

Solving Device Tree Issues - part 2

Debugging devicetree issues is painful. Last year I presented some under-development tools and techniques to debug devicetree issues.

This year I will provide an update on the status of those tools and present some new tools and techniques.

Read this later

skip

Any slides with 'skip' in the upper right hand corner will be skipped over in my talk. They contain information that will be useful when the slides are used for reference.

Obligatory Outline

Device tree concepts

Update of Part 1

The Kernel Configuration Problem

The Kernel Configuration Solution

Why this talk?

Debugging device tree problems is not easy.

Why this talk?

Debugging device tree problems is not easy.

- tools do not exist or are not sufficient
- error and warning message may not be available or helpful
- state data is not easy to access and correlate
- debug process is not well documented
- add your own reason here

Why this talk?

At the end of this talk, you will know how to:

- resolve some common device tree issues related to the Linux kernel configuration

Chapter 1

Device tree concepts

why device tree?

A device tree describes hardware that can not be located by probing.

what is device tree?

A device tree is a tree data structure with most nodes describing the devices in a system.

Each node may have properties that contain values.

(based on: ePAPR v1.1)

Key vocabulary

skip

node

- the tree structure
- contain properties and other nodes

property

- contains zero or more data values providing information about a node

Key vocabulary

skip

'compatible' property has pre-defined use

node '/':

- will be used to match a machine_desc entry

other nodes:

- will be used to match a driver
(slight simplification)

.dts - device tree source file

```
/ { /* incomplete .dts example */
  compatible = "qcom,apq8074-dragonboard";

  soc: soc {
    compatible = "simple-bus";
    intc: interrupt-controller@f9000000 {
      compatible = "qcom,msm-qgic2";
      interrupt-controller;
    };

    console: serial@f991e000 {
      compatible = "qcom,msm-uartdm-v1.4", "qcom,msm-uartdm";
      reg = <0xf991e000 0x1000>;
      interrupts = <0 108 0x0>;
    };
  };
};
```

.dts - **Node** – a chunk of HW

```
/ {  
    compatible = "qcom,apq8074-dragonboard";  
  
    soc: soc {  
        compatible = "simple-bus";  
        intc: interrupt-controller@f9000000 {  
            compatible = "qcom,msm-qgic2";  
            interrupt-controller;  
        };  
  
        console: serial@f991e000 {  
            compatible = "qcom,msm-uartdm-v1.4", "qcom,msm-uartdm";  
            reg = <0xf991e000 0x1000>;  
            interrupts = <0 108 0x0>;  
        };  
    };  
};
```

.dts - node compatible property

```
/ { /* incomplete .dts example */
    compatible = "qcom,apq8074-dragonboard";

    soc: soc {
        compatible = "simple-bus";
        intc: interrupt-controller@f9000000 {
            compatible = "qcom,msm-qgic2";
            interrupt-controller;
        };

        console: serial@f991e000 {
            compatible = "qcom,msm-uartdm-v1.4", "qcom,msm-uartdm";
            reg = <0xf991e000 0x1000>;
            interrupts = <0 108 0x0>;
        };
    };
};
```

.dts - node compatible property

A device node's “compatible” property is commonly used to determine the proper driver for the node.

.dts - Reference

skip

Thomas Pettazzoni's ELC 2014 talk “Device Tree For Dummies” is an excellent introduction to device tree source and concepts.

[http://elinux.org/images/f/f9/
Petazzoni-device-tree-dummies_0.pdf](http://elinux.org/images/f/f9/Petazzoni-device-tree-dummies_0.pdf)

<https://www.youtube.com/watch?v=uzBwHFjJ0vU>

More references at

http://elinux.org/Device_Tree_presentations_papers_articles
“introduction to device tree, overviews, and howtos” section

Update of Part 1

Part 1 slides:

http://elinux.org/images/0/04/Dt_debugging_elce_2015_151006_0421.pdf

Supporting material for Part 1:

http://elinux.org/Device_Tree_frowand

The ELCE slides are an updated version of my LinuxCon Japan 2015 slides.

At LinuxCon, I skipped over ~67 slides to fit into one hour.

