

## 第9章 配置 FreeBSD 的内核

(翻译中出现的任何问题或错误，请广大读者及时反馈给我：freebsdhandbook@163.com)

### 9.1 概要

内核是 FreeBSD 的核心。它用来管理内存，执行安全控制，网络，磁盘访问等等。而有时你需要重新配置和编译你的内核。

读完这章，你将了解到：

- 为什么需要建构一个定制的内核。
- 如何写一个内核配置文件，或修改已存在的配置文件。
- 如何使用内核配置文件创建和建构一个新的内核。
- 如何安装一个新内核。
- 如何在/dev 中创建设备文件的入口。
- 如何解决一些常见的问题。

### 9.2 为什么要定制一个的内核？

定制内核基本上是每一个 UNIX 用户必须经历的一关。这样做将会对你的 FreeBSD 系统带来很多好处。不象 GENERIC 内核，它需要支持很广泛的硬件设备，而定制的内核将只支持你机器的硬件设备，这样会带来很多好处：

1. 快速启动。既然系统内核只检测你机器上的硬件，那它所花费的启动时间将大大减少。
2. 较少的内存使用。一个定制的内核会比 GENERIC 内核使用较少的内存。这一点非常重要，因为内核在处理时必须经常使用内存。所以，一个定制的内核对于内存较少的机器来说是非常有用的。
3. 额外的硬件支持。一个定制的内核允许你加入象声卡之类设备的支持，而这恰恰是 GENERIC 内核所做不到的。

### 9.3 建立并安装一个定制的内核

首先,让我们先来看一下建立内核的目录。所有提到的目录都在 `/usr/src/sys` 中,也可以通过 `/sys` 访问。有许多子目录充当了内核的不同部分。但最重要的是 `/arch/conf`,你可以在这儿配置内核,然后进行编译。这里的 `arch` 也可能是 `i386`, `alpha`, `pc98`。在一个特殊的架构目录内,所有的东西都是为这个架构服务的;其它部分的代码是与所有平台共享的。注意一下这个目录的逻辑结构,所有支持的硬件驱动程序、文件系统、参数选项,都放在各自的目录下。另外,所有放在 `i386` 目录下的是只跟 PC 硬件有关的,而 `i386` 目录之外的其它目录,则是 FreeBSD 可能移植到的平台会共享到的部分。

**注意:** 如果你的系统没有这个 `/usr/src/sys` 目录,那么内核源代码就没有被安装。最快速的安装方式是以 `root` 登陆,然后运行 `/stand/sysinstall`,选择 `configure`→`Distributions`→`src`→`sys`。

接着,切换到 `arch/conf` 目录,拷贝 `GENERIC` 配置文件,并给这个文件起一个容易辨认的名称,它就是你的内核名称。例如:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

通常,这个名称是大写的,如果正维护着多台不同硬件的 FreeBSD 机器,以你机器的域名来命名是非常好的主意。我们把它命名为 `MYKERNEL` 就是这个原因。

**注意:** 你必须以 `root` 帐户登陆再执行下面的命令,否则你会没有权限,而导致错误。

现在,用你喜欢的文本编辑器编辑 `MYKERNEL`。如果你是初学者,那唯一能用的编辑器可能就是 `vi`,它由于太复杂而不在这儿介绍了,但在参考书目中有许多书会介绍到它。然而,FreeBSD 中最容易的编辑器是 `ee`,如果你是一个工程师,它是一个非常好的选择。你可以很自由地改变注释行来反映你的配置情况,或你在 `GENERIC` 中已经做的变化。如果你在 `Sunos` 或其它 BSD 系统下定制过内核,那这个文件中的绝大部分将对你非常熟悉。如果你使用的是诸如 `DOS` 这样的系统,那 `GENERIC` 配置文件就看起来非常困难,所以在下面的配置文件章节将慢慢地、仔细地进行介绍。

**注意:** 确信经常检查 `/usr/src/UPDATING`,在你执行任何修改之前,记得要用最新发布的源代码来同步你的源代码树。在这个文件中所有重要的升级都要记下来。`/usr/src/UPDATING` 总是符合你的 FreeBSD 源代码版本,而且总是比手册所说的信息更加精确。

当你完成以上步骤之后,如果你使用的 FreeBSD 是 4.0 之前的版本的话,就执行下面的

命令进行编译和安装内核；如果你使用的是 4.0 版或是之后的版本，你的 `/usr/src/` 目录可能已经包含了 `sys/` 子目录了。

**注意：**如果你设法从一个旧版本的 FreeBSD 升级你的内核，你可能必须从你得到新内核源代码的地方，找到新版的配制文件，重建并安装它。

```
# /usr/sbin/config MYKERNEL
# cd ../../compile/MYKERNEL
# make depend
# make
# make install
```

