概述Linux系统的驱动框架及驱动加载 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E6_A6_82_E8_BF_B0Linu_c103_144140.htm 本讲主要概述Linux设备驱动框架、驱动程序的配置文件及常用的加载驱动程序的方法；并且介绍Red Hat Linux安装程序是如何加载驱动的，通过了解这个过程， 我们可以自己将驱动程序放到引导盘中；安装完系统后，使用kudzu自动配置硬件程序。 Linux设备驱动概述 1. 内核和驱动模块 操作系统是通过各种驱动程序来驾驭硬件设备，它为用户屏蔽了各种各样的设备，驱动硬件是操作系统最基本的功能，并且提供统一的操作方式。正如我们查看屏幕上的文档时,不用去管到底使用nVIDIA芯片，还是ATI芯片的显示卡，只需知道输入命令后，需要的文字就显示在屏幕上。硬件驱动程序是操作系统最基本的组成部分，在Linux内核源程序中也占有较高的比例。 Linux内核中采用可加载的模块化设计（LKMs，Loadable Kernel Modules），一般情况下编译的Linux内核是支持可插入式模块的，也就是将最基本的核心代码编译在内核中，其它的代码可以选择是在内核中，或者编译为内核的模块文件。 如果需要某种功能，比如需要访问一个NTFS分区，就加载相应的NTFS模块。这种设计可以使内核文件不至于太大，但是又可以支持很多的功能，必要时动态地加载。这是一种跟微内核设计不太一样，但却是切实可行的内核设计方案。 我们常见的驱动程序就是作为内核模块动态加载的，比如声卡驱动和网卡驱动等，而Linux最基础的驱动，如CPU、PCI总线、TCP/IP协议、APM（高级电源管理）、VFS等驱动程序则编译在内核文

件中。有时也把内核模块就叫做驱动程序，只不过驱动的内容不一定是硬件罢了，比如ext3文件系统的驱动。 理解这一点很重要。因此，加载驱动时就是加载内核模块。下面来看一下有关模块的命令，在加载驱动程序要用到它们：lsmod、modprob、insmod、rmmod、modinfo。 lsmod 列出当前系统中加载的模块，例如： #lsmod （与cat /proc/modules 得出的内容是一致的）Module Size Used by Not taintedradeon 115364 1 agpgart 56664 3 nls_iso8859-1 3516 1 (autoclean)loop 12120 3 (autoclean)smbfs 44528 2 (autoclean)parport_pc 19076 1 (autoclean)lp 9028 0 (autoclean)parport 37088 1 (autoclean) [parport_pc lp]autofs 13364 0 (autoclean) (unused)ds 8704 2 yenta_socket 13760 2 pcmcia_core 57184 0 [ds yenta_socket]tg3 55112 1 sg 36940 0 (autoclean)sr_mod 18104 0 (autoclean)microcode 4724 0 (autoclean)ide-scsi 12208 0 scsi_mod 108968 3 [sg sr_mod ide-scsi]ide-cd 35680 0 cdrom 33696 0 [sr_mod ide-cd]nls_cp936 124988 1 (autoclean)nls_cp437 5148 1 (autoclean)vfat 13004 1 (autoclean)fat 38872 0 (autoclean) [vfat]keybdev 2976 0 (unused)mousedev 5524 1 hid 22212 0 (unused)input 5888 0 [keybdev mousedev hid]ehci-hcd 20104 0 (unused)usb-uhci 26412 0 (unused)usbcore 79392 1 [hid ehci-hcd usb-uhci]ext3 91592 2 jbd 52336 2 [ext3]上面显示了当前系统中加载的模块，左边数第一列是模块名，第二列是该模块大小，第三列则是该模块使用的数量。 如果后面为unused，则表示该模块当前没在使用。如果后面有autoclean，则该模块可以被rmmod -a命令自动清洗。rmmod -a命令会将目前有autoclean的模块卸载，如果这时候某个模块未被使用，则

