10^4
0
200
400
600
800
1000

Header format compatibility

Older software can read newer FDT (ignoring new fields)

Newer software can read older FDT (not trying to touch / use new fields)

- I am suggesting a change that breaks compatibility
- We want this to be a RARE event - so do all changes that will cause a break in one shot

Breaking Compatibility Impacts

dtc compiler (and related tools)

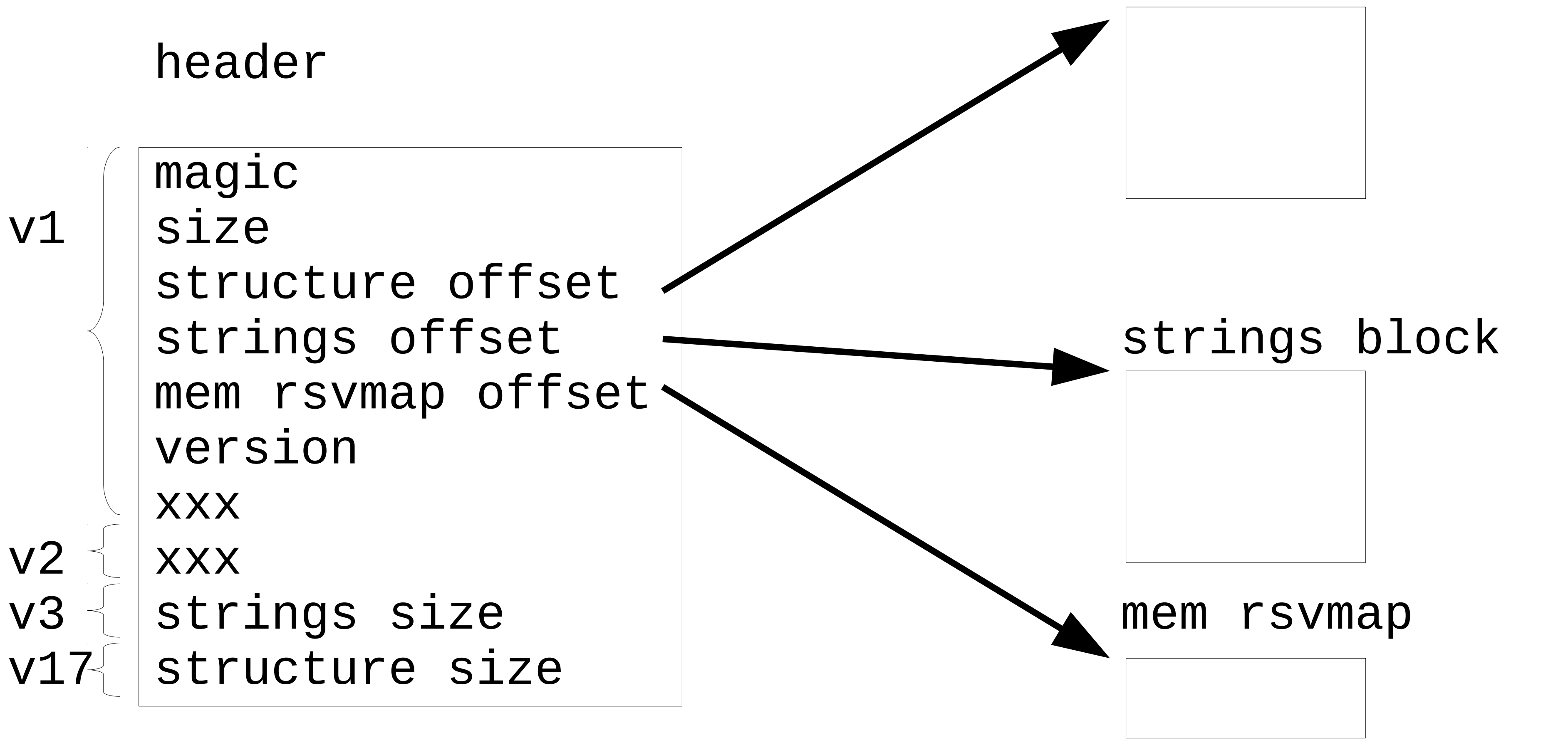
libfdt

boot loader

kernels (Linux, BSD, Zephyr)

