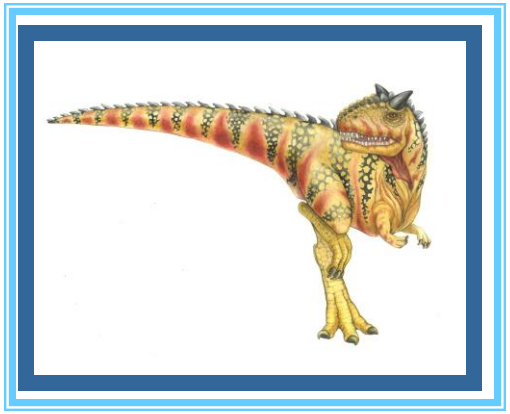
Linux 内核重建

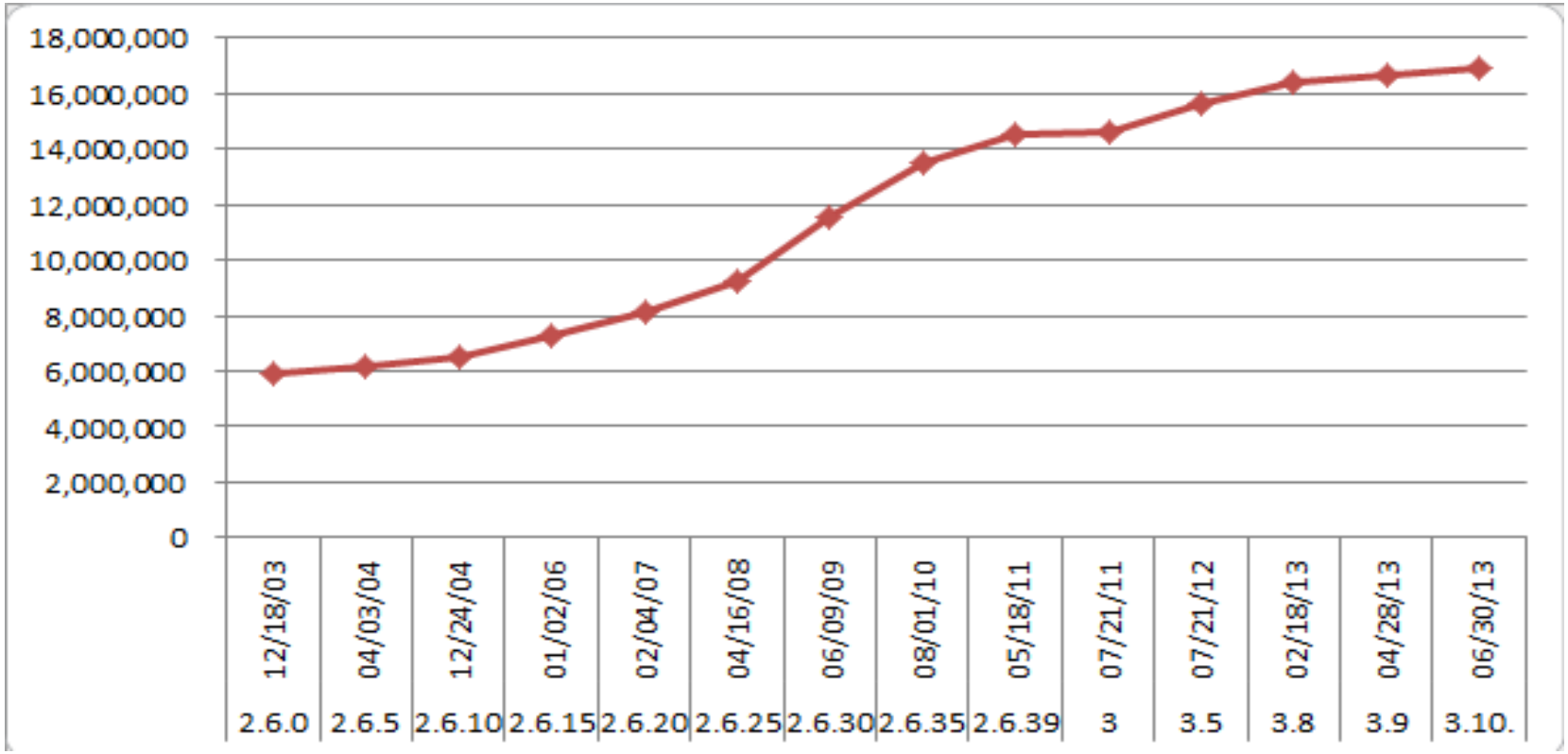


本章内容

- Linux内核
- 编译Linux内核
- /proc 虚拟文件系统
- Linux启动*

一、Linux内核

- Linux内核近两年每2个月左右发布一个新版本，http://www.kernel.org/
  - 2013年6月30日，Latest Stable Kernel: 3.10.0 (共1695多万行代码



在不到 21 年的时间内，Linux 内核已经从 10,000 多行代码增长到 1700 万行代码。

Linux内核

- “Linux内核入门是不容易，它之所以难，在于庞大的规模和复杂的层面。规模一大，就不容易现出本来面目，浑然一体，自然不容易找到着手之处；层面一多，就会让人眼花缭乱，盘根错节，怎能让人提纲挈领”——《Linux内核设计与实现》译者序

Linux内核

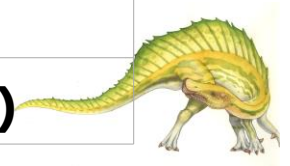
- Andre Morton：内核的学习曲线变得越来越长，也越来越陡峭。系统规模不断扩大，复杂程度不断提高。长此以往，虽然现在这一拨内核开发者对内核的掌握越发炉火纯青，但却会造成新手无法跟上内核发展步伐，出现青黄不接的断层。

