知乎

入职Linux驱动工程师后，我才知道的真相.....

Vincent
软件开发行业 从业人员

关注

220 人赞同了该文章 〉

大家好，我是Vincent。

做Linux驱动工程师也有一段时间了，今天分享一下我曾经入职才知道的一些事情，算是一个菜鸟的经历吧。

设备树+

起初学习Linux驱动，是从最简单的一个.c文件开始。

在.c中实现 `module_init` 和 `module_exit` 这两个函数，然后在 `module_init` 的函数里加个 `printk` ，输出个hello world。

把.c编译成.ko，然后 `insmod` 加载驱动，看到有打印，就算成功了。

到了后来，我接触到设备树，加载 `dtb` 和 `.ko` 驱动，看到有打印，又成功了。心想，设备树应该就是这样吧。

直到有一天，工作中遇到一个uboot+没打印问题，主管问我：你设备树编进uboot了吗？

我懵了：把...把Linux的设备树编进uboot里？这能用？

主管：不是啊，uboot自己的设备树啊。

曾经我一度以为设备树是Linux的东西，别的程序没有，另外当时对uboot了解很浅，只简单用过几个命令。

直到那天，我才明白，所有的程序都可以有自己的设备树，不仅仅是uboot和linux，只要它实现了设备树这一套机制就可以。包括Linux的那一套 `Kconfig`+ 机制，在其他的一些开源项目中也在广泛应用。

工具链

编译一个程序在开发板上跑，通常我们都是用厂商提供交叉编译工具链+。但我们自己换一个工具链，可能编译出的程序在开发板上就跑不了。

知乎

另外我们平时用的像 arm-linux-gcc 这种工具链名称都是简写，除了架构和运行平台，看不出其他的区别。

其实不同的工具链除了架构和运行平台上的区别，主要还有C库、gcc版本的区别。

交叉工具链在制作的时候，可以选择具体的C标准库， glibc 、 uclibc 或者 musl 等等，如果这个工具链的C标准库是 glibc 的，那么用这个工具链编译出来的程序，就不能在 uclibc 或者 musl 这些非 glibc 库的文件系统下运行，否则就会报 command not found 错误，明明有这个可执行文件，却说找不到，让你百思不得其解。

交叉工具链所使用的gcc版本影响也很大，因为不同的gcc版本所对应的C库版本不一样。例如，用 gcc10 的工具链做的 glibc 文件系统，里面C库版本只支持到 2.30 ，这时用一个 gcc12 的工具链编译一个程序在该文件系统上运行，就会提示 glibc 版本找不到的错误。

文件系统

就像上面说的，工具链和文件系统的关系是很大的。

虽然是做驱动开发，但是文件系统的一些东西也是要知道的。

曾经刚入职的时候，要把一个.ko驱动在系统启动完成前就加载，这时才知道原来可以把命令放到 /etc/init.d/rcS 里

作为驱动工程师，可以不用把文件系统了解的太深，但起码要知道 inittab 、 rcS 、 passwd 和 shadow 这几个文件的作用，还有就是前面说的C库。

rcS 是文件系统启动时要执行的一些命令， inittab 、 passwd 和 shadow 主要是修改系统登录时的用户名和密码，包括设置免密登录等等。

通常把文件系统提供给客户前，都会把用户名改为 root ，密码改为自己公司的名字，这时就会用到这几个文件。

驱动怎么编进内核

一开始，我知道Linux内核源码中每个目录下都有一个 Makefile ，我以为改个 Makefile 就行了。

但一运行，驱动没生效。后来才知道，原来还要修改 Kconfig 。

把驱动编进内核，其实正确的做法应该是通过 menconfig 菜单能够配置这个驱动是否编译，即修改 Kconfig 。

Makefile 和 Kconfig 都修改了，驱动还是没生效?

这时就要看 .o 文件是否编译出来了，如果编译出来了。进一步看这个驱动的初始化级别，看是 module_init 、 arch_initcall 还是其他的级别。

然后加打印，把内核 initcall 的等级打印出来，看内核是否已经跑了该等级的初始化。如果已经跑到了，再把该等级的 initcall 执行函数的地址打印出来，然后反汇编 vmlinux ，看是否已经执行了新加驱动的probe函数。

基本上，通过以上过程的分析，就能够定位到为题所在。

手动修改defconfig配置引发的问题

编译内核时，通常都会用厂商提供的一个默认配置文件，例如 make xxx_defconfig 。

知乎

这是很多新手改 defconfig 都会遇到的问题，其实是没有搞懂如何正确修改 defconfig 文件。

在 defconfig 中定义了 CONFIG_XXX=y 后，还要在 Kconfig 文件中添加一个 config XXX 的配置才会生效。

另外，如何某个配置选项存在依赖关系，但依赖的配置选项没打开，也会出现这种不生效的情况。

所以还是建议通过 menconfig 菜单进行配置，除非真的弄清楚了这些关系才能去手动修改 defconfig 。

源码阅读/跟踪问题

虽然有一些比较有用的内核调试技巧，但真正常用的，还是加打印跟踪，其它调试技巧归根到底还是辅助性的，很多情况下还是靠加打印分析问题。

在跟踪源码，解决一些问题的时候，也要注意一些技巧。

曾经在解决一个内核自解压的问题时，加打印跟了很久，甚至跟踪到解压缩算法，但其实像这种算法性、协议性的东西，尽量少去怀疑它的错误，就是不用花太多时间去跟踪这些协议算法是怎么实现的，而是跟踪一些函数传参过程，关注是否正确使用，是否正确传参问题。因为最终解决问题可能只是一个很简单的操作。

需要学会多少个驱动才行?

初级的驱动工程师，只需要会一些简单的驱动，例如一些简单的时钟、定时器、led这些驱动。

但如果往上进阶，就需要会更多的驱动，例如USB、网卡这些复杂的，这些驱动的前提是你得懂时钟、复位、dma等这些驱动，因为会用上这些驱动的接口，所以要求会高一点。

当你能够掌握一个比较复杂的驱动，这其中自然就会涉及到其他的一些基础驱动，自然会的更多，这往往对应的是高级工程师。

所以，会多少驱动，这其实是一个进阶的过程。没有说一定要会多少个，但随着工作经验的增长，自身所掌握的驱动也会越来越多，承担的责任也越大。

工作职责问题

如前面所说，虽然叫Linux驱动工程师，但工作绝不仅仅是写Linux的驱动。

准确的说，应该叫底层开发工程师。因为除了Linux驱动，uboot、文件系统、系统移植都是要弄的。

我由于部门的特殊性，还需要做一些芯片流片前的验证，就经常需要看一些汇编代码，反汇编程序，通过仿真出来波形进行分析。流片回来后，还需要bringup。

所以，只要是底层相关的工作，多少都会涉及的，只不过可能有些企业会分得比较细，把系统层和驱动层分开。

编辑于 2023-06-03 17:01 · IP 属地广东

Linux　Linux 内核　工程师