驱动加载过程

先看几个宏：

xref/kernel-4.19/include/linux/module.h#87

[87](http://localhost:8080/source/xref/kernel-4.19/include/linux/module.h" \l "87)  #**define** **[module\_init](http://localhost:8080/source/s?refs=module_init&project=kernel-4.19)**(**[x](http://localhost:8080/source/s?refs=x&project=kernel-4.19)**) [\_\_initcall](http://localhost:8080/source/s?defs=__initcall&project=kernel-4.19)([x](http://localhost:8080/source/s?defs=x&project=kernel-4.19));

xref/kernel-4.19/include/linux/init.h#199

199 #define \_\_define\_initcall(fn, id) \_\_\_define\_initcall(fn, id, .initcall##id)

215 #define pure\_initcall(fn) \_\_define\_initcall(fn, 0)

217 #define core\_initcall(fn) \_\_define\_initcall(fn, 1)

218 #define core\_initcall\_sync(fn) \_\_define\_initcall(fn, 1s)

219 #define postcore\_initcall(fn) \_\_define\_initcall(fn, 2)

220 #define postcore\_initcall\_sync(fn) \_\_define\_initcall(fn, 2s)

221 #define arch\_initcall(fn) \_\_define\_initcall(fn, 3)

222 #define arch\_initcall\_sync(fn) \_\_define\_initcall(fn, 3s)

223 #define subsys\_initcall(fn) \_\_define\_initcall(fn, 4)

224 #define subsys\_initcall\_sync(fn) \_\_define\_initcall(fn, 4s)

225 #define fs\_initcall(fn) \_\_define\_initcall(fn, 5)

226 #define fs\_initcall\_sync(fn) \_\_define\_initcall(fn, 5s)

227 #define rootfs\_initcall(fn) \_\_define\_initcall(fn, rootfs)

228 #define device\_initcall(fn) \_\_define\_initcall(fn, 6)

229 #define device\_initcall\_sync(fn) \_\_define\_initcall(fn, 6s)

230 #define late\_initcall(fn) \_\_define\_initcall(fn, 7)

231 #define late\_initcall\_sync(fn) \_\_define\_initcall(fn, 7s)

233 #define \_\_initcall(fn) device\_initcall(fn)

从上述宏定义中可以指知道，我们通常使用的module\_init的装载优先级是6。系统根据优先级的顺序装载所有的驱动程序，数值越小，优先级越高。

当bootloader装载并进入kernel后执行的第一个方法是start\_kernel。

xref/kernel-4.19/init/main.c#530

[530](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "530)  [asmlinkage](http://localhost:8080/source/s?defs=asmlinkage&project=kernel-4.19) [\_\_visible](http://localhost:8080/source/s?defs=__visible&project=kernel-4.19) **void** [\_\_init](http://localhost:8080/source/s?defs=__init&project=kernel-4.19) **[start\_kernel](http://localhost:8080/source/s?refs=start_kernel&project=kernel-4.19)**(**void**)

......

[736](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "736)   **[rest\_init](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "rest_init)**();

[396](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "396)  **static** [noinline](http://localhost:8080/source/s?defs=noinline&project=kernel-4.19) **void** [\_\_ref](http://localhost:8080/source/s?defs=__ref&project=kernel-4.19) **[rest\_init](http://localhost:8080/source/s?refs=rest_init&project=kernel-4.19)**(**void**)

......

[407](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "407)   **[pid](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "pid)** = [kernel\_thread](http://localhost:8080/source/s?defs=kernel_thread&project=kernel-4.19)(**[kernel\_init](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "kernel_init)**, [NULL](http://localhost:8080/source/s?defs=NULL&project=kernel-4.19), [CLONE\_FS](http://localhost:8080/source/s?defs=CLONE_FS&project=kernel-4.19));

[1058](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "1058)  **static** **int** [\_\_ref](http://localhost:8080/source/s?defs=__ref&project=kernel-4.19) **[kernel\_init](http://localhost:8080/source/s?refs=kernel_init&project=kernel-4.19)**(**void** \***[unused](http://localhost:8080/source/s?refs=unused&project=kernel-4.19)**)

......