Andrew Keith Paul Morton

Andrew Morton speaking at Interop, Moscow, 2008

Born	1959 England
Other names	akpm
Education	Electrical engineering
Occupation	Programmer
Employer	**Google**
Known for	**-mm tree**
Spouse(s)	Kathryn Morton
Children	Victoria Morton, Micheal Morton, Matthew Morton

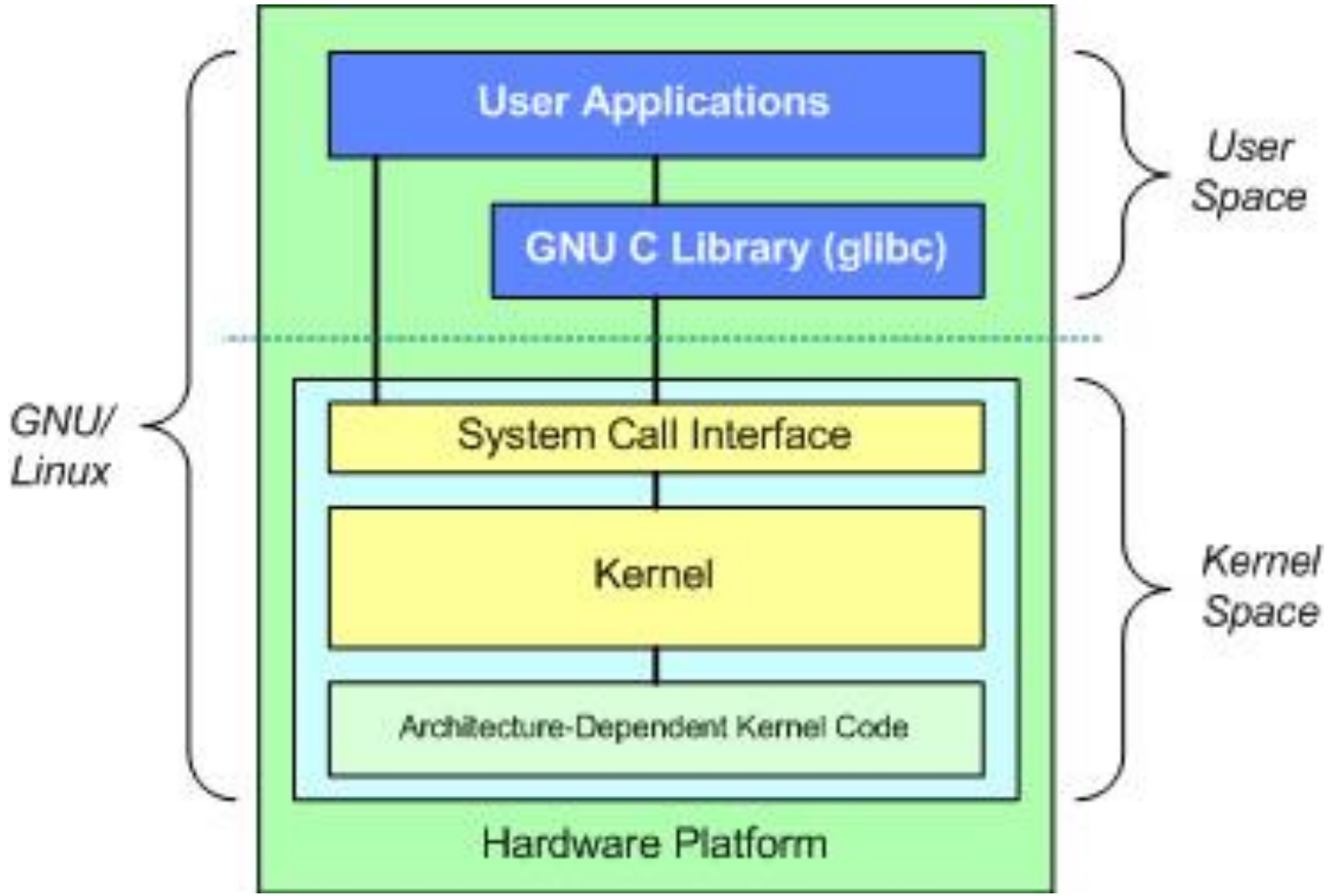
http://en.wikipedia.org/wiki/Andrew_Morton_(computer_programmer)



为什么要学习Linux内核

- 对学习操作系统原理有帮助外，学习Linux内核还有：
  - Linux内核是全世界最优秀的程序员写的
  - Linux内核结构非常好
  - 学习超大规模软件是如何设计的

