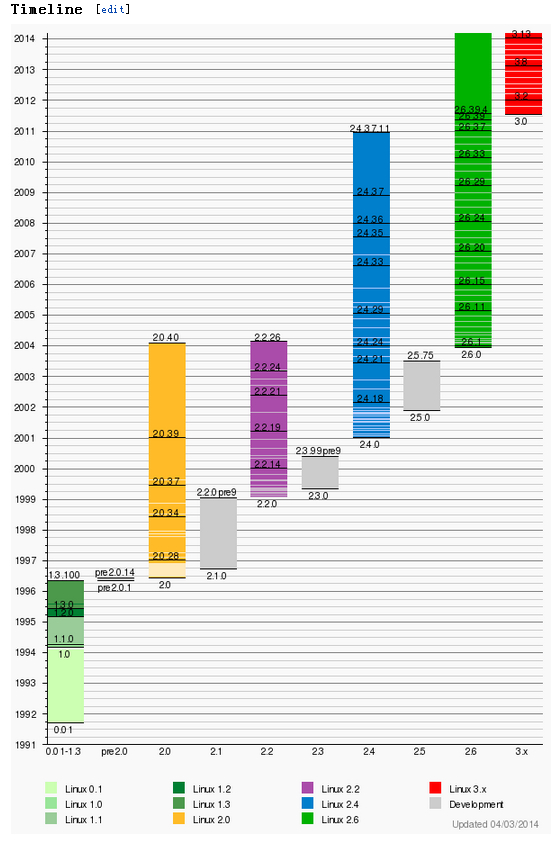
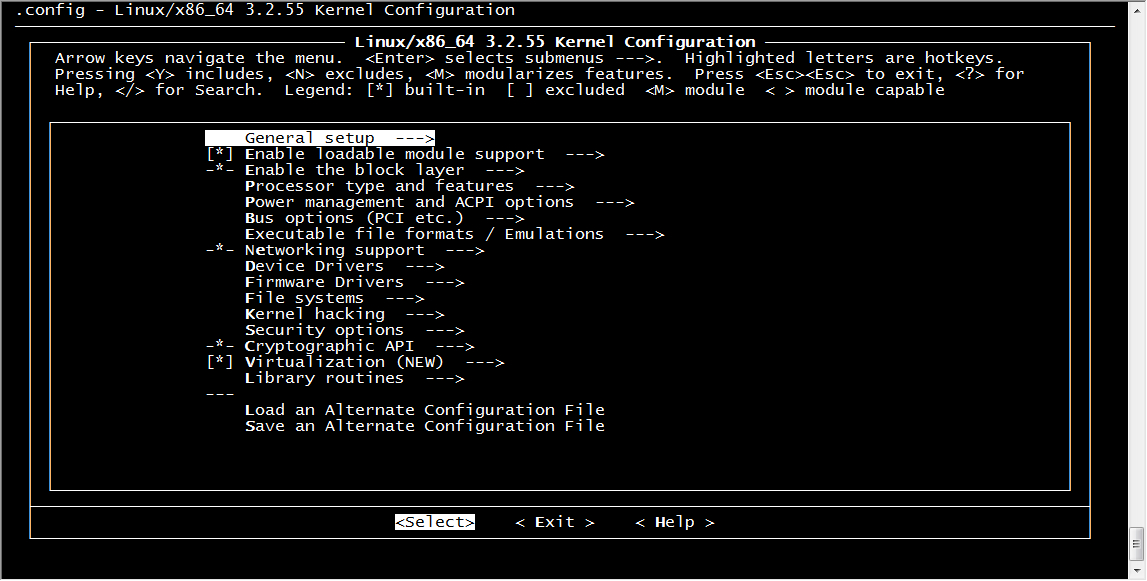
# **网络资料**

**一、Linux内核概览**  
Linux是一个一体化内核（monolithic kernel）系统。  
设备驱动程序可以完全访问硬件。  
Linux内的设备驱动程序可以方便地以模块化（modularize）的形式设置，并在系统运行期间可直接装载或卸载。  
  
1. linux内核  
linux操作系统是一个用来和硬件打交道并为用户程序提供一个有限服务集的低级支撑软件。  
一个计算机系统是一个硬件和软件的共生体，它们互相依赖，不可分割。  
计算机的硬件，含有外围设备、处理器、内存、硬盘和其他的电子设备组成计算机的发动机。  
但是没有软件来操作和控制它，自身是不能工作的。  
完成这个控制工作的软件就称为操作系统，在Linux的术语中被称为“内核”，也可以称为“核心”。  
  