Header - can just extend

Or could take opportunity to make more regular



Structure - break compatibility

```
---   source   ---------   ---   tokenized   ----------------------   ---  cells --------

/ {                        BN  0                                      1 0

    nx {                   BN “ny”                                    1 ‘ny’\0\0
         p1 = <1>;         BP val_len name_offset value               3 4 off_p1 1
         p2 = <2 99>;      BP val_len name_offset value value         3 8 off_p2 2 99

         ny {              BN “ny”                                    1 ‘ny’\0\0
              p3 = <3>;    BP val_len name_offset value               3 4 off_p3 3
         };                EN                                         2
    };                     EN                                         2

    nzz {                  BN “nzz”                                   1 ‘nzz’\0
         p4 = <4 14 24>;   BP val_len name_offset value value value   3 12 off_p4 4 14 24
    };                     EN                                         2
};                         EN                                         2
                           EB                                         9
```

```
key:
BN = Begin Node = 1   EN = End Node = 2   BP = Begin Property = 3  EB = End Block = 9
```

```
sequence of 4-byte cells (3 spaces between entries for readability:

1 0   1 ‘ny’\0\0   3 4 off_p1 1   3 8 off_p2 2 99   1 ‘ny’\0\0   3 4 off_p3 3   2   2
1 ‘nzz’\0   3 12 off_p4 4 14 24   2   2   9
```

metadata in tree name space

```
__symbols__ {
        i2c1_pins = "/fragment@0/__overlay__/pinmux_i2c1_pins";
};

__fixups__ {
        am3353x_pinmux = "/fragment@0:target:0";
        i2c1 = "/fragment@1:target:0";
};

__local_fixups__ {
        fragment@1 {
                __overlay__ {
                        pinctrl-0 = <0x0>;
                };
        };
```

__symbols__ replacement

```
__symbols__ {
        i2c1_pins = "/fragment@0/__overlay__/pinmux_i2c1_pins";
};
```

Each entry in the "ext_phandle_use" block is a tuple of:

```
   u32 phandle_value
   u32 symbol_offset
```

The phandle_value contains the value in this FDT of the phandle property in the labeled node whose label name is described by symbol_offset.

The symbol_offset contains the offset within the "dt_strings" block that contains the name of the label corresponding to the node that contains the phandle value.

__fixups__ replacement

```
__fixups__ {
        am3353x_pinmux = "/fragment@0:target:0";
        i2c1 = "/fragment@1:target:0";
};
```

Each entry in the ext_phandle_use block is a tuple of:

```
        u32 prop_value_offset
        u32 symbol_offset
```

The prop_value_offset contains the offset within the "dt_struct" block of the location within a property value that contains a phandle value.

The symbol_offset contains the offset within the "dt_strings" block that contains the name of the label corresponding to the node that contains the referenced phandle value, where the phandle value refers to a node in a different FDT.

__local_fixups__ replacement

```
__local_fixups__ {
        fragment@1 {
                __overlay__ {
                        pinctrl-0 = <0x0>;
                };
        };
```

```
Each entry in the ext_phandle_use block is a single field of:

      u32 prop_value_offset
```

proposed metadata format

advantages:

- less space in FDT, memory
- simpler overlay apply code

disadvantages

- new format has offsets into structure block and strings block, so modifying structure block or strings block may require modifying the metadata blocks (eg, by bootloader)

dgibson’s improvement

Instead of adding new blocks, add new tags to the structure block

FDT_EXTERNAL_PHANDLE with a property offset and strings table offset would replace a __fixups__ entry

FDT_INTERNAL_PHANDLE with just a property offset would replace a __local_fixups__ entry.

They don't need an explicit property reference, because they would just apply to the immediately preceding property.

That approach means we're back to local data, which can be shuffled around pretty easily for inserts and deletes. You'd have to adjust offsets in the fixups for one property when it was altered but not any further away than that.

How to get a copy of the slides

1) frank.rowand@sony.com

2) https://elinux.org/Device_Tree_presentations_papers_articles