



# Linux Device Tree 使用指南

版本号: 1.5  
发布日期: 2023.11.15

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.11.16	AWA1440	初始版本。
1.1	2022.05.16	AWA1538	修改排版。
1.2	2022.07.26	AWA1739	更新 Linux-5.10 内核版本 Device Tree 说明。
1.3	2022.07.26	XAA0193	更新了使用范围：兼容 linux-5.10 及其高版本内核。
1.4	2022.12.26	XAA0248	修改 dtsi 文件位置，兼容 longan 和 tina5.0。
1.5	2023.11.15	AWA1727	校对内容，修改错别字、语法错误等。

# 目 录

<b>1 概述</b>	<b>1</b>
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
1.4 相关术语	1
<b>2 模块介绍</b>	<b>2</b>
2.1 Device Tree 描述信息	2
2.2 设备树源码文件及结构关系	2
2.3 Device Tree 结构约定	4
2.3.1 结点名称	4
2.3.2 路径名称	5
2.3.3 属性	5
2.3.3.1 属性名称	6
2.3.3.2 属性值	6
2.4 Device Tree 函数接口说明	6
2.4.1 of_device_is_compatible	6
2.4.2 of_device_is_available	7
2.4.3 of_find_compatible_node	7
2.4.4 of_property_read_u8_array	7
2.4.5 of_property_read_u16_array	8
2.4.6 of_property_read_u32_array	8
2.4.7 of_property_read_string	8
2.4.8 of_remove_property	9
2.4.9 of_get_property	9
2.4.10 of_add_property	9
2.4.11 of_update_property	10

# 1 概述

## 1.1 编写目的

介绍 Device Tree 配置、设备驱动如何获取 Device Tree 配置信息等内容，让用户明确掌握 Device Tree 配置与使用方法。

## 1.2 适用范围

表 1-1: 适用产品列表

内核版本	驱动文件
Linux-5.10 及以上版本	bsp/configs/linux-*/*.dtsti

## 1.3 相关人员

Linux 项目组同事，Linux 内核和驱动开发人员。

## 1.4 相关术语

术语 / 缩略语	解释说明
DTS	Device Tree Source File，设备树源码文件。
DTB	Device Tree Blob File，设备树二进制文件。
DTC	Device Tree Compiler，设备树编译工具。

## 2 模块介绍

### 2.1 Device Tree 描述信息

Device Tree 是一种描述硬件信息的数据结构，它表现为一棵由电路板上 CPU、总线、设备组成的树，Device Tree 由一系列被命名的结点 (node) 和属性 (property) 组成，而结点本身可以包含子结点。所谓属性，就是名字对，也就是成对出现的 name 和 value。在 Device Tree 中，可以描述以下信息：

1. CPU 数量和类别。
2. 内存基址和大小。
3. 总线。
4. 外设。
5. 中断控制器。
6. GPIO 控制器。
7. CLOCK 控制器。

Bootloader 在启动内核之前，会将这棵树通过参数形式传递给内核，内核可以识别这棵树的信息，并根据这些信息展开成 Linux 内核中的 platform device、i2c device、spi device 等设备。而这些设备使用的内存、中断、寄存器、时钟等资源，也会通过 DTB 传递给 Linux 内核，Linux 内核会将这些资源绑定到展开的相应设备上。因此，对 Device Tree 的理解，可以按照以下步骤进行：

1. 用于描述硬件设备信息的文本格式，如 \*.dts\*.dtsi 文件。
2. Linux 内核如何根据 DTB，获取硬件设备信息 (对 Device Tree 的解析)。
3. 设备驱动如何使用。

