《openEuler内核编程》

课程讲稿

第一章 第3讲

内核编程环境

软件所制

第一章 第3讲 内核编程环境

**学时：**1学时

**教学目的：**通过现场编码操作演示，了解在真实硬件下，Linux内核的下载方法。并安装编译内核所需的全套工具。学习menuconfig工具来对内核编译进行配置。学会使用make命令来编译内核源码。并安装内核。学习第一个Linux内核模块的编写，简单了解Makefile的编写方法。学习加载与卸载内核模块。**注意：此课时的代码部分需课上实时演示以达到最优教学效果。**

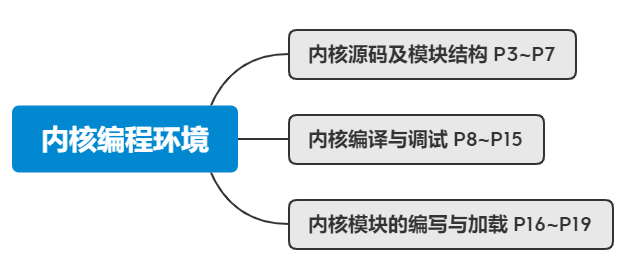
**课程时间线：**



**课外参考读物：**

1. <https://zhuanlan.zhihu.com/p/19796979>
2. https://www.oschina.net/news/111410/linux-market-growth
3. https://lkml.org/lkml/2020/5/17/314
4. https://www.oschina.net/news/100042/linux-september-2018-stats
5. http://www.makelinux.net/kernel\_map/
6. <https://www.tuicool.com/articles/ABbaAb>

**知识框图：**

****

**PPT讲稿：**



下面我们从三个方面来完成对Linux内核的第一次接触。第一是了解内核源码和模块的结构，第二是内核模块的编译与调试，第三是如何自定义一个模块并加载进内核。



工欲善其事必先利其器，想学好内核必须得亲自上手，并准备好趁手的武器。我们学习Linux操作系统，自己也必须安装一个。在过去很多年的操作系统课程中，一般来说我们都是在自己的电脑上安装一个虚拟机，然后在虚拟机里面去装操作系统，然后做各种实验。但这种学习方法总是像雾里看花，毕竟我们学习的就是内核，非常底层的东西，和硬件结合比较紧密，在虚拟机里面学习总感觉缺少了真实的硬件环境。所以这学期我们第一次采用真实硬件来进行内核的实验。当然真实硬件肯定不能用我们的个人电脑，要是不小心把电脑弄坏了，影响其他课程也不好。所以我们采用一款非常好用却又非常廉价的硬件，很多同学可能已经猜到是什么了，就是树莓派。我们几乎所有的实验都会在树莓派上去进行，但树莓派的硬件也有一些限制，比如它的处理器主频不高、内存不大等，后面用到一些复杂计算的功能可能还是需要在虚拟机里去完成。总体的思路，就是以树莓派为主，虚拟机为辅，来进行内核的学习。我们用到的树莓派是树莓派4B，性能上足够我们大部分实验。一个树莓派只要两百多块钱，加上基础配件也才三百多，大家务必保证人手一个。虚拟机大家也要安装，常用的有Virtualbox，vmware，qemu等等，这些根据大家的需要自己选择即可。



Linux的发行版有很多，我们前面也讲过。本课程选择开源的openEuler操作系统。这个操作系统是华为基于Linux 4.19版本自主研发的。我们会用到两个版本，一个就是正常的桌面版，装在虚拟机里。另一个是专门针对树莓派优化过的版本，是树莓派专用的。两个版本大家都要安装好，做实验的时候都要用到。



一般Linux的不同发行版都会把Linux内核的源码也一同放在系统里。位置在usr/src/linux下面，不同的发行版位置可能不同，但都差不多。我们打开openEuler的源码目录，简单说一下每个目录的内容。按顺序来，第一个arch，前面也说过，里面存放的是对不同架构CPU的支持源码，目前支持24种不同的架构。Block是块设备，可以理解为存储设备。Certs是证书相关的内容。Crypto存的是重要和常用的加密算法，比如md5，sha1等。Documentation，没什么好说的，文档，一些内核功能的原理、使用说明在这里可以找到。Drivers存放的是linux所支持硬件的驱动源码。Firmware是固件源码。Fs是file system的缩写，是文件系统相关的代码。Include是各个不同模块公用的头文件。Init存放的是内核启动时初始化的代码，上节课应该刚讲过。Ipc是进程间通信代码，我们课程上也要详细讲。Kernel是内核的进程管理相关的代码。Lib是公用的库函数，要注意在内核中是没有办法用C语言的标准库（如libc/glibc等）的。Mm是memory management的缩写，即内存管理。Net是网络协议栈相关的代码。Samples是一些示例。Scripts是一些辅助的脚本。Security是系统安全相关的代码。Sound是音频驱动相关的代码。Tools是内核的一些实用工具。Usr用于打包和压缩cpio。Virt是虚拟化相关的目录。