[1062](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "1062)   **[kernel\_init\_freeable](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "kernel_init_freeable)**();

[1112](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "1112)  **static** [noinline](http://localhost:8080/source/s?defs=noinline&project=kernel-4.19) **void** [\_\_init](http://localhost:8080/source/s?defs=__init&project=kernel-4.19) **[kernel\_init\_freeable](http://localhost:8080/source/s?refs=kernel_init_freeable&project=kernel-4.19)**(**void**)

......

[1145](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "1145)   **[do\_basic\_setup](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "do_basic_setup)**();

[970](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "970)  **static** **void** [\_\_init](http://localhost:8080/source/s?defs=__init&project=kernel-4.19) **[do\_basic\_setup](http://localhost:8080/source/s?refs=do_basic_setup&project=kernel-4.19)**(**void**)

......

[978](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "978)   **[do\_initcalls](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "do_initcalls)**();

[955](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "955)  **static** **void** [\_\_init](http://localhost:8080/source/s?defs=__init&project=kernel-4.19) **[do\_initcalls](http://localhost:8080/source/s?refs=do_initcalls&project=kernel-4.19)**(**void**)

......

[960](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "960)   **[do\_initcall\_level](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "do_initcall_level)**([level](http://localhost:8080/source/s?defs=level&project=kernel-4.19));

[939](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "939)  **static** **void** [\_\_init](http://localhost:8080/source/s?defs=__init&project=kernel-4.19) **[do\_initcall\_level](http://localhost:8080/source/s?refs=do_initcall_level&project=kernel-4.19)**(**int** **[level](http://localhost:8080/source/s?refs=level&project=kernel-4.19)**)

......

[951](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "951)  **for** ([fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19) = **[initcall\_levels](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "initcall_levels)**[[level](http://localhost:8080/source/s?defs=level&project=kernel-4.19)]; [fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19) < **[initcall\_levels](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "initcall_levels)**[[level](http://localhost:8080/source/s?defs=level&project=kernel-4.19)+1]; [fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)++)

[952](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "952)   **[do\_one\_initcall](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "do_one_initcall)**([initcall\_from\_entry](http://localhost:8080/source/s?defs=initcall_from_entry&project=kernel-4.19)([fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)));

[874](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "874)  **int** [\_\_init\_or\_module](http://localhost:8080/source/s?defs=__init_or_module&project=kernel-4.19) **[do\_one\_initcall](http://localhost:8080/source/s?refs=do_one_initcall&project=kernel-4.19)**([initcall\_t](http://localhost:8080/source/s?defs=initcall_t&project=kernel-4.19) **[fn](http://localhost:8080/source/s?refs=fn&project=kernel-4.19)**)

......

[884](http://localhost:8080/source/xref/kernel-4.19/init/main.c" \l "884)   [ret](http://localhost:8080/source/s?defs=ret&project=kernel-4.19) = [fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)();

上述[fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)就是module\_init([fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19))中的入参。

接下来看看[fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)的值是如何确定的。查看上述[initcall\_from\_entry](http://localhost:8080/source/s?defs=initcall_from_entry&project=kernel-4.19)的定义：

915 static initcall\_entry\_t \***initcall\_levels**[] \_\_initdata = {

916 \_\_initcall0\_start,

917 \_\_initcall1\_start,

918 \_\_initcall2\_start,

919 \_\_initcall3\_start,

920 \_\_initcall4\_start,

921 \_\_initcall5\_start,

922 \_\_initcall6\_start,

923 \_\_initcall7\_start,

924 \_\_initcall\_end,

925 };

[122](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "122)  **static** **inline** **[initcall\_t](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "initcall_t)** **[initcall\_from\_entry](http://localhost:8080/source/s?refs=initcall_from_entry&project=kernel-4.19)**([initcall\_entry\_t](http://localhost:8080/source/s?defs=initcall_entry_t&project=kernel-4.19) \***[entry](http://localhost:8080/source/s?refs=entry&project=kernel-4.19)**)

[123](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "123)  {

[124](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "124)   **return** [offset\_to\_ptr](http://localhost:8080/source/s?defs=offset_to_ptr&project=kernel-4.19)([entry](http://localhost:8080/source/s?defs=entry&project=kernel-4.19));

[125](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "125)  }

xref/kernel-4.19/include/linux/compiler.h?fi=offset\_to\_ptr#372

368 /\*\*

369 \* offset\_to\_ptr - convert a relative memory offset to an absolute pointer

370 \* @off: the address of the 32-bit offset value

371 \*/

372 static inline void \*offset\_to\_ptr(const int \*off)