### 2.2 设备树源码文件及结构关系

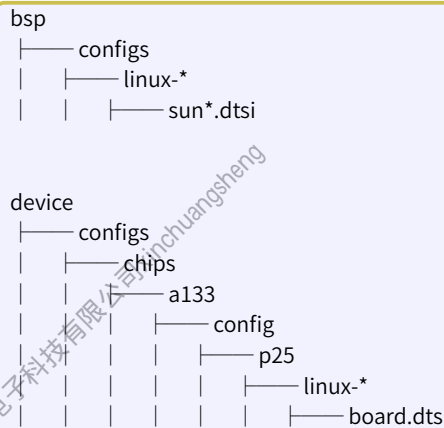
dts 文件是 ASCII 文本格式文件的 Device Tree 描述，在 ARM Linux 中，一个 dts 文件对应一个 ARM 的 machine。由于一个 SoC 可能对应多个 machine，因此 dts 文件可能需要包含许多共同的部分，Linux 内核为了简化，采用了 C 语言包含头文件的方式，将 SoC 公共部分或者多个 machine 共同的部分提炼到 dtsi 文件中，其他 machine 对应的 dts 文件就通过包含方式 include 这个 dtsi 文件即可。

设备树文件的配置是该 SoC 所有方案的通用配置。

对于 Linux-5.10 及其更高内核版本 + BSP 独立方案来说，dts 文件涉及如下两部分：

- 对于 ARM64 CPU 而言，设备树的路径为：bsp/configs/{KERNEL}/{CHIP}.dtsi。
- 板级设备树（board.dts）路径：/device/config/chips/{IC}/configs/{BOARD}/{KERNEL}/board.dts

device tree 的源码结构关系如下：



Device Tree 是一种包含结点和属性的简单树状数据结构，属性就是键-值对 (名字对)，而结点可以同时包含属性和子结点。下图就描述了一棵简单的树。

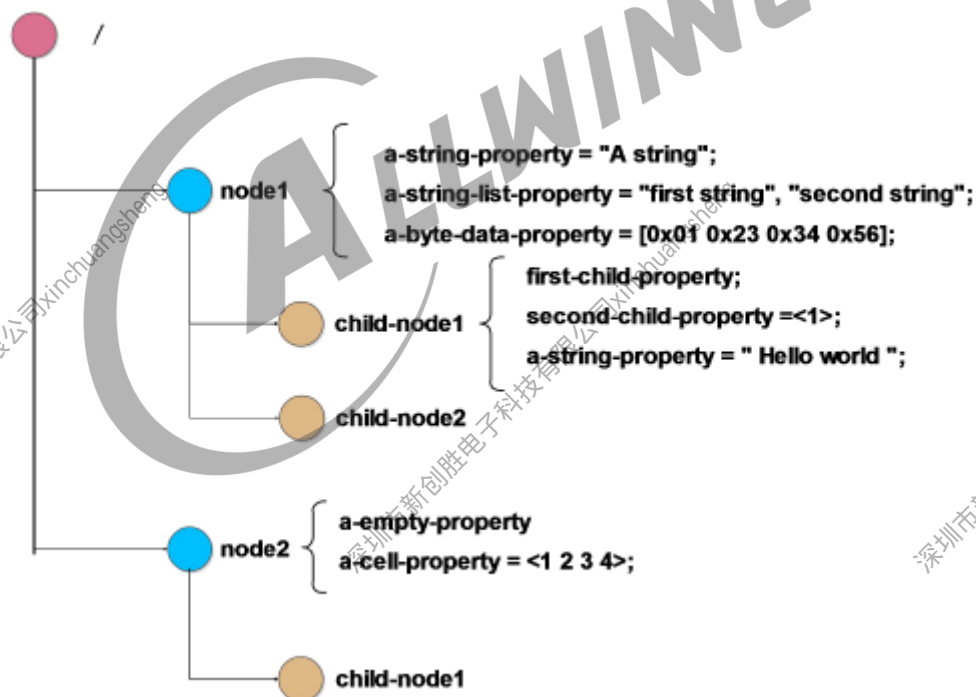


图 2-1: dts 简单树示例

这棵树显然没有什么作用，因为它没有描述任何硬件相关信息，但它确实体现了结点的一些属性，如下：

1. 一个单独的根结点：“/”。

2. 两个子结点: “node1” 和 “node2”。
3. 两个 node1 结点的子结点: “child-node1” 和 “child-node2”。
4. 一些分散在树里的属性。