Update of Part 1

dttdiff

- renamed to scripts/dtc/dtx_diff
- merged in 4.6-rc1

dttdiff was in the ~67 slides that I skipped over at LinuxCon Japan 2015

Update of Part 1

dt_to_config

- vastly improved

- v2 submitted to mainline, might land in 4.8-rc1

Tools that remain proof of concept:

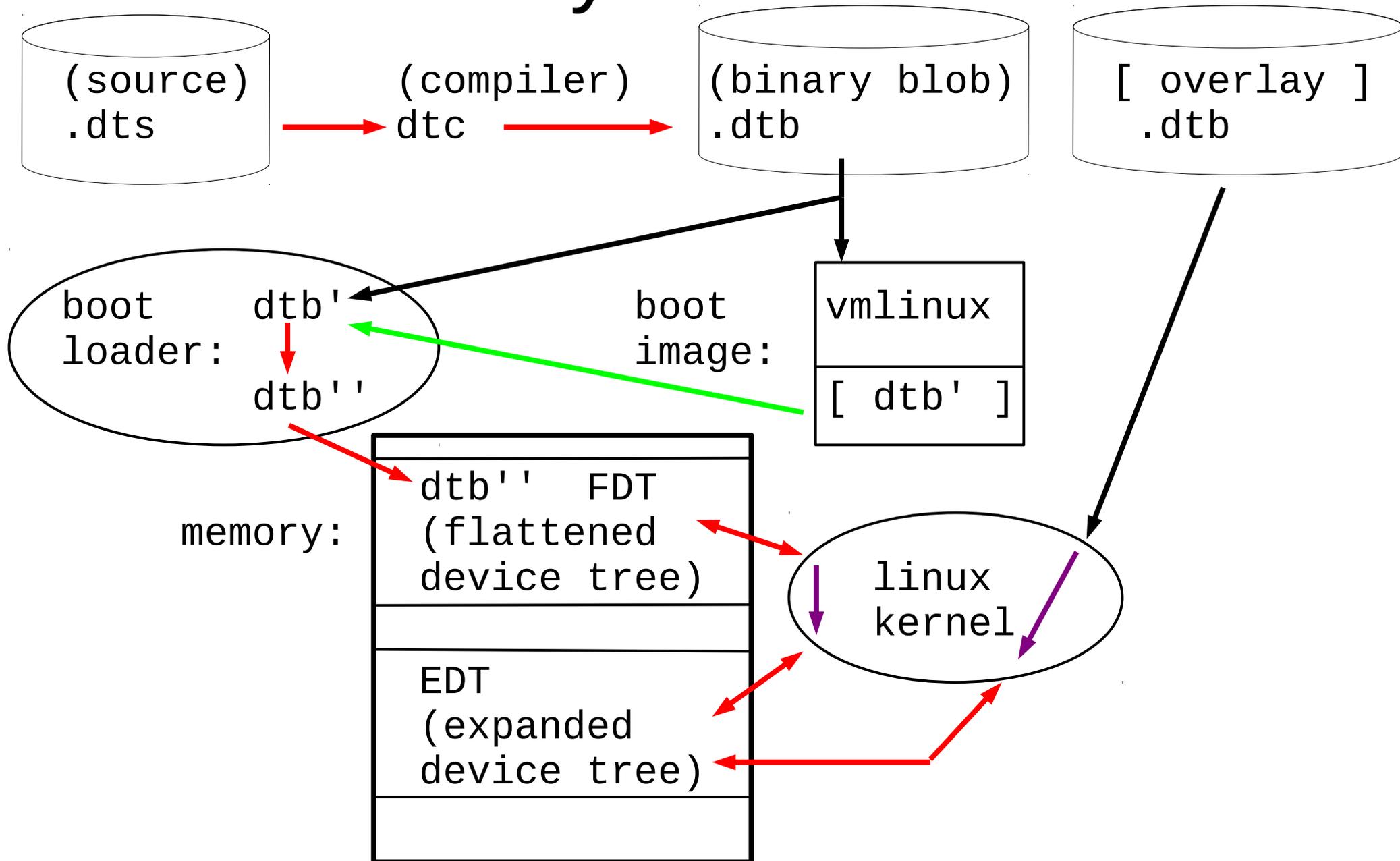
dtc --annotate

dt_node_info

dt_stat

Quick review of Part 1 - dtx_diff

DT data life cycle



DT data life cycle

skip

dtc creates .dtb from .dts

boot loader copies .dtb into memory FDT

Linux kernel reads FDT, creates Expanded DT

.dtb may be modified by
build process
boot loader

FDT and Expanded DT may be modified by
Linux kernel

DT data life cycle

- device tree source
- compiled device tree
- expanded device tree visible in `/proc/device-tree` as a file system tree

dtx_diff - compare two objects

dtx_diff compares device trees in various formats

- source (.dts and the .dtsi includes)
- dtb (binary blob)
- file system tree

dtx_diff - process one .dts

For one source device tree

- pre-process include file directives and create resulting source (that is, converts .dts files and included .dtsi files into a single .dts)

The underlying method uses dtc to compile the source then output the resulting device tree source.

