Device Tree Tools

What tools exist to support device tree development and debugging?

Where are they?

What new tools have been proposed or requested?

Frank Rowand, Sony Mobile Communications

August 11, 2015

150813_2112

Static Tools

dtdiff

Compare:

- Source (.dts, .dtsi)
- Binary blob (.dtb)
- file system trees (eg EDT)

FDT and EDT are from the target system

FDT is /sys/firmware/fdt

EDT is /proc/device-tree
(currently a link to /sys/firmware/devicetree/base)

dtdiff

Device tree data can be modified by

- build and install
- boot loader
- boot
- running kernel

dtdiff - improvements

Does not properly handle #include and /include/ for .dts and .dtsi files in the normal locations in the Linux kernel source tree.

Work In Progress patch to fix this and to add features such as pre-process single .dts file is at:

http://elinux.org/Device_Tree_frowand

.dtb ---> .dts

A common problem that dtdiff does not solve:

A property is defined (and re-defined) in multiple .dts and .dtsi files.

Which of the many source locations is the one that ends up in the .dtb?

.dtb ---> .dts

current solution:

scan the cpp output, from bottom to top, for the cpp comment that provides the file name

cpp output is available at

`${KBUILD_OUTPUT}/arch/${ARCH}/boot/dts/XXX.dts.dtb.tmp`

for `XXX.dtb`

Incomplete solution:

dtc /include/ directive not processed

DT Source Validation attempts

March 2012, Jon Smirl
http://news.gmane.org/find-root.php?message_id=CAKON4OyW00PUX3-50GrMSa0RhXLHZX3abjQmVHHiYPY2DCN%3dmw@mail.gmail.com

RFC by Benoit Cousson and Fabien Parent
http://thread.gmane.org/gmane.linux.ports.arm.kernel/268685

RFC by Tomasz Figa
http://thread.gmane.org/gmane.linux.ports.arm.kernel/274640

RFC by Stephen Warren
http://thread.gmane.org/gmane.linux.ports.arm.kernel/275896

RFC by Tomasz Figa
http://thread.gmane.org/gmane.comp.devicetree.compiler/56

source: Tomasz Figa's elc 2014

Trees need care: A Solution to Device Tree Validation Problem

http://elinux.org/images/3/35/ELC14-Device_Tree_validation_0.pdf

Kernel Config

What config options are required to enable the drivers and frameworks that are required to use the nodes specified in a device tree?

There are at least three tools to generate a list of config options or a config.

Dynamic Tools

Debugging Boot Problems

Investigating problems with:

- create devices
- register drivers
- bind drivers to devices
- run time modification of the device tree

dt_node_info - What is this tool?

/proc/device-tree and /sys/devices provide visibility into the state and data of

- Flattened Device Tree
- Expanded Device Tree
- Devices

dt_stat	script to probe this information to create various reports
dt_node_info	packages the information from dt_stat in an easy to scan summary

dt_node_info - Where do I get it?

Work In Progress patch is at:

http://elinux.org/Device_Tree_frowand
http://elinux.org/images/a/a3/Dt_stat.patch

Dependency:

requires device tree information to be present in sysfs

Tested:

only on Linux 4.1-rc2, 4.2-rc5 dragonboard

Might work as early as Linux 3.17. Please let me know if it works for you on versions before 4.1.

dt_stat - usage:

```
$ dt_stat --help

usage:
   dt_stat

      -h          synonym for --help
      -help       synonym for --help
      --help      print this message and exit

      --d         report devices
      --n         report nodes
      --nb        report nodes bound to a driver
      --nd        report nodes with a device
      --nxb       report nodes not bound to a driver
      --nxd       report nodes without a device
```

Example - device not created

```
$ dt_node_info coincell
===== devices

===== nodes
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,

===== nodes bound to a driver

===== nodes with a device

===== nodes not bound to a driver
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,

===== nodes without a device
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,
```

Example - driver not bound

```
$ dt_node_info coincell
===== devices
/sys/devices/platform/soc/fc4cf000.spmi/spmi-0/0-00/

===== nodes
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,

===== nodes bound to a driver

===== nodes with a device
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,

===== nodes not bound to a driver
/soc/spmi@fc4cf000/pm8941@0/qcom,coincell@2800 qcom,

===== nodes without a device
```

dyndbg - example 1

Was the driver registered at boot?
On the correct type bus?