373 {

374 return (void \*)((unsigned long)off + \*off);

375 }

上述过程是内核启动过程中查找fn在内存中的绝对地址的过程。

那么fn的绝对地址是如何确定的？在执行make命令时，先将源码编译成.o，然后由ld链接器再根据链接描述文件lds，对多个.o文件进行组装。

xref/kernel-4.19/arch/x86/kernel/vmlinux.lds#82

[.init.data](http://localhost:8080/source/s?path=.init.data&project=kernel-4.19) : AT(ADDR([.init.data](http://localhost:8080/source/s?path=.init.data&project=kernel-4.19)) - 0xffffffff80000000) { KEEP(\*(SORT(\_\_\_kentry+\*))) \*([.init.data](http://localhost:8080/source/s?path=.init.data&project=kernel-4.19) [init.data.](http://localhost:8080/source/s?path=init.data.&project=kernel-4.19)\*) \*([.meminit.data](http://localhost:8080/source/s?path=.meminit.data&project=kernel-4.19)\*) \*([.init.rodata](http://localhost:8080/source/s?path=.init.rodata&project=kernel-4.19) [.init.rodata.](http://localhost:8080/source/s?path=.init.rodata.&project=kernel-4.19)\*) . = ALIGN(8); \_\_start\_ftrace\_events = .; KEEP(\*(\_ftrace\_events)) \_\_stop\_ftrace\_events = .; \_\_start\_ftrace\_eval\_maps = .; KEEP(\*(\_ftrace\_eval\_map)) \_\_stop\_ftrace\_eval\_maps = .; . = ALIGN(8); \_\_start\_kprobe\_blacklist = .; KEEP(\*(\_kprobe\_blacklist)) \_\_stop\_kprobe\_blacklist = .; . = ALIGN(32); \_\_start\_error\_injection\_whitelist = .; KEEP(\*(\_error\_injection\_whitelist)) \_\_stop\_error\_injection\_whitelist = .; \*([.meminit.rodata](http://localhost:8080/source/s?path=.meminit.rodata&project=kernel-4.19)) . = ALIGN(8); \_\_clk\_of\_table = .; KEEP(\*(\_\_clk\_of\_table)) KEEP(\*(\_\_clk\_of\_table\_end)) . = ALIGN(8); \_\_cpu\_method\_of\_table = .; KEEP(\*(\_\_cpu\_method\_of\_table)) KEEP(\*(\_\_cpu\_method\_of\_table\_end)) . = ALIGN(8); \_\_cpuidle\_method\_of\_table = .; KEEP(\*(\_\_cpuidle\_method\_of\_table)) KEEP(\*(\_\_cpuidle\_method\_of\_table\_end)) . = ALIGN(32); \_\_dtb\_start = .; KEEP(\*([.dtb.init.rodata](http://localhost:8080/source/s?path=.dtb.init.rodata&project=kernel-4.19))) \_\_dtb\_end = .; . = ALIGN(8); \_\_irqchip\_acpi\_probe\_table = .; KEEP(\*(\_\_irqchip\_acpi\_probe\_table)) \_\_irqchip\_acpi\_probe\_table\_end = .; . = ALIGN(8); \_\_timer\_acpi\_probe\_table = .; KEEP(\*(\_\_timer\_acpi\_probe\_table)) \_\_timer\_acpi\_probe\_table\_end = .; . = ALIGN(8); \_\_earlycon\_table = .; KEEP(\*(\_\_earlycon\_table)) \_\_earlycon\_table\_end = .; . = ALIGN(16); \_\_setup\_start = .; KEEP(\*([.init.setup](http://localhost:8080/source/s?path=.init.setup&project=kernel-4.19))) \_\_setup\_end = .; \_\_initcall\_start = .; KEEP(\*([.initcallearly.init](http://localhost:8080/source/s?path=.initcallearly.init&project=kernel-4.19))) \_\_initcall0\_start = .; KEEP(\*([.initcall0.init](http://localhost:8080/source/s?path=.initcall0.init&project=kernel-4.19))) KEEP(\*([.initcall0s.init](http://localhost:8080/source/s?path=.initcall0s.init&project=kernel-4.19))) \_\_initcall1\_start = .; KEEP(\*([.initcall1.init](http://localhost:8080/source/s?path=.initcall1.init&project=kernel-4.19))) KEEP(\*([.initcall1s.init](http://localhost:8080/source/s?path=.initcall1s.init&project=kernel-4.19))) \_\_initcall2\_start = .; KEEP(\*([.initcall2.init](http://localhost:8080/source/s?path=.initcall2.init&project=kernel-4.19))) KEEP(\*([.initcall2s.init](http://localhost:8080/source/s?path=.initcall2s.init&project=kernel-4.19))) \_\_initcall3\_start = .; KEEP(\*([.initcall3.init](http://localhost:8080/source/s?path=.initcall3.init&project=kernel-4.19))) KEEP(\*([.initcall3s.init](http://localhost:8080/source/s?path=.initcall3s.init&project=kernel-4.19))) \_\_initcall4\_start = .; KEEP(\*([.initcall4.init](http://localhost:8080/source/s?path=.initcall4.init&project=kernel-4.19))) KEEP(\*([.initcall4s.init](http://localhost:8080/source/s?path=.initcall4s.init&project=kernel-4.19))) \_\_initcall5\_start = .; KEEP(\*([.initcall5.init](http://localhost:8080/source/s?path=.initcall5.init&project=kernel-4.19))) KEEP(\*([.initcall5s.init](http://localhost:8080/source/s?path=.initcall5s.init&project=kernel-4.19))) \_\_initcallrootfs\_start = .; KEEP(\*([.initcallrootfs.init](http://localhost:8080/source/s?path=.initcallrootfs.init&project=kernel-4.19))) KEEP(\*([.initcallrootfss.init](http://localhost:8080/source/s?path=.initcallrootfss.init&project=kernel-4.19))) \_\_initcall6\_start = .; KEEP(\*([.initcall6.init](http://localhost:8080/source/s?path=.initcall6.init&project=kernel-4.19))) KEEP(\*([.initcall6s.init](http://localhost:8080/source/s?path=.initcall6s.init&project=kernel-4.19))) \_\_initcall7\_start = .; KEEP(\*([.initcall7.init](http://localhost:8080/source/s?path=.initcall7.init&project=kernel-4.19))) KEEP(\*([.initcall7s.init](http://localhost:8080/source/s?path=.initcall7s.init&project=kernel-4.19))) \_\_initcall\_end = .; \_\_con\_initcall\_start = .; KEEP(\*([.con\_initcall.init](http://localhost:8080/source/s?path=.con_initcall.init&project=kernel-4.19))) \_\_con\_initcall\_end = .; \_\_security\_initcall\_start = .; KEEP(\*([.security\_initcall.init](http://localhost:8080/source/s?path=.security_initcall.init&project=kernel-4.19))) \_\_security\_initcall\_end = .; . = ALIGN(4); \_\_initramfs\_start = .; KEEP(\*([.init.ramfs](http://localhost:8080/source/s?path=.init.ramfs&project=kernel-4.19))) . = ALIGN(8); KEEP(\*([.init.ramfs.info](http://localhost:8080/source/s?path=.init.ramfs.info&project=kernel-4.19))) }