如果你刚升级到一个比较新的版本，确信你已经重建了整个系统，然后运行下面的命令：

```
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

如果你还没有升级你的源代码树，那你应当按顺序执行：

```
config,make depend,make,make install
```

**警告：**如果你已经升级了你的源代码，你必须使用 `make buildkernel` 方法来定制你的内核。否则，用旧的工具来定制内核，可能会引起错误。如果你已经升级了源代码，不要使用 `config/make` 来定制内核。

新的内核将会被拷贝到 `/kernel` 目录下，而旧的内核将会被移到 `/kernel.old`。现在，关闭系统，然后重新用你的内核启动系统。如果发生错误，在这章结尾会有一些故障的解决办法。一旦你的新内核不能启动，请务必读一下有关如何恢复的章节。

**注意：**如果你已经加入了新的设备（如声卡），你可能在使用之前，要先在你的 `/dev` 目录下加入这些设备节点。

### 9.4 配置文件

内核设置的格式是很简单的，每一行包含了一个关键词(keyword)与一个或多个参数，而大多数的设置都只包含一个参数。`#`号之后的文字都是注释，会被程序忽略掉。下面的每个小节，将依次介绍每个列在 `GENERIC` 中的参数，虽然各相关主题(如网络)的关键词会放在同一小节，但是这些关键词可能位于 `GENERIC` 的很多地方。详细地列出各个选项(option)，而 `LINT` 列出了绝大部分的选项(options)，比起在同一目录下的 `GENERIC`，有更详细的解释。

如果你不能确定某一行设定的目的是否必要，请先看看 LINT。

目前内核在处理各个选项上正转到一个比较好的模式。一般地，设置文件中的各个选项都转换到内核程序的 Makefile 中，属于 CFLAGS 的一个 -D 参数。时间一长，就造成了一个选项处理机制的问题，没有人知道在哪个文件中参考使用了那个选项。

在新的机制中，使用 #ifdef 来参考选项的程序代码是存放在由执行 config 时所产生的 opt\_foo.h。由 config 所产生的有效的选项清单存储在两个文件里：独立于系统架构的选项放在 /sys/conf/options，与系统架构有关的选项则放在 /sys/arch/conf/options.arch，其中 arch 的一个例子便是 i386。

**数字的引号限制(Quoting numbers):** 目前所有 FreeBSD 版本，包括 3-stable 版，其内核设置文件中如果有任何属于文字形式的数字(如 i386)，必须用双引号引起来，不然执行 config 时会出错。

如果数字是表示数目、个数，如 maxuser 64 这个设置，则不要加双引号。目前 FreeBSD CURRENT 版本已经去掉上述限制，不论是哪种形式的数字都不需要加双引号。本章的例子，在文字形式的数字前后仍加了双引号(“ ”)，如果你使用 FreeBSD CURRENT，请自动去除双引号。

下面是一个带有很多额外注释的 GENERIC 内核配置文件的例子。这个例子与 /usr/src/sys/i386/conf/GENERIC 非常相似。有关内核配置的最详细的选项，请参看 /usr/src/sys/i386/conf/LINT。

```
#
# GENERIC -- Generic kernel configuration file for FreeBSD/i386
#
# For more information on this file, please read the handbook section on
# Kernel Configuration Files:
#
#   http://www.FreeBSD.org/handbook/kernelconfig-config.html
#
# The handbook is also available locally in /usr/share/doc/handbook
# if you've installed the doc distribution, otherwise always see the
# FreeBSD World Wide Web server (http://www.FreeBSD.ORG/) for the
```

```
# latest information.  
  
#  
  
# An exhaustive list of options and more detailed explanations of the  
# device lines is also present in the ./LINT configuration file. If you are  
# in doubt as to the purpose or necessity of a line, check first in LINT.  
  