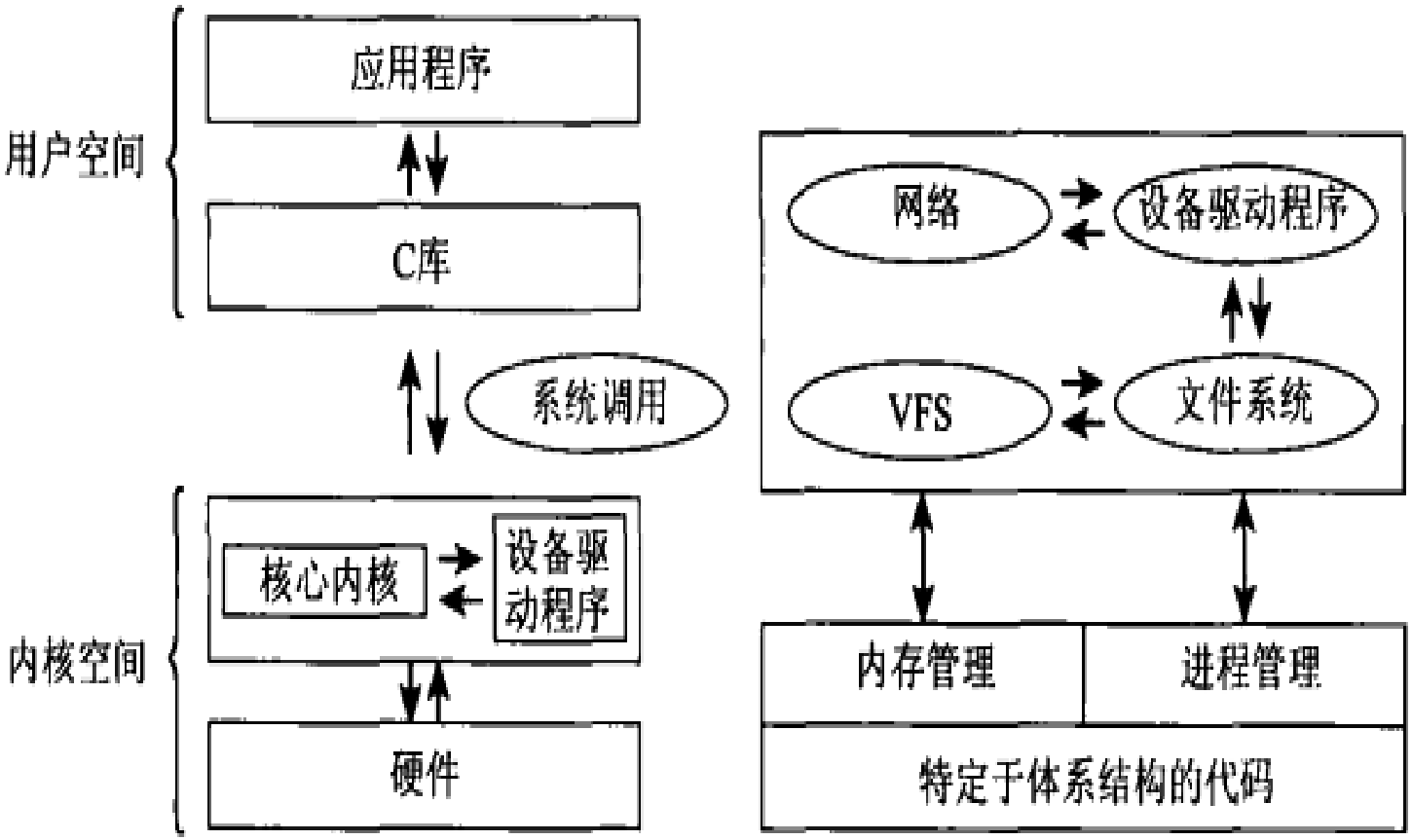
GNU/Linux 操作系统基本层次结构



Linux系统层次结构

- **GNU C Library**（glibc）：它提供了连接内核的系统调用接口，还提供了在用户空间应用程序和内核之间进行转换的机制。
- **Linux内核**—系统调用接口、进程管理、内存管理、虚拟文件系统、网络堆栈、设备驱动程序、硬件架构的相关代码 。

Linux内核(续)



Linux 内核的体系结构图

Linux内核子系统(续)

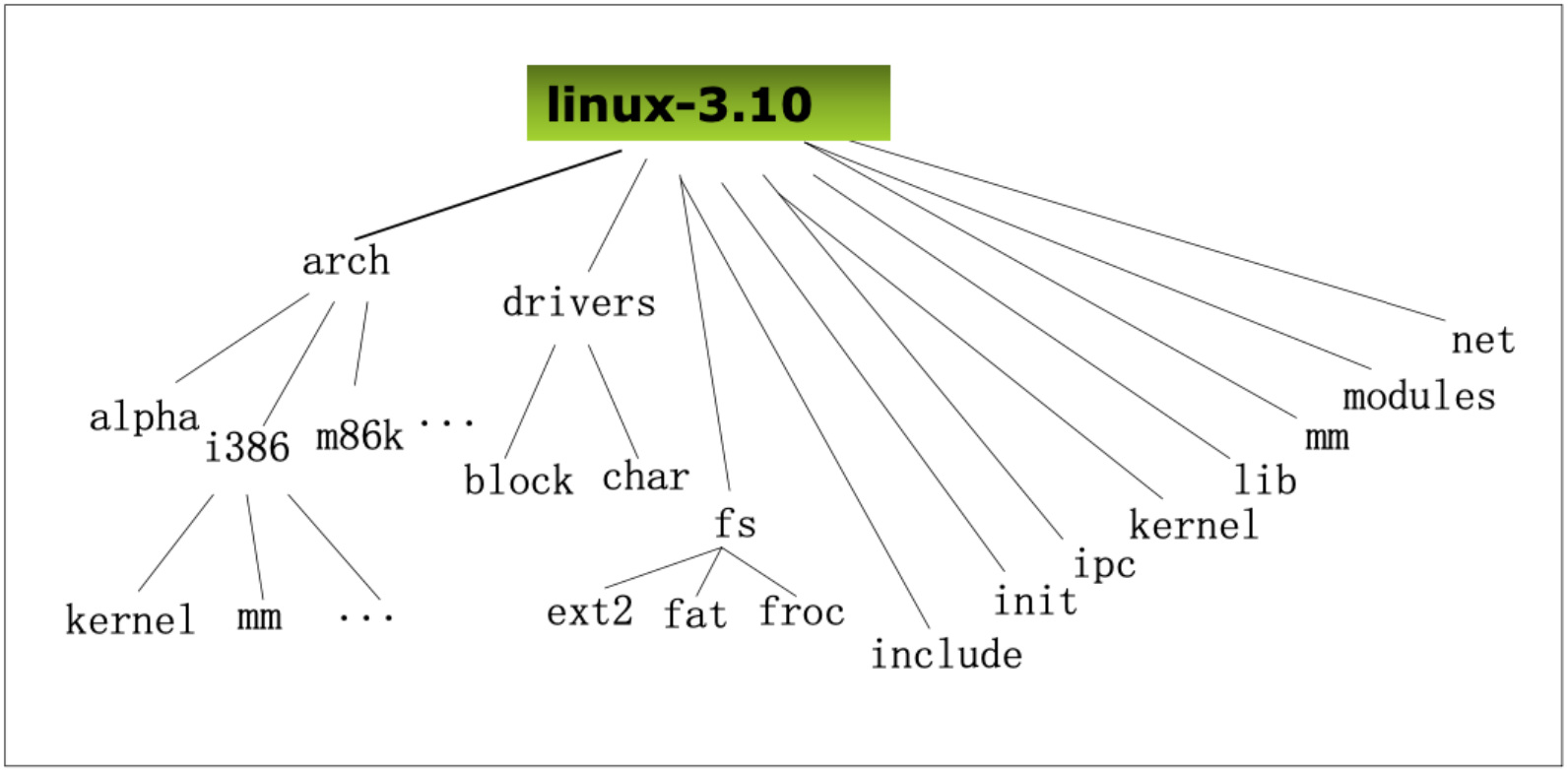
- 系统调用接口：提供了某些机制执行从用户空间到内核的函数调用。它实现了一些基本的功能，例如 read 和 write。
- 进程调度：控制着进程对CPU的访问。
- 内存管理：允许多个进程安全地共享内存区域
- 虚拟文件系统（VFS）：隐藏各种不同硬件的具体细节，为所有设备提供统一的接口。
- 网络：提供了对各种网络标准协议的存取和各种网络硬件的支持。
- 进程间通信(IPC)：支持进程间各种通信机制，包括共享内存、消息队列及管道等。

