Embed Linux

Learn Note

haobo.gao

ZhengZhou

September 26, 2019

Contents

Chapter 1

Device Tree

1.1 在 kernel 启动时

在 kernel 的 main.c 中的 setup_arch，在设置 cpu 后，进行设备树的设置，如果 setup_machine_fdt 返回 NULL 则会去尝试解析 tags.

```
mdesc = setup_machine_fdt(__atags_pointer);
if (!mdesc)
  mdesc = setup_machine_tags(__atags_pointer, __machine_arch_type);
    machine_desc = mdesc;
  machine_name = mdesc->name;
```

接下来的一些文字我希望分析启动过程中和 device tree 相关的事情。现在看来是要和 setup_machine_fdt 耗上了。

1.1.1 setup_machine_fdt

```
    /**
     * setup_machine_fdt - Machine setup when an dtb was passed to the kernel
     * @dt_phys: physical address of dt blob
     *
     * If a dtb was passed to the kernel in r2, then use it to choose the
     * correct machine_desc and to setup the system.
     */
    const struct machine_desc * __init setup_machine_fdt(unsigned int dt_phys)
    {
        const struct machine_desc *mdesc, *mdesc_best = NULL;

    #ifdef CONFIG_ARCH_MULTIPLATFORM
        DT_MACHINE_START(GENERIC_DT, "Generic DT based system")
        MACHINE_END

        mdesc_best = &__mach_desc_GENERIC_DT;
    #endif

        if (!dt_phys || !early_init_dt_verify(phys_to_virt(dt_phys)))
            return NULL;

        mdesc = of_flat_dt_match_machine(mdesc_best, arch_get_next_mach);
```

```
23
24          if (!mdesc) {
25              const char *prop;
26              int size;
27              unsigned long dt_root;
28
29              early_print("\nError: unrecognized/unsupported "
30                      "device tree compatible list:\n[ ");
31
32              dt_root = of_get_flat_dt_root();
33              prop = of_get_flat_dt_prop(dt_root, "compatible", &size);
34              while (size > 0) {
35                  early_print("'%s' ", prop);
36                  size -= strlen(prop) + 1;
37                  prop += strlen(prop) + 1;
38              }
39              early_print("]\n\n");
40
41              dump_machine_table(); /* does not return */
42          }
43
44          /* We really don't want to do this, but sometimes firmware provides buggy data
                */
45          if (mdesc->dt_fixup)
46              mdesc->dt_fixup();
47
48          early_init_dt_scan_nodes();
49
50          /* Change machine number to match the mdesc we're using */
51          __machine_arch_type = mdesc->nr;
52
53          return mdesc;
54      }
```

从注释上看这个函数是要在 dtb 传入内核的时候，来根据 dtb 做一些相应设置的。

在一开始，如果发现 dt_phys 为空，即没有 dtb 传递进来，那么，直接返回 NULL，返回后会又 tags 的方法去用 tags 的方法进行设置。

```
1  early_init_dt_verify(phys_to_virt(dt_phys)
2
3  bool __init early_init_dt_verify(void *params)
4  {
5    if (!params)
6      return false;
7
8    /* check device tree validity */
9    if (fdt_check_header(params))
10     return false;
11
12   /* Setup flat device-tree pointer */
13   initial_boot_params = params;
14   of_fdt_crc32 = crc32_be(~0, initial_boot_params,
15         fdt_totalsize(initial_boot_params));
16   return true;
17 }
```

上面的 code 先验证 device tree 是不是有效。然后用 dtb 的虚拟地址赋值为 initial_boot_params

.