Side effects of this processing include:

- sub-nodes of each node are in sorted order

Takeaway

skip

- There are many ways that a device tree can be changed between the original source and the Extended DT in Linux kernel memory.
- DT includes suggest a different mental model than for C language includes, when investigating
- dtdiff is a powerful tool for investigating changes, but may hide important changes
- In some cases diff is more useful than dtdiff

Chapter 2

The Kernel Configuration Problem

The motivation from last year's presentation

Problem - driver not bound (1)

```
$ 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
```

Problem - driver not bound (1)

Was the driver configured into the kernel?

Device tree node in the .dts file:

```
pm8941_coincell: qcom,coincell@2800 {  
    compatible = "qcom,pm8941-coincell";  
    reg = <0x2800>;  
    status = "disabled";  
};
```

Search for compatible = "qcom,pm8941-coincell"
in the kernel source

Problem - driver not bound (1)

Search for compatible = "qcom,pm8941-coincell" in the kernel source

```
$ git grep "qcom,pm8941-coincell"
```

```
arch/arm/boot/dts/qcom-pm8941.dtsi:                compatible = "qcom,pm8941-coincell", "qcom,pm8941-coincell";
drivers/misc/qcom-coincell.c:    { .compatible = "qcom,pm8941-coincell", },
drivers/misc/qcom-coincell.c:    .name = "qcom,pm8941-coincell"
(END)
```

driver is drivers/misc/qcom-coincell.c

Search drivers/misc/Makefile for the config option to compile the driver

Problem - driver not bound (1)

Search for the config option to compile the driver.

```
$ grep qcom-coincell drivers/misc/Makefile  
obj-$(CONFIG_QCOM_COINCELL) += qcom-coincell.o
```

Problem - driver not bound (1)

Search for the config option to compile the driver. Is it enabled?

```
$ grep qcom-coincell drivers/misc/Makefile  
obj-$(CONFIG_QCOM_COINCELL) += qcom-coincell.o
```

```
$ grep CONFIG_QCOM_COINCELL ${KBUILD_OUTPUT}/.config  
# CONFIG_QCOM_COINCELL is not set
```

Sidetrack

Q. Why is there no tool to generate a list of config options required by a device tree?

A. There have been attempts...
It is not trivial to totally automate.

Sidetrack

Proof of concept tool

```
$ dt_to_config \  
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \  
  > dragon_config_info
```

```
$ grep -i coin dragon_config_info
```

```
# node qcom,coincell@2800 : ok : compatible qcom,pm8941-coincell : driver drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL
```

```
# node qcom,coincell@2800  
: compatible qcom,pm8941-coincell  
: driver drivers/misc/qcom-coincell.c  
: CONFIG_QCOM_COINCELL
```

Fast Forward to today

```
scripts/dtc/dt_to_config \
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \
  --short-name \
  --config ${KBUILD_OUTPUT}/.config \
  > dragon_config_info

grep "qcom,pm8941-coincell" dragon_config_info

-d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL : n
```

Fast Forward to today

```
-d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL : n
```

```
-d-c-----n--F  
: coincell@2800  
: qcom,pm8941-coincell  
: drivers/misc/qcom-coincell.c  
: CONFIG_QCOM_COINCELL  
: n
```

flags, node, compatible, driver, config option, **.config value**

- flags field is new
- checking value of config option in **.config** is new
- field names removed to reduce verbosity

***** WARNING 1a *****

For the rest of this presentation, I will frequently split the output of dt_to_config on the “:” field separator so the information fits on the slide

```
-d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL : n
```

```
-d-c-----n--F  
: coincell@2800  
: qcom,pm8941-coincell  
: drivers/misc/qcom-coincell.c  
: CONFIG_QCOM_COINCELL  
: n
```

***** WARNING 1b *****

For the rest of this presentation, I will frequently remove extraneous fields so the information fits on the slide

```
-d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL : n
```

```
-d-c-----n--F : coincell@2800 : : : CONFIG_QCOM_COINCELL : n
```

***** WARNING 2 *****

For the rest of this presentation, I will show
“dt_to_config” instead of “scripts/dtc/dt_to_config”.