将该模块标记为autoclean。如果在行尾的[ ]括号内有模块名称，则括号内的模块就依赖于该模块。例如： cdrom 34144 0 [sr_mod ide-cd]其中ide-cd及sr_mod模块就依赖于cdrom模块。系统的模块文件保存在/lib/modules/2.4.XXX/kerne目录中，根据分类分别在fs、net等子目录中，他们的互相依存关系则保存在/lib/modules/2.4.XXX/modules.dep 文件中。 需要注意，该文件不仅写入了模块的依存关系，同时内核查找模块也是在这个文件中，使用modprobe命令，可以智能插入模块，它可以根据模块间依存关系，以及/etc/modules.conf文件中的内容智能插入模块。比如希望加载ide的光驱驱动，则可运行下面命令： # modprobe ide-cd此时会发现，cdrom模块也会自动插入。 insmod也是插入模块的命令，但是它不会自动解决依存关系，所以一般加载内核模块时使用的命令为modprobe。 rmmod可以删除模块，但是它只可以删除没有使用的模块。 Modinfo用来查看模块信息，如modinfo -d cdrom，在Red Hat Linux系统中，模块的相关命令在modutils的RPM包中。 2．设备文件 当我们加载了设备驱动模块后，应该怎样访问这些设备呢？Linux是一种类Unix系统，Unix的一个基本特点是“ 一切皆为文件 ”，它抽象了设备的处理，将所有的硬件设备都像普通文件一样看待，也就是说硬件可以跟普通文件一样来打开、关闭和读写。 系统中的设备都用一个设备特殊文件代表，叫做设备文件，设备文件又分为Block（块）型设备文件、Character（字符）型设备文件和Socket（网络插件）型设备文件。Block设备文件常常指定哪些需要以块（如512字节）的方式写入的设备，比如IDE硬盘、SCSI硬盘、光驱等。 而Character型设备文件常指定直接读写，没有缓冲区的设备

，比如并口、虚拟控制台等。Socket（网络插件）型设备文件指定的是网络设备访问的BSD socket 接口。 ＃ls -l /dev/hda /dev/video0 /dev/logbrw-rw---- 1 root disk 3, 0 Sep 15 2003 /dev/hdasrw-rw-rw- 1 root root 0 Jun 3 16:55 /dev/logcrw------- 1 root root 81, 0 Sep 15 2003 /dev/video0上面显示的是三种设备文件，注意它们最前面的字符，Block型设备为b，Character型设备为c，Socket设备为s。 由此可以看出，设备文件都放在/dev目录下，比如硬盘就是用/dev/hd*来表示，/dev/hda表示第一个IDE接口的主设备，/dev/hda1表示第一个硬盘上的第一个分区；而/dev/hdc 表示第二个IDE接口的主设备。可以使用下面命令： ＃dd if=/dev/hda of=/root/a.img bs＝446 count＝1把第一个硬盘上前446个字节的MBR信息导入到a.img文件中。 对于Block和Character型设备，使用主（Major）和辅（minor）设备编号来描述设备。主设备编号来表示某种驱动程序，同一个设备驱动程序模块所控制的所有设备都有一个共同的主设备编号，而辅设备编号用于区分该控制器下不同的设备，比如，/dev/hda1（block 3/1）、/dev/hda2(block 3/2) 和/dev/hda3(block3/3)都代表着同一块硬盘的三个分区，他们的主设备号都是3，辅设备号分别为1、2、3。 这些设备特殊文件用mknod命令来创建： # mknod harddisk b 3 0我们就在当前位置创建出一个与 /dev/hda一样的、可以访问第一个IDE设备主硬盘的文件，文件名叫做harddisk。 使用下面命令可以查看设备编号： #file /dev/hda/dev/hda: block special (3/0)其中Block代表/dev/hda是系统的Block型（块型）设备文件，它的主设备编号为3，辅设备编号为0。 #ls -l /dev/hda /dev/hdb brw-rw---- 1 root disk 3, 0 Sep 15 2003 /dev/hdabrw-rw---- 1 root