Linux内核的主要模块（或组件）分以下几个部分：  
. 进程管理（process management）   
. 定时器（timer）  
. 中断管理（interrupt management）  
. 内存管理（memory management）  
. 模块管理（module management）  
. 虚拟文件系统接口（VFS layer）  
. 文件系统（file system）  
. 设备驱动程序（device driver）  
. 进程间通信（inter-process communication）  
. 网络管理（network management  
. 系统启动（system init）等操作系统功能的实现。  
  
2. linux内核版本号  
Linux内核使用三种不同的版本编号方式。  
. 第一种方式用于1.0版本之前（包括1.0）。  
  第一个版本是0.01，紧接着是0.02、0.03、0.10、0.11、0.12、0.95、0.96、0.97、0.98、0.99和之后的1.0。  
. 第二种方式用于1.0之后到2.6，数字由三部分“A.B.C”，A代表主版本号，B代表次主版本号，C代表较小的末版本号。  
  只有在内核发生很大变化时（历史上只发生过两次，1994年的1.0,1996年的2.0），A才变化。  
  可以通过数字B来判断Linux是否稳定，偶数的B代表稳定版，奇数的B代表开发版。C代表一些bug修复，安全更新，新特性和驱动的次数。  
  以版本2.4.0为例，2代表主版本号，4代表次版本号，0代表改动较小的末版本号。  
  在版本号中，序号的第二位为偶数的版本表明这是一个可以使用的稳定版本，如2.2.5;  
  而序号的第二位为奇数的版本一般有一些新的东西加入，是个不一定很稳定的测试版本，如2.3.1。  
  这样稳定版本来源于上一个测试版升级版本号，而一个稳定版本发展到完全成熟后就不再发展。  
. 第三种方式从2004年2.6.0版本开始，使用一种“time-based”的方式。  
  3.0版本之前，是一种“A.B.C.D”的格式。  
  七年里，前两个数字A.B即“2.6”保持不变，C随着新版本的发布而增加,D代表一些bug修复，安全更新，添加新特性和驱动的次数。  
  3.0版本之后是“A.B.C”格式，B随着新版本的发布而增加,C代表一些bug修复，安全更新，新特性和驱动的次数。  
  第三种方式中不使用偶数代表稳定版，奇数代表开发版这样的命名方式。  
  举个例子：3.7.0代表的不是开发版，而是稳定版！  
  
linux内核升级时间图谱如下:  
  
linux内核版本的详细表格如下(http://en.wikipedia.org/wiki/Linux\_kernel#Version\_numbering):  
  
  
longterm support:提供长期支持的内核版本   
stable:          稳定版本，而不是Beta测试版  
   关系就是：稳定版本不一定都提供长期支持，而提供长期支持的一定是稳定版本  
  
3. linux发行版  
Linux内核的发展过程中，我们还不得不提一下各种Linux发行版的作用，因为正是它们推动了Linux的应用，  
从而也让更多的人开始关注Linux。  
一些组织或厂家，将Linux系统的内核与外围实用程序（Utilities）软件和文档包装起来，  
并提供一些系统安装界面和系统配置、设定与管理工具，就构成了一种发行版本（distribution），  
Linux的发行版本其实就是Linux核心再加上外围的实用程序组成的一个大软件包而已。  
相对于Linux操作系统内核版本，发行版本的版本号随发布者的不同而不同，与Linux系统内核的版本号是相对独立的。  
因此把SUSE、RedHat、Ubuntu、Slackware等直接说成是Linux是不确切的，它们是Linux的发行版本，  
更确切地说，应该叫做“以Linux为核心的操作系统软件包”。  
根据GPL准则，这些发行版本虽然都源自一个内核，并且都有自己各自的贡献，但都没有自己的版权。  
Linux的各个发行版本（distribution），都是使用Linus主导开发并发布的同一个Linux内核，因此在内核层不存在什么兼容性问题。  
每个版本都不一样的感觉，只是在发行版本的最外层才有所体现，而绝不是Linux本身特别是内核不统一或是不兼容。  
  
90年代初期Linux开始出现的时候，仅仅是以源代码形式出现，用户需要在其他操作系统下进行编译才能使用。后来出现了一些正式版本。  
目前最流行的几个正式版本有：SUSE、RedHat、Fedora、Debian、Ubuntu、CentOS、Gentoo，等等。  
用户可根据自己的经验和喜好选用合适的Linux发行版。  
  