#  
  
# $FreeBSD: src/sys/i386/conf/GENERIC,v 1.246 2000/03/09 16:32:55 jlemon Exp  
$
```

下面这个选项在每个内核中都要有：

```
machine      i386
```

上面的选项指出了在你的系统中所用的 CPU 类型。你可以使用多个 CPU 类型（例如，你不确信你使用的是 I586\_CPU 还是 I686\_CPU），然而，对于一个定制的内核，最好是只指定你所拥有的 CPU 类型。如果你不能确定你的 CPU 类型，你可以使用 `dmesg` 命令来看看你的启动信息。

Alpha 结构的机器使用下面的选项：

```
cpu          EV4  
  
cpu          EV5
```

如果你正使用一台 Alpha 机器，你应当使用上面的 CPU 类型。

```
ident        GENERIC
```

`ident` 是一个内核的标识符。你应该自己命名一个容易辨认的名称，以有别于 `GENERIC` 的名称 `GENERIC`。放在 `ident` 后的参数，将在你用这个内核启动时显示在屏幕上。如果你同时设置了几个内核，采用不同的名称是个不错的主意。

**注意**，象 `machine` 与 `cpu` 这两个设置。如果你的内核名称包含数字，请记得用双引号把它括起来。内核名称将会使用 `-D` 参数传给编译器，所以不要用像 `DEBUG` 或是其它可能干扰编译器的机器、CPU 名称，如 `vax`。

```
maxusers     32
```

这个设置的大小值定义了重要的 `system tables`。这个数值粗略地假设你的机器同时会有多少使用者。然而，在一般情况下，你最少要设置 4 个以上，特别是你要执行 X Window 或是编译程序。原因是 `maxusers` 值决定了系统同时可有多少个进程（`processes`），其算法是  $20 + 16 * \text{maxusers}$ 。如果你设置 `maxusers` 值是 1，则你的系统只能同时存在 36 个

processes，包括 18 个(或更多)系统启动要占去的 processes；如果你执行了 X Window，则又要用掉 15 个以上。甚至阅读一个 man page 也会使用九个 processes 来过滤、解压缩、然后显示文件。设置 maxusers=4，则系统可以同时有 84 个 processes，对任何人应该都是足够的。当你执行程序时，得到像是“proc table full”这样的错误信息，或者你要建一个同时会有很多人来访问的网站(如 Walnut Creek CDRom 的 FTP)时，你就要增加这个设置的值，然后重新编译内核。

**注意:** maxuser 这个参数并不限制可以登陆你系统的用户的数目。它只是系统中使用者可以最多执行的程序的个数。有一个关键词的值则真的限制了可以同时远程连接(remote logins)的人数：**pseudo-device pty 16**。

```
# Floating point support - do not disable.  
device      npx0      at nexus? port IO_NPX irq 13
```

npx0 是连接 FreeBSD 中浮点运算处理器的一个接口。不论你有硬件的浮点运算处理器还是使用软件仿真，都需要这个设置。这个设置并不是可有可无的。

```
# Pseudo devices - the number indicates how many units to allocate.  
pseudo-device loop      # Network loopback
```

loop 是一个通用的 TCP/IP 接口。如果你用 telnet 或 ftp 连到 localhost(等同于 127.0.0.1)，则该连接会通过这个虚拟设备连回来。这是一定要有的设置，请不要去掉。

```
#makeoptions    DEBUG=-g      #Build kernel with gdb(1) debug symbols  
options         MATH_EMULATE  #Support for x87 emulation
```

如果你的计算机没有浮点运算处理器(386 或 486SX)，你可以加入这行，使得内核提供软件仿真的浮点运算处理器。如果你用的是 486DX 或是 386SX、486SX(还加装了 387、487 芯片)或更高的 (Pentium、Pentium II 等)则不需要这行设置。

**注意:** 这个仿真的浮点运算处理器并不是很精确。如果你没有浮点运算器，还需要较高的精度，你可以改用 GPL\_MATH\_EMULATE 参数，这将会使用 GNU 的浮点运算仿真器。至于为什么这个仿真器不是系统默认值，是因为 GNU 使用许可的关系。

```
options         INET      #InterNETworking
```

提供网络支持。就算你不打算连上网络，你还是要留着这个选项。对于绝大部分的程序，这个选项是一定要有的。

```
options          INET6          #IPv6 communications protocols
```

这个启用 IPv6 通讯协议。

```
options          FFS           #Berkeley Fast Filesystem
```

```
options          FFS_ROOT      #FFS usable as root device [keep this!]
```

最基本的硬盘文件系统。如果你要从硬盘启动，就留着。

```
options          MFS           #Memory Filesystem
```

```
options          MD_ROOT      #MD is a potential root device
```

Memory-mapped 文件系统。提供 RAM disk，以供需要快速访问的资料或是暂存资料用。如果你分了很大的 swap 空间，使用这个选项可以给你不少好处。把 /tmp 挂到 MFS 分区是一个相当好的想法，因为不少程序都会在此暂存资料。要把 /tmp 挂到 MFS RAM disk，可以修改 /etc/fstab，加入以下一行：