----- Target system -----

Kernel command line: debug
dyndbg="func bus_add_driver +p"

$ dmesg | grep "add driver"
bus: 'platform': add driver CCI-400 PMU
bus: 'platform': add driver CCI-400
...

Examples of bus types skip

```
$ dmesg | grep "add driver"
bus: 'platform': add driver gcc-msm8974
bus: 'i2c': add driver dummy
bus: 'mdio_bus': add driver Generic PHY
bus: 'usb': add driver hub
bus: 'qcom_smd': add driver wcnss_ctrl
bus: 'spmi': add driver pmic-spmi
bus: 'scsi': add driver sd
bus: 'spi': add driver m25p80
bus: 'mmc': add driver mmcblk
bus: 'amba': add driver mmci-pl18x
bus: 'hid': add driver hid-generic
```

dyndbg - example 2

Was the driver probe successful at boot?

----- Target system -----

Kernel command line:

dyndbg="func bus_add_driver +p"
dyndbg=”func really_probe +p”

$ dmesg | grep coin

bus: 'platform': add driver qcom,pm8941-coincell
bus: 'platform': really_probe: probing driver qcom,pm8941-coincell
with device fc4cf000.spmi:pm8941@0:qcom,coincell@2800
qcom,pm8941-coincell: probe of fc4cf000.spmi:pm8941@0:qcom,
coincell@2800 failed with error -22

dyndbg - example 3

Deferred probe issues

----- Target system -----

Kernel command line:

```
dyndbg="func deferred_probe_work_func +p"
dyndbg="func driver_deferred_probe_add +p"
dyndbg="func driver_deferred_probe_add +p"
dyndbg="func driver_deferred_probe_del +p"
```

Typical driver binding patterns skip

Make these substitutions on the following slides

- BUS --- the bus name
- DEV --- the device name
- DVR --- the driver name

Device Creation ---> probe skip

create child: NODE
device: 'DEV': device_add
bus: 'BUS': driver_probe_device: matched device DEV with driver DVR
bus: 'BUS': really_probe: probing driver DVR with device DEV

===== messages from driver probe function =====

driver: 'DVR': driver_bound: bound to device 'DEV'
bus: 'BUS': really_probe: bound device DEV to driver DVR

Driver Register ---> probe skip

bus: 'BUS': add driver DVR
bus: 'BUS': driver_probe_device: matched device DEV with driver DVR
bus: 'BUS': really_probe: probing driver DVR with device DEV

===== messages from driver probe function =====

driver: 'DVR': driver_bound: bound to device 'DEV'
bus: 'BUS': really_probe: bound device DEV to driver DVR

Deferred Probe ---> re-probe skip

bus: 'BUS': add driver DVR
device: 'DEV': device_add
bus: 'BUS': driver_probe_device: matched device DEV with DVR
bus: 'BUS': really_probe: probing driver DVR with device DEV

===== messages from driver probe function =====

BUS DEV: Driver DVR requests probe deferral
BUS DEV: Added to deferred list
BUS DEV: Retrying from deferred list
bus: 'BUS': driver_probe_device: matched DEV with driver DVR
bus: 'BUS': really_probe: probing driver DVR with device DEV

===== messages from driver probe function =====

driver: 'DVR': driver_bound: bound to device 'DEV'
bus: 'BUS': really_probe: bound device DEV to driver DVR

Useful data: device and driver

Summary:

dyndbg="func of_platform_bus_create +p"
dyndbg="func bus_add_driver +p"
dyndbg="func device_add +p"
dyndbg="func driver_probe_device +p"
dyndbg="func really_probe +p"
dyndbg="func driver_bound +p"
dyndbg="func deferred_probe_work_func +p"
dyndbg="func driver_deferred_probe_add +p"
dyndbg="func driver_deferred_probe_add +p"
dyndbg="func driver_deferred_probe_del +p"

Properties probed on target vs. contents of .dts

Proof of concept.
Not quite ready for prime time.

```
drivers/of/base.c |   44 ++++++++-
dt_prop           |  249 ++++++++++++++++++...
slash_to_brace.c  |   76 ++++++++++++++++++
3 files changed, 362 insertions(+),
7 deletions(-)
```