**二、内核升级攻略**  
1. 查看linux内核版本:  
# uname -r  
2.6.18-194.el5  
  
2. 下载新的内核版本:  
去官网上下载最新的内核版本和对应的补丁:  
https://www.kernel.org/  
我选的是3.2.14  
#wget -c http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.2.14.tar.bz2  
  
3. 解压  
将所需文件复制到 /usr/src 下并解压。  
(**注：如果/usr/src下的空间不够，将源文件放在别的目录下也是一样的**)  
# cp linux-3.2.14.tar.gz /usr/src/  
# cd /usr/src/  
  
将文件解压到 /usr/src/kernels  
# tar -xvf linux-3.2.14.tar.gz -C /usr/src/kernels  
  
4. 清理文件  
清除掉以前升级留下的一些不正确文件。  
# cd /usr/src/kernels/linux-3.2.14  
# yum install ncurses-devel       #升级ncurses  
# make mrproper  
# cp /boot/config-‘uname -r’ /usr/src/kernels/linux3.2.14/.config  #使用原来的配置文件  
  
5. 通过菜单方式配置内核  
# cd /usr/src/kernels/linux-3.2.14  
# make menuconfig  
linux配置菜单如下图所示:  
  
  
一定要勾选（输入y选中，M编译为模块）  
a、General setup→[\*] enable deprecated sysfs features to support old userspace tools  
b、Processor type and features→HighMemory Support。  
c、找到以下选中选项并选中：  
networking support → networking options → network packet filtering framework(netfilter)  
(1)Corenetfilter configuration  
. 勾中"Netfilter connection tracking support" -m state相关模块是依赖它的，不选则没有。  
. 将netbios name service protocal support(new) 编译成模块,不然后面升级iptables后启动时会出错  
. 勾中"Netfilter Xtables support (required for ip\_tables)"  
  
(2)IP: Netfilter Configuration  
. 将 "IPv4 connection tracking support (require for NAT)" 编译成模块。  
. 勾中IP tables support (required for filtering/masq/NAT) 。  
. 将 "Full NAT" 下的 "MASQUERADE target support" 和 "REDIRECT target support" 编译成模块  
  
6. 编译并安装内核  
# make clean          // 清除下内核编译的目标文件  
# make bzImage    // 生成内核文件  
# make modules    // 编码模块  
# make modules\_install // 安装模块  
# make install         // 安装  
  
7. 编辑开机菜单选项grub文件  
将 default=1 改为 default=0  
# vim /etc/grub.conf  
  
# grub.conf generated by anaconda  
#  
# Note that you do not have to rerun grub after making changes to this file  
# NOTICE:  You do not have a /boot partition.  This means that  
#          all kernel and initrd paths are relative to /, eg.  
#          root (hd0,0)  
#          kernel /boot/vmlinuz-version ro root=/dev/sda1  
#          initrd /boot/initrd-version.img  
#boot=/dev/sda  
default=0  
timeout=5  
splashimage=(hd0,0)/boot/grub/splash.xpm.gz  
hiddenmenu  
title Red Hat Enterprise Linux Server (3.2.14)  
        root (hd0,0)  
        kernel /boot/vmlinuz-3.2.14 ro root=LABEL=/  
        initrd /boot/initrd-3.2.14.img  
title Red Hat Enterprise Linux Server (2.6.18-274.el5)  
        root (hd0,0)  
        kernel /boot/vmlinuz-2.6.18-274.el5 ro root=LABEL=/  
        initrd /boot/initrd-2.6.18-274.el5.img  
【保存并退出】  
  
8. 重启  
# shutdown -r "now"  
  