Linux内核(续)

- 内核结构：
  - 单内核/宏内核(monolithic kernel)：Unix、Linux
  - 微内核(Microkernel)：windows NT、Mac OS
- 从操作系统内核结构上看，Linux是一个单内核，Linux内核运行在单独的内核地址空间。
- Linux吸取了微内核的精华：其引以为豪的是模块化设计、抢占式内核、支持内核线程以及动态装载和卸载内核模块。

Linux内核源码目录结构

- Linux内核代码可以安装在 ~/linux-x.x.x目录下



- Linux内核源代码包含在42423个C语言和汇编等文件中，3.10代码大约有16955582行，占用430M多空间。

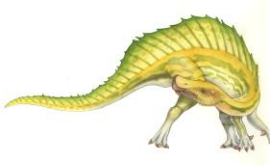
Linux内核源码目录结构(续)

- **arch**：该子目录包括了所有和体系结构相关的内核代码。它的每一个子目录都代表一种支持的体系结构，例如i386就是关于intel cpu及与之相兼容体系结构的子目录。PC机一般都基于此目录。
- **Include**: 该子包括编译内核所需要的大部分头文件。与平台无关的头文件在include/linux 子目录下，与intel cpu相关的头文件在include/asm-i386子目录下,而include/scsi目录则是有关scsi设备的头文件目录。
- **init**：该子目录包含内核的初始化代码，包含两个文件main.c和version.c。
- **mm**：该子目录包括所有独立于cpu体系结构的内存管理代码，如页式存储管理内存的分配和释放等；而和体系结构相关的内存管理代码则位于arch/*/mm/，例如arch/i386/mm/fault.c

Linux内核源码目录结构(续)

- **kernel**：主要的内核代码，此目录下的文件实现了大多数linux系统的内核函数，其中最重要的文件当属sched.c；同样，和体系结构相关的代码在arch/*/kernel中。
- **drivers**：放置系统所有的设备驱动程序;每类驱动程序又各占用一个子目录：如，/block下为块设备驱动程序，比如ide（ide.c）。设备初始化程序在drivers/block/genhd.c中的device_setup()。
- **lib**：放置内核的库代码。
- **net**：内核与网络相关的代码。
- **ipc**：这个目录包含内核的进程间通讯的代码。
- **fs**：所有的文件系统代码和各种类型的文件操作代码，它的每一个子目录支持一个文件系统，例如fat和ext2;
- **scripts**：此目录包含用于编译内核的脚本文件等。

在大多数目录下，都有一个Kconfig文件和一个Makefile文件，这两个文件都是编译时使用的辅助文件；而且，在有的目录下还有readme文件，它是对该目录下的文件的一些说明，同样有利于对内核源码的理解。



内核开发的特点

- 内核编程时不能访问C库；
- 内核编程时必须使用GNU C；
- 内核编程时缺乏像用户空间那样的内存保护机制；
- 内核编程时浮点数很难使用；
- 内核只有很小的定长堆栈；
- 由于内核支持异步中断、抢占式和SMP，因此必须时刻注意同步和并发；
- 要考虑可移植性的重要性

Linux内核源代码分析工具

- Windows平台下的源代码阅读工具
  - Source Insight
- Linux平台下的源代码阅读工具
  - SourceNavigator
- LXR（Linux Cross Reference），代码交叉检索工具。
  - http://lxr.linux.no
  - http://lxr.oss.org.cn/
  - http://os.zju.edu.cn/newlxr/source/

内核镜像

- Linux系统引导过程使用内核镜像，/boot目录下文件名称如：vmlinuz-2.6.15.5：
  - 普通内核镜像：zImage（Image compressed with gzip），大小不能超过512k
  - 大内核镜像：bzImage（big Image compressed with gzip），包含了大部分系统核心组件：系统初始化、进程调度、内核管理模块

编译Linux内核(Vmware Play, ubuntu 13.04 ， 3.11.4 kernel)

ubuntu编译内核步骤:

- sudo apt-get install libncurses5-dev //如果没有ncurses库，则安装
- 下载内核源代码：linux-3.11.4.tar.xz 文件。下载地址：
  http://www.kernel.org/ 或 http://os.zju.edu.cn/newlinux/files/jijiangmin
- 部署内核代码
  - 把内核代码文件linux-3.11.4.tar.xz 存放在主目录（~）中。
  - tar -xvf linux-3.11.4.tar.xz //解压内核包，生成的内核源代码放在linux.3.11.4目录中
  - cd linux-3.11.4
  - cp /boot/config-`uname -r` .config //使用系统的原配置文件
    - 或 cp /boot/config-<Tab> .config // <Tab> 为<Tab> 键
- make menuconfig // 同时生成.config文件

配置内核

- 进行配置时，大部分选项可以使用其缺省值，只有小部分需要根据用户不同的需要选择。例如，如果硬盘分区采用ext2文件系统（或ext3文件系统），则配置项应支持ext2文件系统（ext3文件系统）。又例如，系统如果配有SCSI总线及设备，需要在配置中选择SCSI卡的支持。
- 对每一个配置选项，用户有三种选择，它们分别代表的含义如下：
  - “<*>”或“[*]” — 将该功能编译进内核
  - “[ ]” — 不将该功能编译进内核
  - “[M]” — 将该功能编译成可以在需要时动态插入到内核中的模块
    - 与核心其它部分关系较远且不经常使用的部分功能代码编译成为可加载模块，有利于减小内核的长度，减小内核消耗的内存，简化该功能相应的环境改变时对内核的影响。许多功能都可以这样处理，例如像上面提到的对SCSI卡的支持，等等。

- make 或 make -j4
  //虚拟机上需要2个多小时
- sudo make modules_install
- sudo make install
- 查看启动选项
  - sudo gedit /boot/grub/grub.cfg
- 重新启动
  - sudo reboot //启动时忽略错误信息提示
- 启动后查看内核版本号
  - uname -r

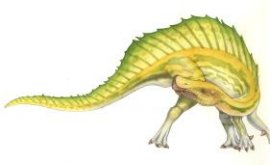
    3.11.4

Grub 配置

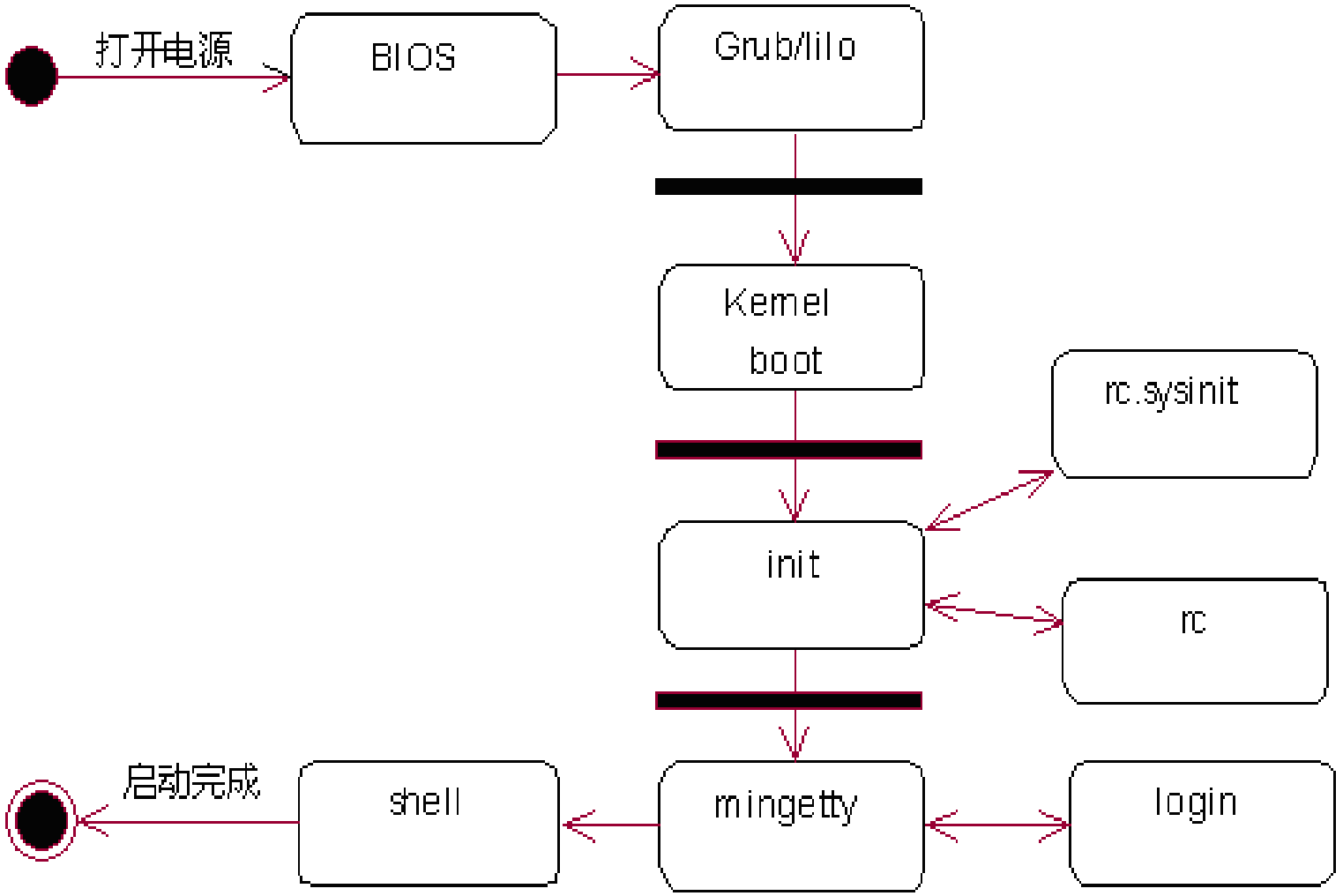
```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda2
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
Hiddenmenu

title GreatTurbo Enterprise Server (2.6.18-8.2PAE)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-8.2PAE ro root=LABEL=/1 rhgb quiet
        initrd /initrd-2.6.18-8.2PAE.img

title GreatTurbo Enterprise Server-base (2.6.18-8.2)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-8.2 ro root=LABEL=/1 rhgb quiet
        initrd /initrd-2.6.18-8.2.img
```