```
/dev/ad1s2b     /tmp mfs rw 0 0
```

现在重新启动，或是键入 mount /tmp 命令：

```
options          NFS           #Network Filesystem
```

```
options          NFS_ROOT      #NFS usable as root device, NFS required
```

网络文件系统(Network Filesystem, NFS)，除非你要从网络上的其它机器加载目录，不然你可以用 # 号注释掉这行设置。

```
options          MSDOSFS      #MSDOS Filesystem
```

MS-DOS 文件系统。除非你要在启动时挂上一个 DOS 格式的硬盘，不然你可以放心地把这行注释掉。如前所述，在你第一次挂上一个 DOS 分区时，内核将会自动加载模块来支持它。此外，`mttools` 是个相当不错的软件(可在 ports 里面找到)，可以让你在访问 DOS 磁盘时，不需要挂入或卸载软盘(而且也不需要 MSDOSFS 的支持)。

```
options          CD9660       #ISO 9660 Filesystem
```

```
options          CD9660_ROOT  #CD-ROM usable as root, CD9660 required
```

CD-ROM 使用的 ISO 9660 文件系统。如果你没有光驱，或是很少用光驱，可以注释掉这一行(内核会在第一次挂入时动态加载模块以支持它)。音乐 CD 则不会用到这个文件系统。

```
options          PROCFS       #Process filesystem
```

Process filesystem。这是一个虚拟的文件系统，挂在 /proc 下，允许一些程序，像 ps 来读取资料，提供你正在执行的 processes 的信息。

```
options          COMPAT_43      #Compatible with BSD 4.3 [KEEP THIS!]
```

使系统兼容 4.3BSD。不要去掉这一行，不然有些程序将无法正常运行。

```
options          SCSI_DELAY=15000  #Delay (in ms) before probing SCSI
```

这行设置告诉内核等待 15 秒钟，以供 SCSI 控制器扫描你计算机上的 SCSI 设备。如果你只有 IDE 硬盘，你可以不理睬这个设置，不然你可能会想要降低这个值，也许会降到五秒，以增加启动的速度。如果你发现降低后，FreeBSD 无法正确辨认你的 SCSI 设备，那么你应该提高这个值，延长等待时间。

```
options          UCONSOLE          #Allow users to grab the console
```

允许使用者找到 console 信息，对 X Window 很有用。举例来说，你可以输入 xterm -C 来打开一个 console xterm，这个窗口将显示任何 write、talk 等命令的信息，以及你发出去的信息。当然，kernel 产生的信息也会在这里出现。

```
options          USERCONFIG          #boot -c editor
```

这个选项允许你从启动菜单启动配置编辑器。

```
options          VISUAL_USERCONFIG  #visual boot -c editor
```

这个选项允许你从启动菜单启动虚拟配置编辑器。

```
options          KTRACE              #ktrace support
```

这个选项启用内核进程跟踪，在调试时很有用。

```
options          SYSVSHM             #SYSV-style shared memory
```

提供 System V Shared memory(SHM)的支持，最常用到 SHM 的应该是 X Window 的 XSHM 延伸，不少绘图相关程序(像影片播放程序 XAnim 与 Linux DOOM 游戏)会自动使用 SHM 来提供额外的速度。如果你要使用 X Window，你最好加入这个选项。

```
options          SYSVSEM             #SYSV-style semaphores
```

支持 System V semaphores，不常用到，只在 kernel 中占用几百字节的空间。

```
options          SYSVMSG             #SYSV-style message queues
```

支持 System V messages，一样的，只占用 kernel 几百字节的空间。

**注意：** `ipcs` 命令可以显示出任何使用到上述三个 System V 功能的 processes。

```
options    P1003_1B          #Posix P1003_1B real-time extensions
```

```
options    _KPOSIX_PRIORITY_SCHEDULING
```

在 1993 POSIX 中添加的实时扩展。在 ports collection 中某些应用程序会用到这些(如 Star Office)。

```
options    ICMP_BANDLIM      #Rate limit bad replies
```

这个选项启用 ICMP 的带宽限制的错误响应。你使用这个选项可以帮助你保护你的机器免受拒绝式服务攻击。

```
# To make an SMP kernel, the next two are needed
```

```
#options    SMP                # Symmetric MultiProcessor Kernel
```

```
#options    APIC_IO            # Symmetric (APIC) I/O
```

上面两个选项都支持 SMP。

```
device     isa
```

所有 FreeBSD 支持的 PC 都需要这行设置。如果你使用 IBM PS/2 (微信道架构)计算机, 则你无法在该机器上执行 FreeBSD。