Fast Forward to today

```
dt_to_config \
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \
  --short-name \
  --config ${KBUILD_OUTPUT}/.config \
  --config-format \
  > dragon_config_info
```

```
$ grep "qcom,pm8941-coincell" dragon_config_info
```

```
# -d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM_COINCELL : n
# CONFIG_QCOM_COINCELL is not set
# CONFIG_QCOM_COINCELL=y
```

```
# -d-c-----n--F : coincell@2800 : qcom,pm8941-coincell : .....
# CONFIG_QCOM_COINCELL is not set
# CONFIG_QCOM_COINCELL=y
```

Config fragment available for kernel .config

Review

device tree node “compatible”

==> driver source

==> driver object in makefile

==> kernel CONFIG option from makefile

==> is the CONFIG option enabled?

Disabled node

As a bonus feature, dt_to_config will warn of disabled nodes.

enabled:

```
-d-c- - - -n--F : coincell@2800 :
```

disabled:

```
-d-c- E - - -n--F : coincell@2800 :
```

Review

device tree node “compatible”

==> driver source

==> driver object in makefile

==> kernel CONFIG option from makefile

==> is the CONFIG option enabled?

bonus feature: is the device tree node enabled?

The Kernel Configuration Problem

First the motivation from last year's presentation

- driver missing because configuration option not enabled

The Kernel Configuration Problem

First the motivation from last year's presentation

- driver missing because configuration option not enabled

But there are many ways the configuration can be incorrect

- the wrong driver might be enabled
- multiple drivers might be enabled
- etc

dt_to_config is an aid to fix configuration issues

Disabled node - why “E”?

disabled:

```
-d-c-E---n--F : coincide1@2800 :
```

“E” seems like a strange value for Disabled.

“E” was chosen, because “D” is already used for a different purpose. “E” really stands for problem related to enabled. Upper case letters are used to flag nodes that may have a problem.

The option **--include-suspect** is used to show only nodes that may have a problem.

Flags -- dt_to_config --help

- M** multiple compatibles found for this node
- d** driver found for this compatible
- D** multiple drivers found for this compatible
- c** kernel config found for this driver
- C** multiple config options found for this driver
- E** node is not enabled
- W** compatible is white listed
- H** matching driver and/or kernel config is hard coded
- x** kernel config hard coded in Makefile
- n** one or more kernel config file options is not set
- m** one or more kernel config file options is set to 'm'
- y** one or more kernel config file options is set to 'y'
- F** one of more kernel config file options fails to have correct value

Flags -- oddities

- n one or more kernel config file options is not set
- m one or more kernel config file options is set to 'm'
- y one or more kernel config file options is set to 'y'
- F one of more kernel config file options fails to have correct value

Flags “n” and “m” might indicate that a node has a problem. Why are they lower case?

Suppose that a node requires
`CONFIG_A && !CONFIG_B`

Flags -- oddities

Suppose that a node requires
`CONFIG_A && !CONFIG_B`

“n” would be a problem for `CONFIG_A`, but
not for `CONFIG_B`

“y” would be a problem for `CONFIG_B`, but
not for `CONFIG_A`

The “F” flag was created to indicate that one or
more of the config values did not match the
required value.

Flags - filtering

The values of flags can be used to filter which rows to include in the report.

```
dt_to_config --include-flag C \
  arch/arm/boot/dts/s5pv210-smdkc110.dts \
  --short-name \
  --config ${KBUILD_OUTPUT}/.config \
| grep codec@f1700000

-dDcC----n--F : codec@f1700000 : samsung,mfc-v5 : arch/arm/mach
-dDcC----n--F : codec@f1700000 : samsung,mfc-v5 : arch/arm/mach
```

Flags - filtering

```
dt_to_config --include-flag C \
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_ARCH_EXYNOS :
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_EXYNOS5420_MCPM :

dt_to_config --include-flag CD \
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_ARCH_EXYNOS :
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_EXYNOS5420_MCPM :
-dDc----x---- : : : drivers/media/platform/s5p-mfc/s5p_mfc.c : s5p-mfc-y :
```