四、Linux系统启动*

- Linux系统的启动过程大致可分成以下几个阶段：
  0. 硬件检测（自检）；
  1. 由BIOS加载操作系统引导装入程序；
  2. 由操作系统引导装入程序加载操作系统内核；
  3. 内核代码解压缩；
  4. 内核初始化；
  5. 生成init进程；
  6. 系统初始化，shell命令文本的执行；
     - 通过/etc/inittab文件进行初始化
     - 如设置键盘、字体、装载模块，设置网络等。
  7. 生成各终端进程；
  8. 用户登录。
     - 执行/bin/login程序



Linux系统启动



Linux系统启动过程

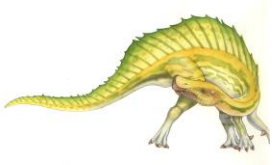
Linux启动过程

- “只读内存”（read-only memory，缩写为ROM），**开机程序被刷入ROM芯片**
- 计算机通电后，第一件事就是读取它
- 这块芯片里的程序叫做"**基本輸出輸入系統**"（Basic Input/Output System），简称为BIOS

Linux启动过程

- BIOS程序首先检查，计算机硬件能否满足运行的基本条件，这叫做"硬件自检"（Power-On Self-Test）

```
Diskette Drive B  : None                Serial Port(s)    : 3F0 2F0
Pri. Master Disk  : LBA,ATA 100,  250GB Parallel Port(s)  : 370
Pri. Slave  Disk  : LBA,ATA 100,  250GB DDR at Bank(s)    : 0 1 2
Sec. Master Disk  : None
Sec. Slave  Disk  : None

Pri. Master Disk  HDD S.M.A.R.T. capability ... Disabled
Pri. Slave  Disk  HDD S.M.A.R.T. capability ... Disabled

PCI Devices Listing ...
Bus  Dev  Fun  Vendor Device  SVID  SSID Class  Device Class              IRQ
---------------------------------------------------------------------------
  0   27    0  8086   2668    1458  A005  0403  Multimedia Device          5
  0   29    0  8086   2658    1458  2658  0C03  USB 1.1 Host Cntrlr        9
  0   29    1  8086   2659    1458  2659  0C03  USB 1.1 Host Cntrlr       11
  0   29    2  8086   265A    1458  265A  0C03  USB 1.1 Host Cntrlr       11
  0   29    3  8086   265B    1458  265A  0C03  USB 1.1 Host Cntrlr        5
  0   29    7  8086   265C    1458  5006  0C03  USB 1.1 Host Cntrlr        9
  0   31    2  8086   2651    1458  2651  0101  IDE Cntrlr                14
  0   31    3  8086   266A    1458  266A  0C05  SMBus Cntrlr              11
  1    0    0  10DE   0421    10DE  0479  0300  Display Cntrlr             5
  2    0    0  1283   8212    0000  0000  0180  Mass Storage Cntrlr       10
  2    5    0  11AB   4320    1458  E000  0200  Network Cntrlr            12
                                                ACPI Controller            9
```



Linux启动过程

- 硬件自检完成后，BIOS把控制权转交给下一阶段的启动程序。
- BIOS需要有一个外部储存设备的排序，排在前面的设备就是优先转交控制权的设备。这种排序叫做"启动顺序"（Boot Sequence）。
- 打开BIOS的操作界面，里面有一项就是"设定启动顺序"。

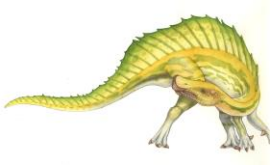
Linux启动过程

- BIOS按照"启动顺序"，把控制权转交给排在第一位的储存设备。
- 这时，计算机读取该设备的第一个扇区，也就是读取最前面的512个字节。如果这512个字节的最后两个字节是0x55和0xAA，表明这个设备可以用于启动；如果不是，表明设备不能用于启动，控制权于是被转交给"启动顺序"中的下一个设备。
- 这最前面的512个字节，就叫做"主引导记录"（Master boot record，缩写为MBR）

Linux启动过程

- "主引导记录"只有512个字节，放不了太多东西。它的主要作用是，告诉计算机到硬盘的哪一个位置去找操作系统。
- 主引导记录由三个部分组成：

（1）第1-446字节：调用操作系统的机器码。
（2）第447-510字节：分区表（Partition table）。
（3）第511-512字节：主引导记录签名（0x55和0xAA）。



Linux启动过程

- 硬盘分区有很多好处。考虑到每个区可以安装不同的操作系统，"主引导记录"因此必须知道将控制权转交给哪个区。
- 分区表的长度只有64个字节，里面又分成四项，每项16个字节。所以，一个硬盘最多只能分四个一级分区，又叫做"主分区"。