```
device     eisa
```

如果你的主机板上有 EISA 总线, 加入这个设置。使用这个选项可以自动扫描并设置所有连接在 EISA 总线上的设备。

```
device     pci
```

如果你的主板有 PCI 总线, 就加入这个选项。使用这个选项可以自动扫描 PCI 卡, 并在 PCI 到 ISA 之间建立通路。

```
# Floppy drives
```

```
device     fdc0                at isa? port IO_FD1 irq 6 drq 2
```

```
device     fd0                 at fdc0 drive 0
```

```
device     fd1                 at fdc0 drive 1
```

软盘控制器: fd0 是 A: 盘, fd1 是 B: 盘。ft0 则是连接到软盘的 QIC-80 磁带机。如果你没有上述设备, 就注释掉这几行设置。

```
device     ata
```

这个驱动器支持所有 ATA 和 ATAPI 设备。你只要在内核中加入 `ata` 选项，就可以让内核支持现代计算机上的所有 PCI ATA/ATAPI 设备。

```
device      atadisk          # ATA disk drives
```

这个是 ATAPI 磁盘驱动器所必须的。

```
device      atapicd          # ATAPI CDROM drives
```

这个是 ATAPI CDROM 驱动器所必须的。

```
device      atapifd          # ATAPI floppy drives
```

这个是 ATAPI 软盘驱动器所必须的。

```
device      atapist          # ATAPI tape drives
```

这个是 ATAPI 磁带机驱动器所必须的。

```
options     ATA_STATIC_ID      #Static device numbering
```

这个可以静态分配控制器的编号，也可以动态分配设备的编号。

```
# ATA and ATAPI devices
```

```
device      ata0          at isa? port IO_WD1 irq 14
```

```
device      ata1          at isa? port IO_WD2 irq 15
```

上面的选项用在比较老的，非 PCI 的系统中。

```
# SCSI Controllers
```

```
device      ahb          # EISA AHA1742 family
```

```
device      ahc          # AHA2940 and onboard AIC7xxx devices
```

```
device      amd          # AMD 53C974 (Teckram DC-390(T))
```

```
device      dpt          # DPT Smartcache - See LINT for options!
```

```
device      isp          # QLogic family
```

```
device      ncr          # NCR/Symbios Logic
```

```
device      sym          # NCR/Symbios Logic (newer chipsets)
```

```
device      adv0         at isa?
```

```
device      adw
```

```
device      bt0          at isa?
```

```
device      aha0         at isa?
```

```
device      aic0      at isa?
```

SCSI 控制器。可以注释掉你系统中没有的设备。如果你只有 IDE 设备，你可以把这些一起删掉。

```
# SCSI peripherals
```

```
device      scbus     # SCSI bus (required)
```

```
device      da       # Direct Access (disks)
```

```
device      sa       # Sequential Access (tape etc)
```

```
device      cd       # CD
```

```
device      pass     # Passthrough device (direct SCSI  
access)
```

SCSI 外围设备。也可以象上面一样操作。

```
# RAID controllers
```

```
device      ida      # Compaq Smart RAID
```

```
device      amr      # AMI MegaRAID
```

```
device      mlx      # Myl ex DAC960 family
```

支持 RAID 控制器。如果你没有这些，可以把它们注释掉或是删掉。

```
# atkbd0 controls both the keyboard and the PS/2 mouse
```

```
device      atkbd0   at isa? port IO_KBD
```

键盘控制器 atkbd 提供 AT 键盘输入以及 PS/2 指针设备的 I/O 服务。键盘驱动程序 atkbd 与 PS/2 鼠标驱动程序 psm 需要这个控制器，所以不要删除它。

```
device      atkbd0   at atkbd? irq 1
```

atkbd 驱动程序，与 atkbd 控制器一起作用，提供连接到 AT 键盘控制器的 AT 84 键盘与 AT 加强型键盘的访问服务。

```
device      psm0     at atkbd? irq 12
```

如果你的鼠标连接到 PS/2 鼠标端口，就使用这个设备驱动程序。

```
device      vga0     at isa?
```

显卡驱动。

```
# splash screen/screen saver
```

```
pseudo-device      splash
```

在启动时的启动画面！屏幕保护也需要这个。

```
# syscons is the default console driver, resembling an SCO console
device      sc0      at isa?
```

sc0 是默认的 console 驱动程序，绝大部分全屏幕程序都通过 termcap 这类 terminal database library 来访问 console，因此不论你用这个驱动程序或是 VT220 兼容 console 驱动程序 vt0，这中间并没有太大差别。如果使用 sc0，且你在登陆系统后，执行全屏幕程序时遇到问题，请将你的 TERM 设置成 “scoansi”。