disk 3, 64 Sep 15 2003 /dev/hdb使用ls －l也可以看到设备编号，/dev/hdb代表第一个IDE接口的从设备（Slave）也是Block设备，编号为(3/64),还有另外一种设备文件是/dev/tty*。使用如下命令：#echo "hello tty1" > /dev/tty1将字符串“ hello tty1 ”输出到/dev/tty1代表的第一个虚拟控制台上，此时按“ Alt F1 ”可以看到该字符出现在屏幕上，这个特殊的文件就代表着我们的第一虚拟控制台。 # file /dev/tty1/dev/tty1: character special (4/1)由上可以看到，它的类型为Character 型（字符型）设备文件，主设备号为4，辅设备号为1。同样，/dev/tty2代表着第二个虚拟控制台，是Character设备，编号为 (4/2)。 当将/dev/cdrom加载到/mnt/cdrom中时，只要访问/mnt/cdrom系统就会自动引入到/dev/cdrom对应的驱动程序中，访问实际的数据。 有关设备文件的编号可以看内核文档/usr/src/linux-2.*/Documentation/devices.txt 文件(在Kernel的源文件解包后的Documentation目录中)，其中详细叙述了各种设备文件编号的意义。 3.使用/proc目录中的文件监视驱动程序的状态 通过设备文件怎样访问到相应的驱动程序呢？它们中间有一个桥梁，那就是proc文件系统，它一般会被加载到/proc目录。访问设备文件时，操作系统通常会通过查找/proc目录下的值，确定由哪些驱动模块来完成任务。如果proc文件系统没有加载，访问设备文件时就会出现错误。Linux系统中proc文件系统是内核虚拟的文件系统，其中所有的文件都是内核中虚拟出来的，各种文件实际上是当前内核在内存中的参数。它就像是专门为访问内核而打开的一扇门，比如访问/proc/cpuinfo文件，实际上就是访问目前的CPU的参数，每一次系统启动时系统都会通过/etc/fstab中设置的信息

自动将proc文件系统加载到/proc目录下：# grep proc /etc/fstabnone /proc proc defaults 0 0此外，也可以通过mount命令手动加载: # mount -t proc none /proc通过/proc目录下的文件可以访问或更改内核参数，可以通过/proc目录查询驱动程序的信息。下面先让我们看一下/proc目录中的信息：# ls /proc1 4725 5032 5100 5248 5292 crypto kcore partitions14 4794 5044 5110 5250 5293 devices kmsg pci2 4810 5075 5122 5252 5295 dma ksyms self3 4820 5079 5132 5254 5345 driver loadavg slabinfo4 4831 5080 5151 5256 6 execdomains locks stat4316 4910 5081 5160 5258 7 fb lvm swaps4317 4912 5082 5170 5262 70 filesystems mdstat sys4318 4924 5083 5180 5271 8 fs meminfo sysrq-trigger4319 4950 5084 5189 5287 9 ide misc sysvipc4620 4963 5085 5232 5288 apm interrupts modules tty4676 5 5086 5242 5289 bus iomem mounts uptime4680 5005 5087 5244 5290 cmdline ioports mtrr version4706 5018 5088 5246 5291 cpuinfo irq net需要知道的是，这些文件都是实时产生的虚拟文件，访问它们就是访问内存中真实的数据。这些数据是实时变化产生的，可以通过以下命令来查看文件的具体值：# cat /proc/interruptsCPU00: 50662 XT-PIC timer1: 3 XT-PIC keyboard2: 0 XT-PIC cascade5: 618 XT-PIC ehci-hcd, eth18: 1 XT-PIC rtc9: 0 XT-PIC usb-uhci, usb-uhci11: 50 XT-PIC usb-uhci, eth012: 16 XT-PIC PS/2 Mouse14: 8009 XT-PIC ide015: 0 XT-PIC ide1NMI: 0ERR: 0其它文件的含意见表1所示。/proc/sys目录下的文件一般可以直接更改，相当于直接更改内核的运行参数，例如：# echo 1 > /proc/sys/net/ipv4/ip_forward上面代码可以将内核中的数据包转发功能打开。 另外，Linux系统中提供一些命令来查询系统的状态，如free可以查看目前