9. 查看内核编译结果  
# uname -r  
3.2.14-rt24  
  
**三、内核编译中遇到的错误及解决办法**  
**错误一、** 编译时出现错误提示  
In file included from /usr/include/sys/time.h:31,  
                 from /usr/include/linux/input.h:12,  
                 from samples/hidraw/hid-example.c:14:  
/usr/include/sys/select.h:78: error: conflicting types for 'fd\_set'  
/usr/include/linux/types.h:12: error: previous declaration of 'fd\_set' was here  
In file included from /usr/include/linux/input.h:14,  
                 from samples/hidraw/hid-example.c:14:  
/usr/include/sys/types.h:46: error: conflicting types for 'loff\_t'  
/usr/include/linux/types.h:30: error: previous declaration of 'loff\_t' was here  
/usr/include/sys/types.h:62: error: conflicting types for 'dev\_t'  
/usr/include/linux/types.h:13: error: previous declaration of 'dev\_t' was here  
In file included from /usr/include/sys/types.h:133,  
                 from /usr/include/linux/input.h:14,  
                 from samples/hidraw/hid-example.c:14:  
/usr/include/time.h:105: error: conflicting types for 'timer\_t'  
/usr/include/linux/types.h:22: error: previous declaration of 'timer\_t' was here  
In file included from /usr/include/linux/input.h:14,  
                 from samples/hidraw/hid-example.c:14:  
/usr/include/sys/types.h:198: error: conflicting types for 'int64\_t'  
/usr/include/linux/types.h:98: error: previous declaration of 'int64\_t' was here  
/usr/include/sys/types.h:204: error: conflicting types for 'u\_int64\_t'  
/usr/include/linux/types.h:97: error: previous declaration of 'u\_int64\_t' was here  
In file included from /usr/include/linux/input.h:14,  
                 from samples/hidraw/hid-example.c:14:  
/usr/include/sys/types.h:235: error: conflicting types for 'blkcnt\_t'  
/usr/include/linux/types.h:114: error: previous declaration of 'blkcnt\_t' was here  
samples/hidraw/hid-example.c:15:26: error: linux/hidraw.h: No such file or directory  
samples/hidraw/hid-example.c: In function 'main':  
samples/hidraw/hid-example.c:48: error: storage size of 'rpt\_desc' isn't known  
samples/hidraw/hid-example.c:49: error: storage size of 'info' isn't known  
samples/hidraw/hid-example.c:65: error: 'HIDIOCGRDESCSIZE' undeclared (first use in this function)  
samples/hidraw/hid-example.c:65: error: (Each undeclared identifier is reported only once  
samples/hidraw/hid-example.c:65: error: for each function it appears in.)  
samples/hidraw/hid-example.c:73: error: 'HIDIOCGRDESC' undeclared (first use in this function)  
samples/hidraw/hid-example.c:84: warning: implicit declaration of function 'HIDIOCGRAWNAME'  
samples/hidraw/hid-example.c:91: warning: implicit declaration of function 'HIDIOCGRAWPHYS'  
samples/hidraw/hid-example.c:98: error: 'HIDIOCGRAWINFO' undeclared (first use in this function)  
samples/hidraw/hid-example.c:49: warning: unused variable 'info'  
samples/hidraw/hid-example.c:48: warning: unused variable 'rpt\_desc'  
samples/hidraw/hid-example.c: In function 'bus\_str':  
samples/hidraw/hid-example.c:171: error: 'BUS\_VIRTUAL' undeclared (first use in this function)  
make[2]: \*\*\* [samples/hidraw/hid-example] Error 1  
make[1]: \*\*\* [samples/hidraw] Error 2  
make: \*\*\* [vmlinux] Error 2  
  
解决方案:  
# cp include/linux/hidraw.h /usr/include/linux/  
# cp include/linux/hid.h /usr/include/linux/  
# vim samples/hidraw/hid-example.c  
(在Linux下编译程序有时会遇到这种问题，这貌似是一个Linux历史遗留问题:   
把所有#include 都提到最前面，把#include 的包含放在其后，就可以编译通过了，  
之所以出现面的问题是存在循环引用所致。)  
将13-15行的如下3行移动到33行以后。       
13 #include       
14 #include       
15 #include       
29 #include       
30 #include       
31 #include       
32 #include       
33 #include   
【保存并退出】  
  
# vim /usr/include/linux/input.h  
line 647  
#define BUS\_PCI                 0x01  
#define BUS\_ISAPNP              0x02  
#define BUS\_USB                 0x03  
#define BUS\_HIL                 0x04  
#define BUS\_BLUETOOTH           0x05  
/\*以下一行为新增\*/  
#define BUS\_VIRTUAL             0x06  
#define BUS\_ISA                 0x10  
#define BUS\_I8042               0x11  
#define BUS\_XTKBD               0x12  
#define BUS\_RS232               0x13  
#define BUS\_GAMEPORT            0x14  
#define BUS\_PARPORT             0x15  
#define BUS\_AMIGA               0x16  
#define BUS\_ADB                 0x17  
#define BUS\_I2C                 0x18  
#define BUS\_HOST                0x19  
#define BUS\_GSC                 0x1A  
  