```
# Enable this and PCVT_FREEBSD for pcvt vt220 compatible console driver
#device      vt0      at isa?
#options     XSERVER      # support for X server on a vt console
#options     FAT_CURSOR   # start with block cursor
# If you have a ThinkPAD, uncomment this along with the rest of the PCVT lines
#options     PCVT_SCANSET=2 # IBM keyboards are non-std
```

这是一个兼容 VT220 的 console 驱动程序，并向下兼容 VT100/102。在部分与 sc0 相冲突的笔记本电脑上，这个驱动程序运作良好。当然，当你登陆系统时，记得设置 TERM 参数为 vt100 或是 vt220。当连接到网络上许多计算机时，这个驱动程序也常是很有用的，这是因为许多的计算机上的 termcap 或是 termi nfo 并没有 sc0 的资料—而 vt100 的资料几乎所有的平台上都支持。

```
# Power management support (see LINT for more options)
device      apm0      at nexus? disable flags 0x20 # Advanced Power
Management
```

高级电源管理支持。使用在膝上型电脑上。

```
# PCCARD (PCMCIA) support
device      card
device      pci0      at isa? irq 10 port 0x3e0 iomem 0xd0000
device      pci1      at isa? irq 11 port 0x3e2 iomem 0xd4000 disable
```

PCMCIA 支持。如果你使用膝上型电脑，你需要这个。

```
# Serial (COM) ports
device      sio0      at isa? port IO_COM1 flags 0x10 irq 4
device      sio1      at isa? port IO_COM2 irq 3
device      sio2      at isa? disable port IO_COM3 irq 5
device      sio3      at isa? disable port IO_COM4 irq 9
```

sio0 到 sio3 可看作是 MS-DOS 系统中的 COM1 到 COM4。如果你使用内置式的数据机，且占用 COM4，而你的系统又有 COM2，则你必须修改调制解调器的 IRQ 为 2 (IRQ 2 跟 IRQ 9 是一样的)，这样你才能正常使用调制解调器。部分显示卡 (特别是使用 S3 芯片的卡)，用到 0x\*2e8 这个 I/O 地址，而一些便宜的串行卡，没办法正确译码 16 位的 I/O 寻址空间，因此两张卡会冲突，导致 COM4 无法正常使用。每个串行口都要有一个唯一的 IRQ，所以 COM3 与 COM4 默认的 IRQ 将无法使用。

```
# Parallel port
device      ppc0      at isa? irq 7
```

ISA-bus 并行接口

```
device      ppbus     # Parallel port bus (required)
```

提供并行总线的支持。

```
device      lpt       # Printer
```

提供并口打印机的支持。

**注意：**要使用并口打印机，就必须同时加入上面三行设置。

```
device      plip      # TCP/IP over parallel
```

这是针对并行网络接口的驱动器。

```
device      ppi       # Parallel port interface device
```

普通用途的 I/O (“geek port”) + IEEE1284 I/O。

```
#device     vpo       # Requires scbus and da
```

这是针对 Iomega Zip 驱动器的。它要求 scbus 和 da 的支持。最好的执行效果是工作在 EPP 1.9 模式。

```
# PCI Ethernet NICs.
```

```
device      de          # DEC/Intel DC21x4x ('`Tulip'')
device      fxp         # Intel EtherExpress PRO/100B (82557, 82558)
device      tx          # SMC 9432TX (83c170 ``EPIC'')
device      vx          # 3Com 3c590, 3c595 ('`Vortex'')
device      wx          # Intel Gigabit Ethernet Card ('`Wiseman'')
```

多种 PCI 网卡驱动器。注释或删除在你系统中没有的设备。

```
# PCI Ethernet NICs that use the common MII bus controller code.
```

```
device      miibus     # MII bus support
```

MII 总线支持对于一些 PCI 10/100 Ethernet NICs 来说是必需的。

```
device      dc          # DEC/Intel 21143 and various workalikes
device      rl          # RealTek 8129/8139
device      sf          # Adaptec AIC-6915 ('`Starfire'')
device      sis         # Silicon Integrated Systems SiS 900/SiS 7016
device      ste         # Sundance ST201 (D-Link DFE-550TX)
device      tl          # Texas Instruments ThunderLAN
device      vr          # VIA Rhine, Rhine II
device      wb          # Winbond W89C840F
device      xl          # 3Com 3c90x ('`Boomerang'', ``Cyclone'')
```

使用 MII 总线控制器代码的驱动器。