Properties probed vs .dts (1/2)

```
--- Properties accessed on the Target
+++ Properties present in the Host .dts
--- console_log_dt_prop
+++ arch/arm/boot/dts/qcom-apq8074-dragonboard.dts
 / {
-       #interrupt-cells;
-       dma-coherent;
-       serial-number;
+        model = <>;
        aliases {
-               compatible;
        };
        chosen {
-               compatible;
        };
        cpu-pmu {
-               assigned-clock-parents;
-               assigned-clock-rates;
-               interrupt-parent;
-               interrupts-extended;
-               pinctrl-0;
-               qcom,no-pc-write;
-               reg;
        };
        cpus {
-               compatible;
+                #size-cells = <0x1>;
+                interrupts = <>;
                cpu@0 {
+                       next-level-cache = <>;
+                       qcom,acc = <>;
                };
```

Properties probed vs .dts (2/2)

```
                idle-states {
-                       compatible;
                        spc {
-                               idle-state-name;
-                               local-timer-stop;
-                               wakeup-latency-us;
                        };
                };
                l2-cache {
+                        cache-level = <>;
+                        qcom,saw = <>;
                };
        };
        memory {
-               compatible;
+                reg = <>;
        };
        soc {
-               #interrupt-cells;
-               clock-ranges;
-               dma-coherent;
-               dma-ranges;
-               interrupt-parent;
                clock-controller@f9088000 {
-                       dma-coherent;
-                       interrupts;
-                       interrupts-extended;
-                       reg-names;
                };
```

Properties probed vs .dts skip

Some false positives.

```
      soc {
-             #interrupt-cells;
-             clock-ranges;
-             dma-coherent;
-             dma-ranges;
-             interrupt-parent;
```

Starting at a child node, OF code chases through parents, then follows interrupt-parent phandle:

```
OF_FND -22 /soc/serial@f991e000 interrupt-parent 0
OF_FND -22 /soc #interrupt-cells 0
OF_FND -22 /soc interrupt-parent 0
OF_FND -22 / #interrupt-cells 0
OF_FND   0 / interrupt-parent 4
OF_FND   0 /soc/interrupt-controller@f9000000 #interrupt-cells 4
```

Driver or Framework Messages

$ dmesg | grep xxx

bus: 'platform': add driver qcom,pm8941-xxx

bus: 'platform': really_probe: probing driver qcom,pm8941-xxx
with device fc4cf000.spmi:pm8941@0:qcom,xxx@2800

qcom,pm8941-xxx: probe of fc4cf000.spmi:pm8941@0:qcom,
xxx@2800 failed with error -22

----- Not so good -----

Can read and find problem in
- .dts and dt bindings document

Can read and/or debug
- driver, subsystem framework, and/or OF source

Driver or Framework Messages

$ dmesg | grep xxx

bus: 'platform': add driver qcom,pm8941-xxx

bus: 'platform': really_probe: probing driver qcom,pm8941-xxx
with device fc4cf000.spmi:pm8941@0:qcom,xxx@2800

qcom,pm8941-xxx:fc4cf000.spmi:pm8941@0:qcom,xxx@2800:
can't find 'qcom,rset-ohms' in DT block

qcom,pm8941-xxx: probe of fc4cf000.spmi:pm8941@0:qcom,
xxx@2800 failed with error -22

----- Good -----

Provides clear description of what to fix

Driver & Framework Messages

If enough information is provided in driver and framework error messages then DT source errors should be solvable without reading the driver source.

What is the state of the typical driver and framework error messages?

Other Random

- fdtdump
- fdtget
- fdtput

.dtb instance version

analogous to kernel version, build #, time, location

eg. /proc/version

Discussion - what can we do?

Current pain points.

Insufficient documentation, examples, tutorials?

What types/classes of problems would be easier to resolve with new tools?

- What tools?

What existing tools need to be improved?

- In what ways do they need improvement?

Resources

http://elinux.org/Device_Tree_frowand

- Tools referenced in this talk
- "Solving Device Tree Issues"
  LinuxCon North America 2015 / Linux Plumbers 2015 refereed track