接下来：

```
1
2  static const void * __init arch_get_next_mach(const char *const **match)
3  {
4    static const struct machine_desc *mdesc = __arch_info_begin;
5    const struct machine_desc *m = mdesc;
6
7    if (m >= __arch_info_end)
8      return NULL;
9
10   mdesc++;
11   *match = m->dt_compat;
12   return m;
13 }
14
15
16 mdesc = of_flat_dt_match_machine(mdesc_best, arch_get_next_mach);
```

of_flat_dt_match_machine 从表中找到匹配的 machine . 传入的参数是:

- default_match: 这个位置传入的是 mdesc_best 如果没有在匹配表中找到匹配的, 就会用这个返回。
- get_next_compat : 传入的是 arch_get_next_mach 回调函数, 这个回调函数用来找到下一个合适的匹配表。

arch_get_next_mach 中 __arch_info_begin 到 __arch_info_end 依次返回 machine_desc 的 dt_compat 成员。那么这个区段究竟存了些什么, 谁存的呢？接下来就要把这件事情搞得很明白。

在链接脚本中, 存在这样的区段：

```
1     .init.arch.info : {
2         __arch_info_begin = .;
3         *(.arch.info.init)
4         __arch_info_end = .;
5     }
```

从这里可以得知，为了进行接下来的研究，需要找到把 attribute 设置为在 section ".arch.info.init" 的代码。

而这段代码为:

```
1     arch/arm/include/asm/mach/arch.h
2
3     /*
4  * Set of macros to define architecture features.  This is built into
5  * a table by the linker.
6  */
7  #define MACHINE_START(_type,_name)      \
8  static const struct machine_desc __mach_desc_##_type  \
9   __used             \
10  __attribute__((__section__(".arch.info.init"))) = {  \
11   .nr    = MACH_TYPE_##_type,     \
```

```
  .name   = _name,

#define MACHINE_END       \
};

#define DT_MACHINE_START(_name, _namestr)   \
static const struct machine_desc __mach_desc_##_name  \
 __used            \
 __attribute__((__section__(".arch.info.init"))) = {  \
  .nr   = ~0,       \
  .name   = _namestr,

#endif
```

在 i.mx6ull 中有：

```
    arch/arm/mach-imx/mach-imx6ul.c :

    static const char *imx6ul_dt_compat[] __initconst = {
        "fsl,imx6ul",
        "fsl,imx6ull",
        NULL,
    };

    DT_MACHINE_START(IMX6UL, "Freescale i.MX6 Ultralite (Device Tree)")
        .map_io    = imx6ul_map_io,
        .init_irq = imx6ul_init_irq,
        .init_machine = imx6ul_init_machine,
        .init_late   = imx6ul_init_late,
        .dt_compat   = imx6ul_dt_compat,
    MACHINE_END
```

看到这些后，我们返回来分析 of_flat_dt_match_machine 中的逻辑。dt_root 代表了设备树的根节点。接下来，依次从.arch.info.init 段中，取出 machine_desc 结构，然后提取出其中的 dt_compat 成员和传入的 dtb 中的比对。

具体的匹配过程：

```

    /**
    * of_fdt_match - Return true if node matches a list of compatible values
    */
   int of_fdt_match(const void *blob, unsigned long node,
                    const char *const *compat)
   {
       unsigned int tmp, score = 0;

       if (!compat)
           return 0;

       while (*compat) {
           tmp = of_fdt_is_compatible(blob, node, *compat);
           if (tmp && (score == 0 || (tmp < score)))
               score = tmp;
           compat++;
       }

       return score;
   }
```

```
22
23 /**
24  * of_flat_dt_match - Return true if node matches a list of compatible values
25  */
26 int __init of_flat_dt_match(unsigned long node, const char *const *compat)
27 {
28   return of_fdt_match(initial_boot_params, node, compat);
29 }
30
31 int of_fdt_is_compatible(const void *blob,
32           unsigned long node, const char *compat)
33 {
34   const char *cp;
35   int cplen;
36   unsigned long l, score = 0;
37
38   cp = fdt_getprop(blob, node, "compatible", &cplen);
39   if (cp == NULL)
40     return 0;
41   while (cplen > 0) {
42     score++;
43     if (of_compat_cmp(cp, compat, strlen(compat)) == 0)
44       return score;
45     l = strlen(cp) + 1;
46     cp += l;
47     cplen -= l;
48   }
49
50   return 0;
51 }
```

还记得 initial_boot_params 吗？不记得请 initial_boot_params 1.1.1 , of_flat_dt_match 根据 compat 从跟节点判断 dtb 中的 compatible 字段是否最合适。这里判断是不是合适使用了 score 的方法，当完全匹配时，score 为 1 并返回; 否则就是字符串部分匹配，返回非零值。