C multiple config options found for this driver

D multiple drivers found for this compatible

* node name, compatible, config value fields redacted

Flags - filtering

```
dt_to_config --include-flag C \
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_ARCH_EXYNOS :
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_EXYNOS5420_MCPM :

dt_to_config --include-flag CD \
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_ARCH_EXYNOS :
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_EXYNOS5420_MCPM :
-dDc----x---- : : : drivers/media/platform/s5p-mfc/s5p_mfc.c : s5p-mfc-y :
```

* node name, compatible, config value fields redacted

Filtering on a single upper case flag may lead to missing other important rows for the same node

Flags - filtering

Show only nodes that may have a problem

```
dt_to_config --include-suspect \
  arch/arm/boot/dts/s5pv210-smdkc110.dts \
  --short-name \
  --config ${KBUILD_OUTPUT}/.config \
| grep codec@f1700000

-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_ARCH_EXYNOS :
-dDcC----n--F : : : arch/arm/mach-exynos/exynos.c : CONFIG_EXYNOS5420_MCPM :
-dDc----x---- : : : drivers/media/platform/s5p-mfc/s5p_mfc.c : s5p-mfc-y :
```

* node name, compatible, config value fields redacted

Prevents the single flag filter issue

Flags - filtering

If the report includes all nodes, it is easy to understand where the nodes are in the tree.

The nodes are in the same order as generated by `dtx_diff` (remember that `dtx_diff` sorts nodes)

```
dtx_diff arch/arm/boot/dts/qcom-apq8074-dragonboard.dts
```

```
dt_to_config --short-name \  
arch/arm/boot/dts/qcom-apq8074-dragonboard.dts
```

Flags - filtering

If the report does not include all nodes, it is easy to be confused about with node is referenced.

The same sub-node name may appear in multiple parent nodes.

Node name - full path

If the `--short-name` option is not used then the full path is reported for each node name.

```
dt_to_config \
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \
  --include-suspect \
  --config ${KBUILD_OUTPUT}/.config \
```

```
-d-c-----n--F : /soc/spmi@fc4cf000/pm8941@0/coincell@2800 : qcom,pm8941-coincell : drivers/misc/qcom-coincell.c : CONFIG_QCOM
```

```
: /soc/spmi@fc4cf000/pm8941@1/wled@d800 : : : : n
```

instead of

```
: wled@d800 : : : : n
```

Auto-generate kernel config

```
$ dt_to_config \
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \
  --config ${KBUILD_OUTPUT}/.config \
  --config-format \
  > dragon_config_info
```

Edit dragon_config_info

- Remove the leading “#” for any config option to be enabled (if ...=y) or disabled (if ...=n)

```
$ cat dragon_config_info >> .config
```

```
$ make oldconfig
```

Auto-generate kernel config

This is just one node from the `dt_to_config` command on the previous slide

```
# Md-c-----y- : spc : qcom,idle-state-spc : drivers/soc/qcom/spm.c : CONFIG_QCOM_PM : y
# CONFIG_QCOM_PM is set
# CONFIG_QCOM_PM=y
# MdDc-----y- : spc : arm,idle-state : drivers/cpuidle/cpuidle-arm.c : CONFIG_ARM_CPUIDLE : y
# CONFIG_ARM_CPUIDLE is set
# CONFIG_ARM_CPUIDLE=y
# MdDc-----n--F : spc : arm,idle-state : drivers/cpuidle/cpuidle-big_little.c : CONFIG_ARM_BIG_LITTLE_CPUIDLE : n
# CONFIG_ARM_BIG_LITTLE_CPUIDLE is not set
# CONFIG_ARM_BIG_LITTLE_CPUIDLE=y
```

The command on the previous slide was modified by adding `--short-name` to fit on this slide better

Edit config fragment

```
# Md-c-----y- : spc : qcom,idle-state-spc : drivers/soc/qcom/
# CONFIG_QCOM_PM is set
# CONFIG_QCOM_PM=y
# MdDc-----y- : spc : arm,idle-state : drivers/cpuidle/cpuidl
# CONFIG_ARM_CPUIDLE is set
# CONFIG_ARM_CPUIDLE=y
# MdDc-----n--F : spc : arm,idle-state : drivers/cpuidle/cpuidl
# CONFIG_ARM_BIG_LITTLE_CPUIDLE is not set
# CONFIG_ARM_BIG_LITTLE_CPUIDLE=y
```

- Multiple “compatible” values matched.
- Multiple drivers available.