/\*以下两行为新增\*/  
#define BUS\_ATARI               0x1B  
#define BUS\_SPI                 0x1C   
【保存并退出】  
再重新编译；  
  
**错误二：**  
错误提示：Kernel panic—not syncing :vfs:unable to mount root fs on unkown—block (0,0)  
错误分析：initrd-\*.\*.img文件有问题；未能成功挂载真正的根文件系统。  
解决方法：请你确认执行了make install 命令，我就因为没有执行这条命令而浪费了一周时间  
  
**错误三：**  
错误提示：Create root device  mkrootdev: label / not found   
                     Mountingrootfilesystem   
                     mount :error2 mounting ext3  
                     mount :error 2mountingnone   
                     switch root mount failed:22  
                     umount  /initrd/dev failed :2  
                     kernel panic-not synicncing :Attempted to kill  init .  
(错误提示是不是有点长，貌似错误很多哦，其实可能就一个原因导致)  
错误分析：  
1、根文件系统找不到挂载点 ，修改grub.conf文件为 root=/dev/ \*   
2、硬盘驱动未能找到，请看看你机器的硬盘是都是 SATA盘，是的话同样恭喜你本文对你有帮助。  
  
错误解决：  
1.、修改grub.conf文件试试，具体修改点  
root=/dev/\*  
\* 指的是原系统中/分区的挂载点。修改后，重启进入新内核系统，估计十有八九又会遇到如下错误提示：  
Mounting root filesystem   
mount :error6 mounting ext3  
mount :error 2 mountingnone   
switchroot mount failed:22  
umount  /initrd/dev failed :2  
kernel panic-not synicncing :Attempted tokil  linit .  
如果不幸被我言中，请接着往下看。  
2、硬盘驱动未能成功找到，SCSI驱动有问题。  
我在内核编译升级中遇到这个问题，对内核选项选了又选，方法试了很多种，错误依旧。  
最终让我无意中选正确选项，成功升级内核。  
具体选项是：  
Device Drivers--------SCSI  
device support------SCSI  
low-level drivers----Serial ATA（SATA）support 选为 M，  
然后再选择相应的子选项，我选择了Intel PIIX/ICH  SATA support （new）后机器就成功启动了。  
  
**错误四：**  
错误提示：Enforcing mode requested but no policy loaded. Halting now.  
kernel panic - not syncing: Attenpted to kill init!  
错误分析：原内核中启用了selinux，新内核未选择selinux的选项，不能成功启动。  
解决方法：修改grub.conf文件，在root=LABEL=/ 后加入 enforcing=0   
  
**错误五：**  
编译完重启后出现：  
switchroot:   mount   failed:   22   
umount   /initrd/dev   failed:   2   
kernel   panic   -   not   symcing:   Attempted   to   kill   init!   
的错误,  
  
解决办法:  
重新make menuconfig, 使CONFIG\_SYSFS\_DEPRECATED\_V2=y  
也可以打开文件.config做修改和验证；  
再执行:  
# make bzImage  
# make modules  
# make modules\_install  
# make install  
# shutdown -r "now"  
  
**四、linux内核启动流程**  
计算机在启动时都是先加电,  
然后进行硬件检测并引导操作系统的初始化程序,  
然后操作系统的初始化程序程负责读入系统内核并建产系统的运行环境.  
一这过程相对来说比较复而且与CPU体系结构相关,这里我们通过linux并以i386的体系结构对这一过程进行较为详细的说明.   
  
**1、硬件检测**  
. 当机器加电后它首先执行BIOS(基本输入输出系统)中的代码，BIOS首先执行加电自检程序(POST)，当自检通过程便完成了硬件的启动。  
. POST程序通过对内存及其他硬件的设备的诊断检测确定硬件的存在并可正确操作。  
. BIOS是固化在芯片里的程序，执行这一过程一般只需要几秒钟。  
. 当自检完成后 BIOS按照系统COMS中设置的启动顺序搜寻有效的启动驱动器(这里我们以硬盘为例)，  
  并读入系统引导扇区，并将系统控制权交给引导程序。   
  
2、加载和执行引导程序   
系统引导程序主要是把系统内核装载到内存，启动盘必须在第一个逻辑磁道上包含引导记录。  
这512个字节的扇区又被称作是引导扇区，  
  
在系统完成加电自检后， BIOS从启动盘中将引导扇区读入到内存中。  
引导记录中包含了一些磁盘的物理特性的参数。  
  
在引导扇区被读入内存后，BIOS就能从这里读取到启动盘的物理参数。  
一旦引导记录加载完毕，BIOS就交出系统的执行控制权，跳转到引导程序 的头部执行。  
  
引导记录开头是一条无条件转移指令，它将立即跳转到地址0x03e执行引导程序，  
在引导扇区中这个引导程序将从磁盘中读出其他几个更为复杂的程序并由它们加载系统内核。   
  