的内存使用情况，ide_info可以查看ide设备的信息，例如：#ide_info /dev/had。类似的命令还有scsi_info，可以查看SCSI设备的信息。这些命令一般也是查询/proc目录下的文件，并返回结果。 系统初始化过程驱动程序的安装 在Linux安装过程中，系统上的硬件会被检测，基于检测到的结果安装程序会决定哪些模块需要在引导时被载入。Red Hat的安装程序为anaconda，它提供了自动检测硬件，并且安装的机制。 但是，如果计算机内的某些硬件没有默认的驱动程序，比如一块SCSI卡，我们可以在启动后的boot提示符下，输入“ linux dd ”，在加载完内核后，系统会自动提示插入驱动盘，这时就有机会把该硬件的Linux驱动程序装入。 如果在安装系统时，某种硬件总是因为中断冲突（ISA总线的设备较常见，比如一块ISA网卡）没法正常驱动，或者是缺少驱动程序，那么可以在boot提示符下输入“ linux noprobe ”。在这种模式下，安装程序不会自动配置找到的硬件，可以自己来选择现有驱动，配置驱动程序的参数，或者选择用光盘或软盘加载驱动程序。 定制引导盘 系统启动时是如何加载驱动的？下面让我们来看一下Red Hat的安装光盘是怎样引导的。当Linux安装光盘启动时，加载位于光盘上isolinux中的内核文件vmlinuz，内核运行完毕后，又将initrd.img的虚拟文件系统加载到内存中。这个文件为ext2文件系统的镜像，经过gzip压缩，可以通过以下步骤查看该镜像中的内容： # mount /mnt/cdrom# mkdir /mnt/imgdir# gunzip /ext2img# mount -t ext2 -o loop /ext2img /mnt/imgdir# cd /mnt/imgdir# ls -Fbin@dev/etc/linuxrc@lost found/modules/proc/sbin/tmp/var/# cd modules# ls module-infomodules.cgzmodules.depmodules.pcimappcitable其

中modules.dep为模块的注册文件，同时有各种模块的依存关系。modules.cgz为cpio的打包文件，实际的各种驱动模块就在该文件中。我们可以通过以下命令解包：# cpio -idmv 由此可以看到，解包出来的目录2.4.21-4XXX。进入该目录下的i386目录，就可以看到当前启动盘中支持的所以驱动程序：# ls 3c59x.o3w-xxxx.o8139cp.o8139too.o8390.oaacraid.oacenic.oaic79xx.o……若希望在系统中加入需要的驱动程序，可以相应地修改这些文件，比如在modules.dep中加入该模块的名字和依存关系，将编译好的驱动模块文件加入modules.cgz中，这样就可以制定自己的安装光盘。 硬盘上的系统启动过程与上面类似，但是initrd的镜像文件要更简单些，一般在initrd-2.4.XXX.img的虚拟文件系统中，只会在/lib目录下包含ext3.o jbd.o lvm-mod.o等少数文件，用来驱动硬盘上的ext3的文件系统。加载文件系统后，就可以使用/lib/modules/2.4.XXX/下的modules.dep文件及Kernel目录中的各种驱动文件。自动配置安装 如果安装完Linux系统后，又添加了新的硬件，那么系统必须载入正确的驱动程序才可以使用它。在Red Hat Linux中，可以使用kudzu来配置硬件。这是PnP设备的检测程序，当系统使用新硬件引导后，运行kudzu（默认会自动运行），如果新硬件被支持，那么它就会被自动检测到。该程序还会为它配置驱动模块，把结果写入到文件/etc/sysconfig/hwconf中，kudzu可以通过对比这个文件发现新安装的硬件，并进行配置；也可以通过编辑模块配置文件/etc/modules.conf来手工指定加载模块。 Kudzu服务默认每次启动时都要运行，如果需要缩短启动时间，使用下面命令可以停止系统启动时的kudzu服务：# chkconfig kudzu off

如果要安装新的硬件，可以手动运行kudzu程序。 # kudzu那么kudzu程序如何认识硬件的呢？可以查看/usr/share/hwdata/目录下的文件，根据这些文件中的PnP信息，kudzu可以识别各种硬件设备。 以上介绍了Linux下驱动程序的大体结构、主要的加载方式和相关配置文件，在安装Linux时加载驱动程序，并且根据需要定制自己的引导盘，在安装完成后安装新的、即插即用硬件。下一讲开始，我们将学习具体硬件驱动的安装方法。