编译log显示链接过程如下：

ld -m elf\_x86\_64 -z noreloc-overflow -pie --no-dynamic-linker -T arch/x86/boot/compressed/vmlinux.lds arch/x86/boot/compressed/head\_64.o arch/x86/boot/compressed/misc.o arch/x86/boot/compressed/string.o arch/x86/boot/compressed/cmdline.o arch/x86/boot/compressed/error.o arch/x86/boot/compressed/piggy.o arch/x86/boot/compressed/cpuflags.o arch/x86/boot/compressed/early\_serial\_console.o arch/x86/boot/compressed/kaslr.o arch/x86/boot/compressed/kaslr\_64.o arch/x86/boot/compressed/mem\_encrypt.o arch/x86/boot/compressed/pgtable\_64.o arch/x86/boot/compressed/eboot.o arch/x86/boot/compressed/efi\_stub\_64.o drivers/firmware/efi/libstub/lib.a arch/x86/boot/compressed/efi\_thunk\_64.o -o arch/x86/boot/compressed/vmlinux

lds文件相当于ELF文件的组织框架，将如下宏展开后就将fn这个符号填充到了上面[.initcall6.init](http://localhost:8080/source/s?path=.initcall6.init&project=kernel-4.19)这个位置。

#**define** **[\_\_\_define\_initcall](http://localhost:8080/source/s?refs=___define_initcall&project=kernel-4.19)**(**[fn](http://localhost:8080/source/s?refs=fn&project=kernel-4.19)**, **[id](http://localhost:8080/source/s?refs=id&project=kernel-4.19)**, **[\_\_sec](http://localhost:8080/source/s?refs=__sec&project=kernel-4.19)**) **static** **[initcall\_t](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "initcall_t)** [\_\_initcall\_](http://localhost:8080/source/s?defs=__initcall_&project=kernel-4.19)##[fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19)##[id](http://localhost:8080/source/s?defs=id&project=kernel-4.19) [\_\_used](http://localhost:8080/source/s?defs=__used&project=kernel-4.19) \

[\_\_attribute\_\_](http://localhost:8080/source/s?defs=__attribute__&project=kernel-4.19)(([\_\_section\_\_](http://localhost:8080/source/s?defs=__section__&project=kernel-4.19)(#[\_\_sec](http://localhost:8080/source/s?defs=__sec&project=kernel-4.19) ".init"))) = [fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19);

#**define** **[\_\_define\_initcall](http://localhost:8080/source/s?refs=__define_initcall&project=kernel-4.19)**(**[fn](http://localhost:8080/source/s?refs=fn&project=kernel-4.19)**, **[id](http://localhost:8080/source/s?refs=id&project=kernel-4.19)**) [\_\_\_define\_initcall](http://localhost:8080/source/s?defs=___define_initcall&project=kernel-4.19)([fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19), [id](http://localhost:8080/source/s?defs=id&project=kernel-4.19), .[initcall](http://localhost:8080/source/s?defs=initcall&project=kernel-4.19)##[id](http://localhost:8080/source/s?defs=id&project=kernel-4.19))

#**define** **[device\_initcall](http://localhost:8080/source/s?refs=device_initcall&project=kernel-4.19)**(**[fn](http://localhost:8080/source/s?refs=fn&project=kernel-4.19)**) **[\_\_define\_initcall](http://localhost:8080/source/xref/kernel-4.19/include/linux/init.h" \l "__define_initcall)**([fn](http://localhost:8080/source/s?defs=fn&project=kernel-4.19), 6)

上面[\_\_section\_\_](http://localhost:8080/source/s?defs=__section__&project=kernel-4.19)(#[\_\_sec](http://localhost:8080/source/s?defs=__sec&project=kernel-4.19) ".init")的展开结果就是：

[\_\_section\_\_](http://localhost:8080/source/s?defs=__section__&project=kernel-4.19)([.initcall6.init](http://localhost:8080/source/s?path=.initcall6.init&project=kernel-4.19)) = fn

**附注：**同一等级的优先级的驱动，加载顺序是链接过程决定的，结果是不确定的，我们无法去手动设置谁先谁后。****

**驱动内部加载过程**

**驱动更常见的写法如下，以spi设备为例：**

**static struct spi\_driver spidev\_spi\_driver = {**

**.driver = {**

**.name = "spidev",**

**.of\_match\_table = of\_match\_ptr(spidev\_dt\_ids),**

**},**

**.probe = spidev\_probe,**

**.remove = spidev\_remove,**

**};**

**static int \_\_init spidev\_init(void)**

**{**

**...**

**status = spi\_register\_driver(&spidev\_spi\_driver);**

**...**

**return status;**

**}**

**static void \_\_exit spidev\_exit(void)**

**{**

**spi\_unregister\_driver(&spidev\_spi\_driver);**

**...**

**}**

**module\_init(spidev\_init);**

**module\_exit(spidev\_exit);**

**上面是目前基于“总线设备驱动模型”的通用写法。将设配的的差异化部分提取成配置文件dts。共通部分的逻辑抽取成了平台化代码，开发人员不再赘写。**

**Kernel在开机过程中将解析dts（dtbo）文件并组装成内存对象，kenel又提供了完备的以of\_xxx为前缀的方法来读取这些dts对象。当dts中配置的compatible字段和上面struct spi\_driver结构体中定义的字符匹配时会自动调用在.probe中赋值的方法。**