```
# ISA Ethernet NICs.
```

```
device      ed0        at isa? port 0x280 irq 10 iomem 0xd8000
```

```
device      ex
```

```
device      ep
```

```
# WaveLAN/IEEE 802.11 wireless NICs. Note: the WaveLAN/IEEE really
```

```
# exists only as a PCMCIA device, so there is no ISA attachment needed
```

```
# and resources will always be dynamically assigned by the pccard code.
```

```
device      wi
```

```
# Aironet 4500/4800 802.11 wireless NICs. Note: the declaration below will
```

```
# work for PCMCIA and PCI cards, as well as ISA cards set to ISA PnP
# mode (the factory default). If you set the switches on your ISA
# card for a manually chosen I/O address and IRQ, you must specify
# those parameters here.

device      an

# The probe order of these is presently determined by i386/isa/isa_compat.c.

device      ie0    at isa? port 0x300 irq 10 iomem 0xd0000
device      fe0    at isa? port 0x300
device      le0    at isa? port 0x300 irq 5 iomem 0xd0000
device      lnc0   at isa? port 0x280 irq 10 drq 0
device      cs0    at isa? port 0x300
device      sn0    at isa? port 0x300 irq 10

# requires PCCARD (PCMCIA) support to be activated

#device     xe0    at isa?
```

ISA 以太网驱动器。看看/usr/src/sys/i386/conf/LINT 了解一下哪个卡被哪个驱动器支持。

```
pseudo-device ether      # Ethernet support
```

如果你有一个以太网卡，ether 是必需的。它包含了通用的以太网协议代码。

```
pseudo-device sl        1      # Kernel SLIP
```

SL 是针对 SLIP 支持的。这已经完全被 PPP 所代替，它是早期使用的协议。

```
pseudo-device ppp       1      # Kernel PPP
```

ppp 提供内核模式(kernel-mode)的 PPP 拨号支持。另外有一个 tun 则是用户模式(user-mode)的 PPP 支持，tun 较有弹性且功能较多。如果你要使用这个 PPP 驱动程序，请参考内核模式 PPP 这一章节。如同 sl，number 设置系统最多同时能支持几个 PPP 连接。

```
pseudo-device tun       # Packet tunnel.
```

tun 是使用者模式的 PPP 软件。这个程序设置相当方便，且速度很快。它有一些特殊的功能，像是在需要连接时自动拨号(dial-on-demand)。tun 后面接的数字设置系统同时能支持几个 PPP 连接。参看[使用者模式 PPP](#)以获得更多信息。

```
pseudo-device  pty                # Pseudo-ttys (telnet etc)
```

pty 是虚拟的终端机，或仿真的 login port。Ctelnet 或 rlogin 连接、xterm 以及其它应用程序如 emacs 会用到 pty。number 设置系统的 pty 个数。系统默认值是 16，如果你要提高同时连接数，可以适当增加这个数值，最高可达 256 个。

```
pseudo-device  md                  # Memory "disks"
```

```
Memory disk pseudo-devices.
```

```
pseudo-device  gif      4         # IPv6 and IPv4 tunneling
```

这个执行 IPv6 与 IPv4, IPv4 与 IPv6, IPv4 与 IPv4, IPv6 与 IPv6 之间的转换。

```
pseudo-device  faith  1         # IPv6-to-IPv4 relaying (translation)
```

这个伪设备能检测到接收到的数据包，然后把它们发送给 IPv4/IPv6 翻译程序。

```
# The `bpf' pseudo-device enables the Berkeley Packet Filter.
```

```
# Be aware of the administrative consequences of enabling this!
```

```
pseudo-device  bpf                # Berkeley packet filter
```

这是 Berkeley 的封包过滤器。这个伪设备允许网络接口被放在复杂的模式上，在网络上捕获每个封包。这些封包能被磁盘捕获或被 tcpdump 程序检查。

**注意：**bpf pseudo-device 也可以被 dhclient 用来获得默认路由的 IP 地址。如果你使用 DHCP，不要注释掉这行。

```
# USB support
```

```
#device        uhci                # UHCI PCI->USB interface
```

```
#device        ohci                # OHCI PCI->USB interface
```

```
#device        usb                  # USB Bus (required)
```

```
#device        ugen                 # Generic
```

```
#device        uhid                 # ``Human Interface Devices''
```

```
#device        ukbd                 # Keyboard
```

```
#device        ulpt                 # Printer
```

```
#device        umass                # Disks/Mass storage - Requires scbus and da
```