属性是简单的键-值对，属性的值可以为空或者包含一个任意字节流。虽然数据类型没有编码进数据结构，但在设备树源文件中仍有几个基本的数据类型，如下：

1. 文本字符串，可以用 “” 表示，如:a-string-property= “hello world” ；
2. 数值使用 <> 表示，如:a-cell-property=<1 2 3 4>;
3. 不同表示形式的数据可以使用 “,” 连在一起；
4. “,” 可以用于创建字符串列表:a-string-list-property= “first string” , “second string” 。

## 2.3 Device Tree 结构约定

### 2.3.1 结点名称

规范：Device Tree 中，每个结点的命名遵守以下原则：

node-name@unit-address

1. node-name：结点名称，小于 31 字符长度的字符串，可以包含下表的字符。结点名称首字母必须是英文字母。通常结点名称应该体现设备的类型。
2. @unit-address：如果该结点描述的设备有一个地址，则应该加上设备地址 (unit-address)。通常，设备地址就是用来访问设备的主地址，并且该地址也要在 reg 属性中体现。
3. 同级结点的命名必须是唯一的，但只要地址不一样，多个结点的名称也可以一样，如 (cpu@0 和 cpu@1)。
4. 根结点没有结点名称和设备地址，它是通过 “/” 来识别。

表 2-1: 结点名称支持的字符

字符	描述
0-9	数字
A-Z	大写字母
a-z	小写字母
,	逗号
.	句号
_	下划线
+	加号
-	减号

Device Tree 结点命名规范示例如下图所示：

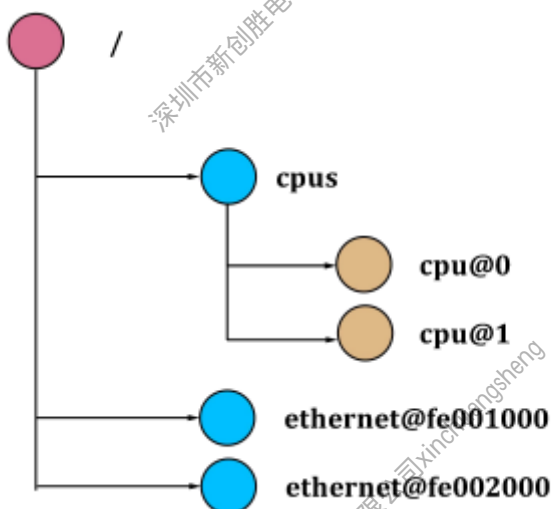


图 2-2: dts 结点命名规范示例

从例子中，可以看到一个根结点 “/” 下有三个子结点，存在两个名称都为 cpu 的结点，但这两个结点可以通过地址 0 和 1 来区分，两个结点名称为 ethernet 的结点，通过地址 fe001000 和 fe002000 来区分。

### 2.3.2 路径名称

在 Device Tree 中，唯一识别结点的另一个方法就是指定绝对路径，即给结点指定从根结点到该结点的完整路径。Device Tree 中约定了完整路径的表达式，如下：

```
/node-name-1/node-name-2/.../node-name-N
```

如，在 dts 结点命名规范示例图中，cpu@1 结点的完整路径如下：

```
/cpus/cpu@1
```

#### 说明

如果完整路径可以明确唯一确认一个结点时，结点之后的地址可以省略。

### 2.3.3 属性

在 Device Tree 中，结点可以用属性来描述该结点的特征，属性由两部分组成：名称和值。



### 2.3.3.1 属性名称

属性名称由长度小于 31 个字符的字符串组成，属性名称支持的字符如结点名称支持的字符表所示。属性名称可以分为标准属性名称和非标准属性名称。非标准属性名称一般需要指定一个唯一的前缀，用来识别是哪家公司或机构定义了该属性，示例如下：

```
fsl,channel-fifo-len 29
ibm,ppc-interrupt-server#s 30
linux,network-index
allwinner,pull = <1>
```