Need to pick the proper kernel config options to enable the desired driver.

Do not cruise on autopilot

Need to pick the proper kernel config options to enable the desired driver.

dt_to_config does not make decisions that require human judgement.

Human Judgement

Choose among multiple “compatible” property values

“M” flag

Human Judgement

Choose among multiple drivers matching a given “compatible” property

“D” flag

Human Judgement

Choose among multiple kernel config options that can enable a driver

“C” flag

Human Judgement

Find the correct driver and kernel config option when dt_to_config is not smart enough

Some Complications

What else would you expect when working with device tree? :-)

obj-y

Some drivers are enabled in their Makefile with obj-y or other constructs instead of a config option

```
# arch/powerpc/boot/dts/tqm8560.dts
# node ethernet@25000
# compatible gianfar
```

```
-dDc----x---- : : : arch/powerpc/platforms/83xx/misc.c : obj-y :
-dDc----x---- : : : arch/powerpc/platforms/85xx/common.c : obj-y :
-dDc----- : : : arch/powerpc/platforms/85xx/ppa8548.c : CONFIG_PPA8548 :
-dDc----x---- : : : arch/powerpc/platforms/86xx/common.c : obj-y :
-dDc----- : : : drivers/net/ethernet/freescale/fsl_pq_mdio.c : CONFIG_FSL_PQ_MDIO :
-dDc----x---- : : : drivers/net/ethernet/freescale/gianfar.c : gianfar_driver-objs :
```

White List, Black List

Some compatible values, drivers, and other source code other than the driver, and makefiles can result in situations that create poor results from the `dt_to_config` heuristics.

There are white lists and black lists to help resolve these situations.

Some reports are very verbose if the white list and black list are not used.

Solving Device Tree Issues - part 2

Debugging devicetree issues is painful. Last year I presented some under-development tools and techniques to debug devicetree issues.

This year I will provide an update on the status of those tools and present some new tools and techniques.

White List, Black List

Be cautious when using the white and black lists. They are created manually, and there is a possibility that they have are not updated for new kernel versions.

White List, Black List

Be cautious when using the white and black lists. They are created manually, and there is a possibility that they have are not updated for new kernel versions.

If you use the white and black lists, the W and B flags will warn you if the results on a report line depend on the lists.

White List, Black List

```
$ dt_to_config --show-lists
```

These compatibles are hard coded to have no driver.

```
none
pci
simple-bus
```

The driver for these compatibles is hard coded (white list).

```
cache          hardcoded_no_driver
eeprom         hardcoded_no_driver
gpio           hardcoded_no_driver
gpio-keys      drivers/input/keyboard/gpio_keys.c
i2c-gpio       drivers/i2c/busses/i2c-gpio.c
isa            arch/mips/mti-malta/malta-dt.c
               arch/x86/kernel/devicetree.c
led            hardcoded_no_driver
m25p32         hardcoded_no_driver
m25p64         hardcoded_no_driver
m25p80         hardcoded_no_driver
mtd-ram        drivers/mtd/maps/physmap_of.c
pwm-backlight  drivers/video/backlight/pwm_bl.c
spidev         hardcoded_no_driver
syscon         drivers/mfd/syscon.c
tlv320aic23    hardcoded_no_driver
wm8731         hardcoded_no_driver
```

White List, Black List

The configuration option for these drivers is hard coded (white list).

<code>arch/arm/mach-imx/platsmp.c</code>	<code>CONFIG_SOC_IMX6 && CONFIG_SMP</code>
<code>drivers/usb/host/ehci-ppc-of.c</code>	<code>CONFIG_SOC_LS1021A && CONFIG_SMP</code>
<code>drivers/usb/host/ehci-xilinx-of.c</code>	<code>CONFIG_USB_EHCI_HCD</code>
<code>drivers/usb/host/ohci-ppc-of.c</code>	<code>CONFIG_USB_EHCI_HCD_PPC_OF</code>
<code>drivers/usb/host/uhci-platform.c</code>	<code>CONFIG_USB_EHCI_HCD</code>
	<code>CONFIG_USB_EHCI_HCD_XILINX</code>
	<code>CONFIG_USB_OHCI_HCD</code>
	<code>CONFIG_USB_OHCI_HCD_PPC_OF</code>
	<code>CONFIG_USB_UHCI_HCD</code>
	<code>CONFIG_USB_UHCI_PLATFORM</code>

These drivers are black listed.