（1）第1个字节：如果为0x80，就表示该主分区是激活分区，控制权要转交给这个分区。四个主分区里面只能有一个是激活的。

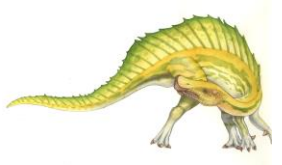
（2）第2-4个字节：主分区第一个扇区的物理位置（柱面、磁头、扇区号等等）。

（3）第5个字节：主分区类型。

（4）第6-8个字节：主分区最后一个扇区的物理位置。

（5）第9-12字节：该主分区第一个扇区的逻辑地址。

（6）第13-16字节：主分区的扇区总数。



Linux启动过程

- 计算机的控制权就要转交给硬盘的某个分区了，这里又分成三种情况。
  - 四个主分区里面，只有一个是激活的。计算机会读取激活分区的第一个扇区，叫做“卷引导记录”（Volume boot record，缩写为VBR）。“卷引导记录”的主要作用是，告诉计算机，操作系统在这个分区里的位置。
  - 四个主分区已经不够了，需要更多的分区。但是，分区表只有四项，因此规定有且仅有一个区可以被定义成"扩展分区”
  - 计算机读取“主引导记录”前面446字节的机器码之后，不再把控制权转交给某一个分区，而是运行事先安装的启动管理器“（boot loader），由用户选择启动哪一个操作系统。Linux环境中，目前最流行的启动管理器是Grub。

Linux启动过程


```
$ ls /boot

config-3.2.0-3-amd64
config-3.2.0-4-amd64
grub
initrd.img-3.2.0-3-amd64
initrd.img-3.2.0-4-amd64
System.map-3.2.0-3-amd64
System.map-3.2.0-4-amd64
vmlinuz-3.2.0-3-amd64
vmlinuz-3.2.0-4-amd64
```

Linux启动过程

- /sbin/init,　pid = 1

Linux启动过程

- init进程首先读取文件 /etc/inittab，它是运行级别的设置文件。如果你打开它，可以看到第一行是这样的：

id:2:initdefault:

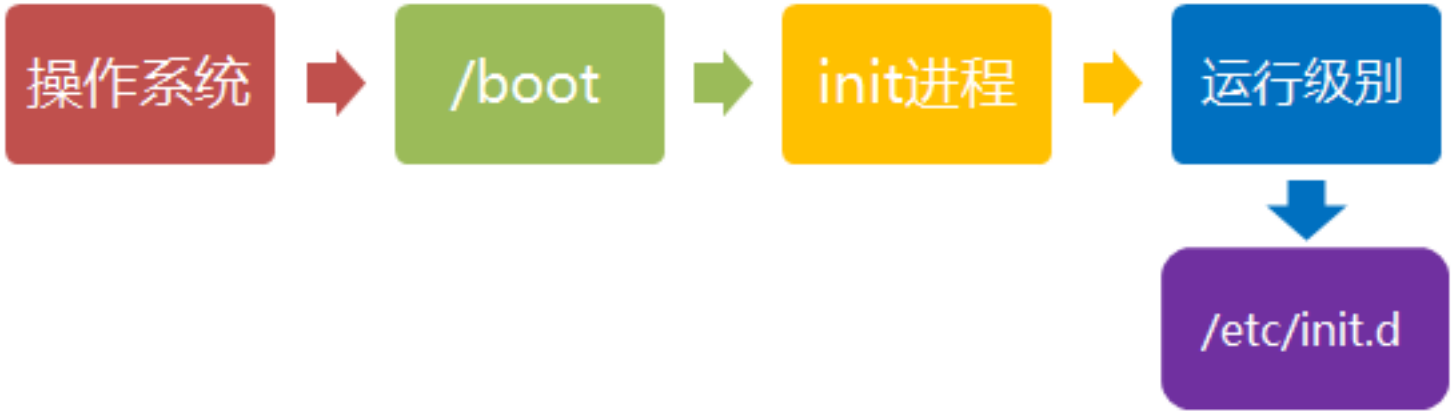
ls /etc/rc2.d

/etc/rc0.d
/etc/rc1.d
/etc/rc2.d
/etc/rc3.d
/etc/rc4.d
/etc/rc5.d
/etc/rc6.d

README
S01motd
S13rpcbind
S14nfs-common
S16binfmt-support
S16rsyslog
S16sudo
S17apache2
S18acpid

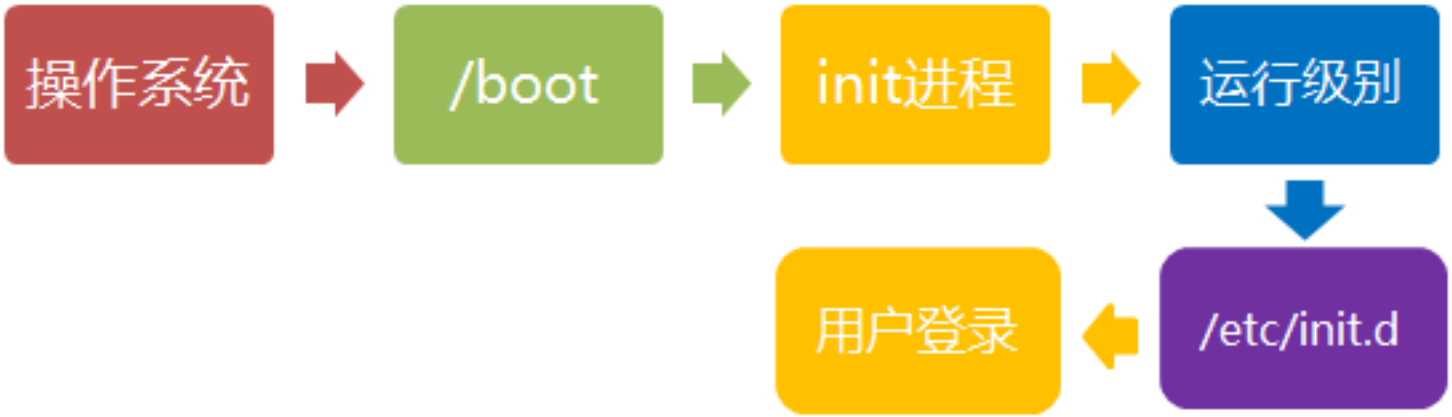
Linux启动过程

```
$ ls -l /etc/rc2.d

README
S01motd -> ../init.d/motd
S13rpcbind -> ../init.d/rpcbind
S14nfs-common -> ../init.d/nfs-common
S16binfmt-support -> ../init.d/binfmt-support
S16rsyslog -> ../init.d/rsyslog
S16sudo -> ../init.d/sudo
S17apache2 -> ../init.d/apache2
S18acpid -> ../init.d/acpid
```