```
#device        ums                  # Mouse
```

```
# USB Ethernet, requires mi i
#device      aue          # ADMtek USB ethernet
#device      cue          # CATC USB ethernet
#device      kue          # Kawasaki LSI USB ethernet
```

支持多种 USB 设备。更多有关 FreeBSD 支持的设备请参考 `/usr/src/sys/i386/conf/LINT`。

### 9.5 建立设备的节点

几乎内核中的每个设备在 `/dev` 目录下都有对应的节点。这些节点看上去是些规则文件，但事实上是程序在使用对应的设备时，与内核联系的进入点。当你一开始安装操作系统时，那些可执行的外壳脚本 `/dev/MAKEDEV` 就创建了几乎所有支持的设备。然而，它并不是建立所有设备，所以当你加入对新设备的支持时，注意确信对应的节点在这个目录下。如果不是，就加入它们。这儿是一个简单的例子：

```
确信在内核中加入了 IDE CD-ROM 的支持。可以这样加入：device acd0
```

这意味着你应当在 `/dev` 目录下找一些以 `acd0` 为起点的入口，通常后面有一个字母，象 `c` 或以 `r` 开头表示这是一个 raw 设备。那些文件不在那儿，必须改变 `/dev` 目录然后键入：

```
# sh MAKEDEV acd0
```

当这些脚本完成后，你要找一下在 `/dev` 目录下有 `acd0c` 和 `racd0c` 等几个入口，表示程序已经正确执行。

以下是加入声卡节点的例子：

```
# sh MAKEDEV snd0
```

**注意：**当创建完诸如声卡这样的设备节点时，如果其它人有权访问你的机器，可能有必要在 `/etc/fstab` 文件中添加这些节点来保护系统安全。可看一下 `fstab` 的联机手册以了解更多信息。

依上述的简单程序，建立任何不在 `GENERIC` 里的硬件设备节点。

**注意：**所有的 SCSI 控制器都使用一样的设备节点，所以你无须重新建立节点。另外，网卡与 SLIP/PPP 虚拟设备并没有任何设备节点，所以你不必担心怎么建立节点。

### 9.6 如果出现问题怎么办？

在定制一个内核时，可能会出现四种问题。它们是：

配置失败

当你在你的内核描述中看到 `config` 命令失败，你可能在某个地方发生了一个小错误。幸运的是，`config` 会显示出错的那一行的行号，你可以用 `vi` 编辑器做修改。例如，如果你看到：`config: line 17: syntax error`

你可以在 `vi` 中用命令模式输入 `17G` 来跳到 17 行。比较 `GENERIC` 内核或其它参考资料，以确定你打对关键字。

### 编译错误

如果编译失败，通常会在你的内核描述中提示一个错误，但 `config` 并没有找出错误。另外，查看一下你的配置信息，如果你仍然无法解决这个问题，可以把你的内核配置信息发邮件到 FreeBSD 普通问题邮件列表 <freebsd-questions@FreeBSD.org>，它将很快地被解决。

### 内核无法启动

如果你的新内核无法启动，或无法识别你的硬件，不要担心！幸运的是，BSD 有一个解决复杂内核错误的很好机制。从 FreeBSD 的启动 loader 中，选择一个你要启动的内核。当配置一个内核时，记得在手头保留一个能正常启动的内核是个好主意。在用一个好的内核启动后，你可以检查一下你的配置文件，再设法重新建立一个内核。一个有帮助的文件是 `/var/log/messages` 文件，它记载了每个成功启动的所有内核信息。同样，`dmesg` 命令会列出当前启动中的内核信息。

**注意：**你在编译内核时，确信保留着一个 `GENERIC` 或以其它名字命名的内核。你不能仅依靠 `kernel.old`，因为当你安装一个新内核时，`kernel.old` 会被最后一次安装的内核所覆盖。所以，尽快将当前正常工作的内核移到不能正常工作的内核，你可以用 `ps` 命令查一下。解开编译安装内核文件的正确命令是：

```
#chflags noschg /kernel
```

另外，如果你要设定内核或其它文件的访问限制，以至它不能被移动或修改，可以用下面的命令：

```
#chflags schg /kernel
```

内核工作，但 `ps` 根本就不工作！

如果你安装了一个不同版本的内核到系统，例如，在 3.x 系统中安装了 4.x 内核，许多系统内建的命令象 `ps` 和 `vmstat` 会根本不工作。你必须重编译 `libkvm` 库以及这些程序。千万不要随意从其它系统拷贝不同版本的内核来使用。