如果学习Linux内核，这一张图一定绕不开，非常经典的内核关系图。我们先从下往上看。最底下一层是和硬件最紧密结合的一层，可以看到有CPU、内存、磁盘控制、网络控制等模块。第二层是硬件接口，和第一层相比有了更加抽象的封装。第三层是设备控制。第四层是逻辑层。越往上面约抽象，越看不见底层的硬件。第五层是桥，这个比较特殊。我们看图也能看出来，这一层的模块会横跨在不同的元素之间，比如页面缓存，就是和内存和硬盘存储相关的模块，网络存储就是和网络及存储均相关的模块。再上一层，第六层就是虚拟层，像线程、虚拟内存，文件系统都是在这一层。到了这一层我们编程的时候就会经常碰到了。最高层第七层就是和用户空间相关的一些接口，比如我们打开一个文件，就是直接调用的这一层的函数。



下面说一下Linux内核的编译，首先要下载内核源码。官网地址是kernel.org。这里我们选和openEuler相同版本的内核4.19，linux下用wget工具就可以了。



下载过后用tar命令解压缩。然后进入到源码目录。



编译源码需要用到很多工具，下面这些一个都不能少。随着时间的推移这里列的一些软件包可能名字会变，有的时候可能还需要其他的软件包，这个大家需要靠自己的努力来把安装工具搞定。简单说一下用的软件包。**ncurses：**一套编程库，它提供了一系列的函数以便使用者调用它们去生成基于文本的用户界面。**xz-utils：**一个解压软件。**bc：**一个命令行计算软件。**flex：**一个字符分析工具。**elfutils：**用来对ELF文件进行读、写等操作。**bison：**一种通用解析器生成器。**m4：**一个通用的宏处理器。**openssl：**加密算法库。

接下来选择具体的编译配置。可选的配置界面有很多，我们在命令行的环境下，直接使用make menuconfig。这个是命令行的可视化界面，操作起来比较方便。每一个模块都有三种编译状态。即Y、N、M。Y代表yes，表示将模块编译到内核内，启动时加载。N表示no，不编译。M表示以模块级module的形式编译成.o文件，需要的时候再加载进内核。这个就是所谓的内核裁剪。我们这次什么都不改，直接保存。

然后输入make命令，如果没问题的话系统就会开始编译了，如果出错，看看自己是不是哪个软件包没有安装。Make有一些参数可以选，比如-j后面加数字就是要用多少个线程同时编译，O表示编译完文件的存放位置。默认的话什么都不写就是用单线程，编译完的文件还是直接存在当前目录下。我在一个虚拟机上的openEuler操作系统进行的编译，一共用了3个多小时。



编译完之后就要安装，分为两步，第一步安装模块，第二步安装内核： make modules\_install make install，安装的速度很快。然后重启，在选择页面就可以看到我们新编译的内核了。



选择新内核后，我们可以看到登陆完之后的欢迎页面已经是新版的内核号了。通过uname -r的命令查询也是新版的内核。到这，我们内核的编译和安装就算是完成了。



现在通过编译内核，感觉稍微摸到了一点Linux内核的门道。下面我们来看看如何自己动手写一个内核模块。写一个Linux内核的Hello World，属于计算机行业的老传统了。编程语言不用说，就是C语言。我拿一个最小化的内核模块源码来给大家拆解。首先是引用的三个头文件，都是必备的。其中，init.h 定义了驱动的初始化和退出相关的函数，kernel.h 定义了经常用到的函数原型及宏定义，module.h 定义了内核模块相关的函数、变量及宏。然后是当前模块的license声明，这个是必须要有的。下面是一个静态函数。这个函数和普通的静态函数多了一个\_\_init修饰符，这个宏的定义实在第一行的init.h头文件中。意思就是这个是模块的入口函数。函数里面就随便写了。我打印一个hello openEuler。这里用的是printk函数。Printf在内核是用不了的。KERN\_INFO表示的信息级别。最后return一个0。如果内核加载失败就返回的不是0。有入口就有出口。下面这个函数也是类似，只不过是没有返回值的。最后是两个函数，用来加载和卸载内核模块用的。



代码写好了，还要编译。我们需要用到Makefile。Makefile主要就是一些配置信息，比如到哪里去找相应的头文件。输入make命令，然后就会得到一个hello.ko文件，这个就是我们写好的内核模块了。



内核模块是可以在运行中进行加载的。命令就是insmod加模块名。加载完之后你会发现，什么都没有发生，老师，说好的hello world呢？刚才我们说到了printk。这个函数是不会再终端里面输出的，相关的信息会输出在文档里，用dmesg命令可以查看。这里就能看到加载时候输出了Hello openEuler。卸载的时候输出了Bye openEuler。至此我们第一个内核模块的体验就结束了，本节课有关内核源码编译以及自定义内核的部分同学们回去要花点时间自己动手做一下。