Linux启动过程

- 命令行登录：init进程调用getty程序（意为get teletype），让用户输入用户名和密码。输入完成后，再调用login程序，核对密码。如果密码正确，就从文件 /etc/passwd 读取该用户指定的shell，然后启动这个shell。



内核引导阶段

- Bootsect
- setup.S
- head.S
- main.c

Init阶段

- init有两项重要的工作：
  - 解释/etc/inittab文件，定义runlevels
  - 运行系统初始化脚本，/etc/rc.d目录定义各个runlevels上运行的服务

inittab文件的格式

- inittab文件中每一行有下列格式

  id:runlevel:action:process

  - id：配置记录标识符，tty为序号
  - runlevel：运行级
  - action：process的执行方式
  - process：要运行的程序的路径和命令选项

8种执行方式

- sysinit：提供init的路径
- respawn：每当一个命令结束后，就重启该命令
- askfirst：类似respawn，但是要先问一下用户
- wait：阻塞式命令，init要等待其运行完毕
- once：只运行一次，不必等待
- ctrlaltdel：三键齐按时，要执行的命令
- shutdown：系统关闭时执行
- restart：系统重启时执行，通常就是init

- 一个可能的inittab如下（id和runlevel都为空）：

```
::sysinit:/etc/init.d/rcS
```

设置/etc/init.d/rcS作为系统初始化文件

```
::respawn:/sbin/getty 115200 ttyS0
```

在串口（115200波特率）启动一个登录会话

```
::respawn:/control-module/bin/init
```

启动控制模块定制的系统初始化脚本

```
::restart:/sbin/init
```

设置/sbin/init为重启时运行的命令

```
::shutdown:/bin/umount -a -r
```

系统关闭时，运行umount

init阶段工作

1、确定用户登录模式

- 在“/etc/inittab”中列出了如下所示的登录模式，主要有单人维护模式、多用户无网络模式、文字界面多用户模式、X-Windows多用户模式等。其中的单人维护模式（run level为1）是类似于Windows中的“安全模式”，在这种情况下，系统不加载复杂的模式从而使系统能够正常启动。

cat /etc/inittab

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode （文本界面启动模式）
#   4 - unused
#   5 - X11 （图形界面启动模式）
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
……
```

init阶段工作

2、执行脚本/etc/rc.d/rc.sysinit

- 将Linux的主机信息读入Linux系统，其内容就是文件“/etc/rc.d/rc.sysinit”中的。查看此文件可以看出，在这里确定了默认路径、主机名称、“/etc/sysconfig/network”中所记录的网络信息等。

```
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
```

init阶段工作

3、启动内核的外挂模块及各运行级的脚本

- 读取模块加载配置文件（/etc/modules.conf），以确认需要加载哪些模块。接下来会根据不同的运行级（run level），通过带参数（运行级）运行“/etc/rc.d/rc”脚本，加载不同的模块，启动系统服务。init进程会等待（wait）“/etc/rc.d/rc”脚本的返回。

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

init阶段工作

4. 进入用户登录界面

- 系统还需要配置一些异常关机的处理部分。
- 最后通过“/sbin/mingetty”打开几个虚拟终端（tty1~tty6），用于用户登录。
- 如果运行级为5（图形界面启动），则运行xdm程序，给用户提供xdm图形界面的登录方式。

例：IA32 Linux* Boot

IA32 as an example

- Power on CPU
  - Registers will be set to default values
  - cs:eip is initialized to 0xffffffff0
- BIOS
  - BIOS has code at 0xffffffff0(EPROM or similar)
  - Initialize devices
  - Prepare tables/info for OS (E820, MPS …)
  - Copy boot device's first sector(MBR) to 0x7c00
  - Jump to 0x7c00
- Boot loader
  - LILO/GRUB
  - Boot loader loads kernel image and initrd (ramdisk)
  - Put kernel image into specific physical address
  - Jump to kernel real mode initialization code

Linux Boot

- Real mode initialization (setup.S)
  - Entry point is in 0x200 offset of kernel image
  - Initialize memory (E820 …)
  - ...
  - Switch to protected mode
  - Copy kernel to 0x100000
  - Jump to startup_32
- Protected mode initialization(head.S::startup_32)
  - Entry point is at physical address 0x100000
  - Clear BSS
  - Copy boot parameters to kernel data
  - Initialize registers/GDT/IDT …
  - Jump to C code 'start_kernel'

Linux Boot

- Start_kernel (main.c)
  - Setup_arch (architecture dependent setup,)
  - Setup direct memory mapping/initialize buddy …
  - Kernel subsystem initialize/...
  - Boot other CPUs
  - Populate_rootfs (from initrd)
  - Initcalls initialize
  - Prepare files for init (0, 1, 2)
  - Mount root fs
  - Fork a new task and execute init process (/sbin/init) in the task

Linux Boot

- User space init (init process)
  - Read /etc/inittab （系统配置文件）
  - Execute init scripts (/etc/rc.d/rc*.d/, …) per current run level
  - Init script will mount fs/start daemons/start x …
  - Start gettty process, open console
  - Type user name/password, login check them, and start shell
- Run level
  - 1 – single user mode
  - 3 – multiple user mode
  - 5 – X mode
  - 6 – reboot