Compound CONFIG

```
dt_to_config --config-format --config \
  --short-name \
  --config arch/arm/configs/s3c6400_defconfig \
  --config-format \
  arch/arm/boot/dts/armada-xp-lenovo-ix4-300d.dts

: /spi3
: spi-gpio
: arch/arm/mach-s3c64xx/mach-smartq.c
: CONFIG_SAMSUNG_ATAGS && CONFIG_MACH_SMARTQ
: n, n
# CONFIG_SAMSUNG_ATAGS is not set
# CONFIG_SAMSUNG_ATAGS=y
# CONFIG_MACH_SMARTQ is not set
# CONFIG_MACH_SMARTQ=y
```

Compound CONFIG

```
: arch/arm/mach-s3c64xx/mach-smartq.c  
: CONFIG_SAMSUNG_ATAGS && CONFIG_MACH_SMARTQ
```

```
arch/arm/mach-s3c64xx/Makefile:  
ifdef CONFIG_SAMSUNG_ATAGS  
...  
obj-$(CONFIG_MACH_SMARTQ) += mach-smartq.o
```

Tricky Makefiles

```
arch/arm/boot/dts/ls1021a-twr.dts
: dcfg@1ee0000
: fsl,ls1021a-dcfg
: arch/arm/mach-imx/platsmp.c
: CONFIG_SOC_LS1021A && CONFIG_SMP
```

arch/arm/mach-imx/Makefile:

```
ifneq ($(CONFIG_SOC_IMX6)$(CONFIG_SOC_LS1021A),)
obj-$(CONFIG_SMP) += headsmp.o platsmp.o
```

*** dt_to_config misses: CONFIG_SOC_IMX6

dt_to_config can give an incorrect result!

Speed

dt_to_config is not fast

On a fast smp system, with plenty of memory, and the Linux kernel source tree already in memory:

```
time \
dt_to_config \
  arch/arm/boot/dts/qcom-apq8074-dragonboard.dts \
  --config ${KBUILD_OUTPUT}/.config \
  > dragon_config_info
```

```
real 0m3.897s
user 0m10.128s
sys 0m9.124s
```

4 seconds for the command to complete

Speed

`dt_to_config` is not fast

On a fast smp system, with plenty of memory, and the Linux kernel source tree already in memory:

Approximately 1 hour 10 minutes to process every `.dts` file in the 4.7-rc2 Linux kernel source tree, not specifying the `--config` option.

Number of compatibles found

Number of drivers found for each compatible

compatibles: 5843

```
0: 2220 <===== no driver found
1: 2961
2: 432
3: 112
4: 35
5: 20
6: 16
7: 1
8: 39
9: 4
>9: 3
```

Linux 4.7-rc2

Number of compatibles found

Number of drivers found for each compatible

compatibles: 5843

0: 2220 <===== no driver found

dt_to_config is not perfect

Use dt_to_config as an aid

Number of compatibles found skip

Number of drivers found for each compatible

outliers (>9):

11	ibm, ebc
13	ibm, opb
11	lm75

Linux 4.7-rc2

Review

device tree node “compatible”

- ==> driver source

- ==> driver object in makefile

- ==> kernel CONFIG option from makefile

- ==> is the CONFIG option enabled?

bonus feature: is the device tree node enabled?

dt_to_config is an aid to fix configuration issues

Do not trust the output of dt_to_config. Verify the result.

Review - Why this talk?

At the end of this talk, you will know how to:

- resolve some common device tree issues related to the Linux kernel configuration

THE END

Thank you for your attention...

Questions?

Resources

http://elinux.org/Device_Tree_presentations_papers_articles

http://elinux.org/Device_Tree_Reference

devicetree: Kernel Internals and Practical Troubleshooting
Frank Rowand, ELCE 2014

http://elinux.org/ELC_Europe_2014_Presentations

Solving Device Tree Issues - Part 1:

http://elinux.org/images/0/04/Dt_debugging_elce_2015_151006_0421.pdf

Supporting material for: Solving Device Tree Issues - Part 1:

http://elinux.org/Device_Tree_frowand

How to get a copy of the slides

- 1) leave a business card with me
- 2) frank.rowand@am.sony.com
- 3) http://elinux.org/Device_Tree_presentations_papers_articles
- 4) <http://events.linuxfoundation.org>