u-boot 命令处理机制

以前在 u-boot 中添加自定命令时，非常惊讶在 u-boot 中添加自定义命令是如此的简单。只需要使用 u-boot 提供的一个宏定义，就可以将自己写好的命令添加到 u-boot 中。对于一些高级语言，比如说 java 可能通过反射的机制很容易实现。但是对于 C 这种语言，要达到如此高的灵活度，实在是无法想象。由于当时比较忙，对于细节也没有深入研究。趁如今比较闲，深入了解一下。

如何在 u-boot 中添加命令

在 u-boot 的 common 下，建立自己的命令文件，如 cmd_test.c。

```
#include <common.h>
#include <command.h>

int do_test(cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
	printf("just a test \n");
	return 0;
}

U_BOOT_CMD(

	dfu, 1 ,	1,	do_test,
	"enter the dfu mode.",
	"\n	enter the dfu mode . "
);
```

只需要将自己定义的要实现的功能 int do_test(cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])用改格式封装好(至于问什么要如此封装，与 cmd_tbl_t 的结构有关)，并使用定义 U_BOOT_CMD 宏将 do_test 方法封装一下就好了。当然还需要修改 common 下的 Makefile 文件，将 cmd_test.c 加入编译方可。

实现分析

先看一下 U_BOOT_CMD 的定义，在 common.h 中。U_BOOT_CMD

的定义如下：

```
#define U_BOOT_CMD(name,maxargs,rep,cmd,usage,help) \
cmd_tbl_t __u_boot_cmd_##name Struct_Section = {#name, maxargs, rep, cmd, usage, help}
```

其中 Struct_Secion 的定义如下：

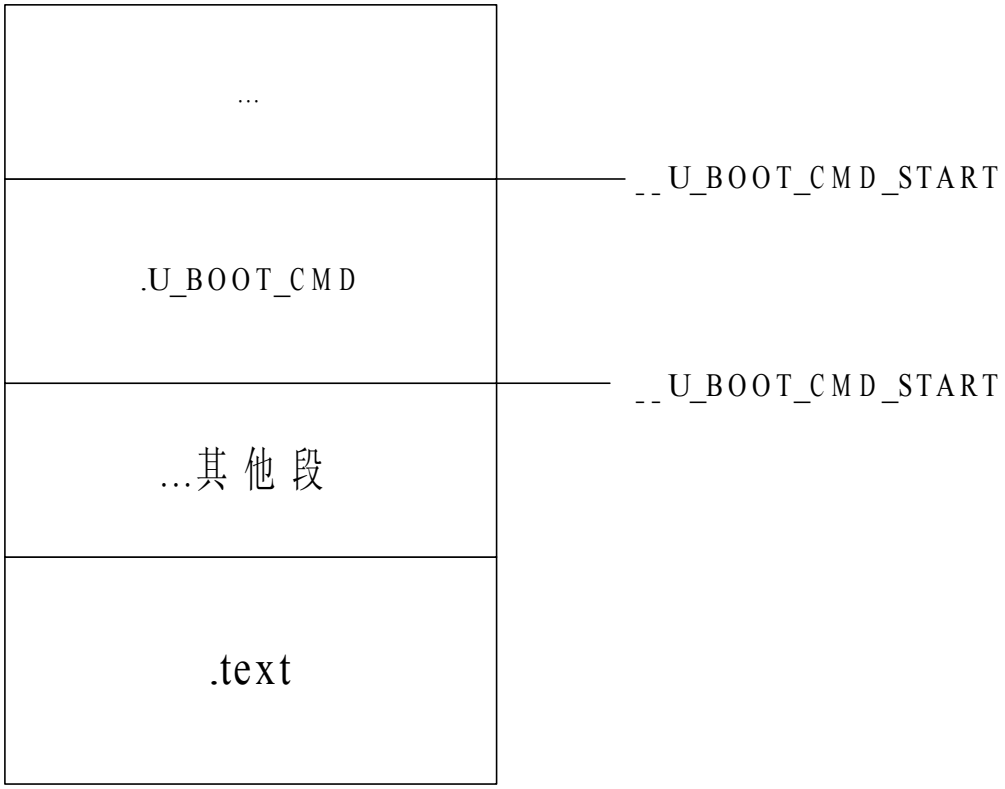
```
#define Struct_Section    __attribute__ ((unused,section (".u_boot_cmd")))
```

由此可见，被 U_BOOT_CMD 定义过的结构体，最终回被放到一个 u_boot_cmd 段中。可以通过 readef 工具进行验证，会发现在生成的目标文件中，确实多了一个.u_boot_mcd 段。

再看一下 u-boot 的链接脚本，其中有这么一段:

```
.u_boot_cmd : {
    __u_boot_cmd_start = .;
    *(.u_boot_cmd)
    __u_boot_cmd_end = .;
 }
 uboot_end_data = .;
```

可以发现该脚本将所有的.u_boot_cmd 放在了一起。而起还定义了两个常量__u_boot_cmd_start 和__u_boot_cmd_end 还表示所有命令的起始位置和结束位置。最后只需对该段进行遍历就可以得到所有的命令了。最后生成的 u-boot 可执行文件中会包含如下的一段:



获取所有 U_BOOT_CMD 的命令如下:

```
cmd_tbl_t *find_cmd (const char *cmd)
{
	int len = &__u_boot_cmd_end - &__u_boot_cmd_start;
	return find_cmd_tbl(cmd, &__u_boot_cmd_start, len);
}
cmd_tbl_t *find_cmd_tbl (const char *cmd, cmd_tbl_t *table, int table_len)
{
	cmd_tbl_t *cmdtp_temp = table;	/*Init value */

	for (cmdtp = table;
		cmdtp != table + table_len;
		cmdtp++) {
		....
	}
}
```