Linux的 引导程序由汇编代码文件arch/i386/boot/bootsect.S生成，  
它利用对BIOS功能的调用将 arch/i386/boot/下的setup.S文件和内核映象加载到内存。  
i386的体系结构的CPU分保护模式和实模式两种，在实模式下只能使用低端的640K内存。  
  
系统在加载引导程序时CPU是处在实模式下，而现在的内核映象文件一般都超过了640K的限制，即使是经过压缩过的内核映象，  
这个内核映象文件通常是bzImage,我们在编译内核时通常要用到这个文件。  
  
由于bzImage超出了640K这一限制，所以linux设计了一个 bootsect\_helper子程序(定义在arch/i386/boot/setup.S中)，  
引导程序通过循环调用bootsect\_helper 将内核映象一块一块的装入内存，当内核加载完毕，系统跳转到setup.S的开始位置开始执行，  
setup.S仍在实模式下运行，主要功能是设置系统参数 (如：内存、磁盘等),  
并为进入保护模式做准备,最后进入到保护模式并跳转到内核映象文件的头部开始执行内核。  
  
这里提一下有关linux的 引导程序 lilo和grub，lilo和grub可以引导多个系统，  
如果机器上要装多系统的话一般都会用到它们，这一引导程序也储存在引导扇区中或者存放在主引导 记录中(MBR)，  
lilo和grub都许允用户自己配置，它们在系统安装时建立了关于系统内核占用磁盘数据块的位置对照表。  
当用户选择启动linux系统后，同样也跳转到setup.S上运行。   
  
3、内核初始化   
当setup.S执行完后，CPU进行保护模式，并开始执行内核，  
. 如果内核是经过压缩的，那么首先执行 arch/i386/boot/compressed目录下的head.S建立堆栈并解压内核映象文件，  
  然后再转入arch/i386/kernel下的 head.S。  
. 如果没有压缩则直接转到arch/i386/kernel下的head.S开始执行。  
  
arch/i386/kernel/head.S程序负责数据区(BBS)、中断描述表(IDT)、段描述表(GDT)、页表和寄存器的初始化。  
最后进入start\_kernel()模块。   
此时系统运行在内核模式(0级别)下，转入到init/main.c中的start\_kernel()。  
  
start\_kernel()继续其他方面的初始化工作，主要是初始化系统的核心数据结构，主要包括：   
  setup\_arch():执行与体系结构相关的设置。   
  trap\_init():设置各种入口地址。   
  init\_IRQ():初始化IRQ中断处理机制。   
  sched\_init():设置并启动第一个进程init\_task()。   
  softirq\_init():对软中断子系统进行初始化。   
  console\_init():初始化控制台、显示器.   
  init\_modules():初始化kernel\_module。   
  fork\_init():定义系统最大进程数.   
  最后进入rest\_init()函数并调用kernel\_thread()创建init内核线程,进行系统配置。   
  
init内核线程占用进程描述表的第一项，由它来创建其他完成系统初始他的进程。   
init内核线程首先要销定内核，然后调用do\_basic\_setup()来初始化外部设备及加载驱动程序。  
主要的初始化工作包括：   
  PCI总线初始化。   
  网络初始化。   
  文件系统初始化。   
  加载文件系统。   
  
在do\_basic\_setup()调用完成后，init()会释放初始化函数据占用的内存，  
并且打开/dev/console 设备重新定向控制台,用系统调用execve来执行用户态程序/sbin/init。  
至此，linux的内核初始化工作完成。   
  
4. 下面的工作就由用户态的/sbin/init程序来完成。  
init程序程读取/etc/inittab文件来决定它具体的工作。在inittab中比较重要的几条是：   
id:5:initdefault 决定操作系统启动时缺省的执行级别(这里说讲的是系统的运行级别，而不同于CPU的级别)   
si:sysinit:/etc/rc.d/rc.sysinit 执行/etc/rc.d/rc.sysinit的脚本。  
rc.sysinit主要的工作是 激活交换分区、检查磁盘、加载硬件模块。   
1:2345:respawn:/sbin/mingetty tty1 显示登录界面   
至此，整个系统的引导过程就完成了。