### 2.3.3.2 属性值

属性值是一个包含属性相关信息的数组，数组可能有 0 个或多个字节。当属性是为了传递真伪消息时，属性值可以为空值。这时属性名称的存在与否，就已经足够描述属性的相关信息了，属性值可以采用下表的形式来描述：

表 2-2: 属性值表

值	描述
	空值
u32	大端格式 32bit 整数
u64	大端格式 64bit 整数
string	一个描述 string 信息的数组
phandle	一个值，提供引用结点方法
stringlist	字符串列表

## 2.4 Device Tree 函数接口说明

### 2.4.1 of\_device\_is\_compatible

- 作用：判断设备结点的 compatible 属性是否包含 compat 指定的字符串
- 参数：
  - device：指向需要匹配的设备结点
  - compat：指向需要匹配的字符串
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.2 of\_device\_is\_available

- 作用：判断设备是否存在
- 参数：
  - device：指向需要匹配的设备结点
- 返回：
  - 成功，返回 1
  - 失败，返回 0

## 2.4.3 of\_find\_compatible\_node

- 作用：根据参数 compatible 属性，获取设备结点
- 参数：
  - from：指向开始查找的结点
  - type：指定查找的设备类型，一般为 NULL
  - compatible：指定匹配的属性
- 返回：
  - 成功，返回设备结点指针
  - 失败，返回错误码

## 2.4.4 of\_property\_read\_u8\_array

- 作用：读取指定设备结点的指定属性，存放到 u8 类型的数组
- 参数：
  - np：指向需要读取的设备结点
  - propname：指向要读取的属性
  - out\_values：指向要存放数据的数组
  - sz：读取数据的大小
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.5 of\_property\_read\_u16\_array

- 作用：读取指定设备结点的指定属性，存放到 u16 类型的数组
- 参数：
  - np：指向需要读取的设备结点
  - propname：指向要读取的属性
  - out\_values：指向要存放数据的数组
  - sz：读取数据的大小
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.6 of\_property\_read\_u32\_array

- 作用：读取指定设备结点的指定属性，存放到 u32 类型的数组
- 参数：
  - np：指向需要读取的设备结点
  - propname：指向要读取的属性
  - out\_values：指向要存放数据的数组
  - sz：读取数据的大小
- 返回：
  - 成功，返回 0。
  - 失败，返回错误码

## 2.4.7 of\_property\_read\_string

- 作用：读取指定设备结点的指定属性中的字符串值
- 参数：
  - np：指向需要读取的设备结点
  - propname：指向要读取的属性
  - out\_string：指向要存放字符串的地址
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.8 of\_remove\_property

- 作用：删除指定设备结点的指定属性
- 参数：
  - np：指向需要删除属性的设备结点
  - prop：指向要删除的属性
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.9 of\_get\_property

- 作用：获取指定设备结点的指定属性
- 参数：
  - np：指向需要获取属性的设备结点
  - name：指向要获取的属性名
  - lenp：获取属性的大小
- 返回：
  - 成功，返回指向属性的指针
  - 失败，返回错误码

## 2.4.10 of\_add\_property

- 作用：添加指定设备结点的指定属性
- 参数：
  - np：指向需要添加属性的设备结点
  - prop：指向要添加的属性
- 返回：
  - 成功，返回 0
  - 失败，返回错误码

## 2.4.11 of\_update\_property

- 作用：更新指定设备结点的指定属性
- 参数：
  - np：指向需要更新属性的设备结点
  - prop：指向要更新到设备结点的属性
- 返回：
  - 成功，返回 0
  - 失败，返回错误码




## 著作权声明

版权所有 © 2023 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

、 **全志科技** （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标、产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。