Gentoo Linux x86手册

Sven Vermeulen *作者*
Grant Goodyear *作者*
Roy Marples *作者*
Daniel Robbins *作者*
Chris Houser *作者*
Jerry Alexandratos *作者*
Seemant Kulleen *Gentoo x86开发人员*
Tavis Ormandy *Gentoo Alpha开发人员*
Jason Huebel *Gentoo AMD64开发人员*
Guy Martin *Gentoo HPPA开发人员*
Pieter Van den Abeele *Gentoo PPC开发人员*
Joe Kallar *Gentoo SPARC开发人员*
John P. Davis *编辑*
Pierre-Henri Jondot *编辑*
Eric Stockbridge *编辑*
Rajiv Manglani *编辑*
Jungmin Seo *编辑*
Stoyan Zhekov *编辑*
Jared Hudson *编辑*
Colin Morey *编辑*
Jorge Paulo *编辑*
Carl Anderson *编辑*
Jon Portnoy *编辑*
Zack Gilburd *编辑*
Jack Morgan *编辑*
Benny Chuang *编辑*
Erwin *编辑*
Joshua Kinard *编辑*
Tobias Scherbaum *编辑*
Xavier Neys *编辑*
Joshua Saddler *编辑*
Gerald J. Normandin Jr. *审校*
Donnie Berkholz *审校*
Ken Nowack *审校*
Lars Weiler *投稿*
Hans Joanphan *译者*
吴传文 *译者*
杨珂 *译者*
陈永骥 *译者*
娄东斌 *译者*
余雷 *译者*
叶宝泰 *译者*
杨小广 *译者*
vivian.ye *译者*
王国辉 *译者*
贾震 *译者*
余鈺炜 *译者*
陈代焱 *译者*
范华 *译者*
张乐 *译者*
沈辰俊 *译者*

更新于2010年 1月 3日
本文档的原始版本最后更新于2010年 11月 7日

内容:

- 安装Gentoo
  在这部分中，你将学习如何在你的系统中安装Gentoo。
  1. 关于如何安装Gentoo Linux
     本章介绍了本手册所讲解的安装方式。
  2. 选择合适的安装方式
     你可以用许多方法安装Gentoo。本章讲解怎样用最小安装光盘安装Gentoo。
  3. 配置网络
     要下载最新的源代码，你要先设置好网络。
  4. 准备磁盘
     为了能够安装Gentoo，你必须创建所需的分区。本章讲解如何给磁盘分区以备后用。
  5. 安装Gentoo安装文件
     我们使用一个stage3文件来安装Gentoo。在这一章里我们将教你如何解压缩stage3文件和配置Portage。
  6. 安装Gentoo基本系统
     安装并配置完stage3以后，你就会有一个可用的Gentoo基本系统了。这一章将教你如何达到这一状

态。

7. 配置内核
   Linux内核是每个发行版的核心。本章节将解释如何配置您自己的内核。
8. 配置系统
   你需要编辑一些重要的配置文件。在这一章中将对这些重要的配置文件作概述，并且介绍如何配置它们。
9. 安装必要的系统工具
   在这一章中我们将帮助你选择并安装一些重要的工具。
10. 配置引导程序
    x86架构存在几种引导程序。它们中的每一种都有自己的配置方法。我们会一步步来告诉你怎样根据你的需求配置一个引导程序。
11. 结束Gentoo的安装
    您几乎已经完成了。接下来我们只需要为您的系统创建一个（或更多）用户就可以了。
12. 下一步该做什么？
    现在你已经拥有了你自己的Gentoo操作系统了，但是下一步该做什么呢？

- 使用Gentoo
  学习如何使用Gentoo：安装软件、更改变量、改变Portage行为方式等等。
  1. Portage入门
     本章节阐述了一些“简单”的步骤，这些步骤是用户要维护自己系统中的软件所必须知道的。
  2. USE标记
     USE标记是Gentoo非常重要的一部分。在本章里，你将学习设定USE标记以及了解USE标记如何影响你的系统构建。
  3. Portage特性
     学习Portage所拥有的特性，比如对分布式编译的支持，以及ccache等等。
  4. 初始化脚本
     Gentoo使用一种特殊的初始化脚本格式，有很多特色，包括由依赖关系驱动的决定和虚拟初始化脚本。本章会解释所有这些特色以及如何与这些脚本打交道。
  5. 环境变量
     使用Gentoo你可以很容易地管理系统的环境变量。本章将教你如何去做，并描述一些常用的变量。
- 使用Portage
  “使用Portage”深入全面介绍了Gentoo的软件管理工具Portage的功能。
  1. 文件和目录
     当你想要深入了解Portage的时候，你需要知道它将文件和数据放在了什么地方。
  2. 通过变量来配置
     Portage所有的设置都可以通过不同的变量来完成，你可以设置在配置文件中或者设置成一个环境变量。
  3. 使用多个软件分支
     Gentoo根据软件的稳定性和架构支持将它们划分在不同的分支中。“使用多个软件分支”告诉你这些分支是如何配置的以及如何在个别情况下使用其他分支的软件。
  4. Portage附加工具
     您可以通过Portage提供的一些附加工具为您的Gentoo旅程带来更多的快乐。通过下文的阅读你会学会如何使用dispatch-conf和其他的一些工具。
  5. 改造Portage树
     “改造Portage树”提供给你一些关于如何使用你自己的Portage树，如何只同步你想要的分类，加入自己的软件包等等的心得技巧。
- Gentoo网络配置
  一份全面的Gentoo网络指南。
  1. 新手上路
     本指南可让你的网卡在大多数通用环境下迅速设置好并运行起来
  2. 高级配置
     这里，我们将学习这些配置是如何起作用的——对于将来学习模块化网络来说，这是必不可少的知识。
  3. 模块化网络
     Gentoo为您提供了灵活的网络支持——本节将告诉你怎样选择不同的DHCP客户端程序，如何配置网络的绑定和桥接以及配置VLAN等等相关的知识。
  4. 无线网络
     配置无线网络并不是一件简单的事情。希望我们能让您的无线网络正常工作。
  5. 附加功能
     如果你喜欢冒险的感觉，你可以在网络中加上自己的功能。
  6. 网络管理
     本章是为笔记本电脑用户和频繁切换网络的用户准备的。

A. 安装Gentoo

1. 关于如何安装Gentoo Linux

1.a. 介绍

欢迎！

首先，*欢迎*使用Gentoo。你将进入一个多选择和高性能的世界。Gentoo就意味着选择。当你安装Gentoo时，你将会清楚的了解到——你能够选择怎样编译软件，如何安装Gentoo，以及使用哪种系统日志程序等等。

Gentoo是一个快速和现代化的元发行版，拥有简洁灵活的设计理念。Gentoo基于自由软件构建，它不会对用户隐瞒任何底层细节。Portage是Gentoo使用的软件包管理系统，由于它是使用Python语言编写的，因此你可以轻松地查看和修改源代码。Gentoo的软件包系统使用的是源代码（尽管也包含对预编译的软件包的支持），配置Gentoo使用的也都是标准的文本文件。换句话说就是，开放无处不在。

希望您能明白的一点是Gentoo的根本在于*选择*，这一点非常重要。我们会尽力做到不强加给用户任何东西，如果你觉得被强迫安装了任何你不喜欢的东西，请请提交一个bug报告。

安装的步骤有哪些？

Gentoo的安装可以被分成10个步骤，从第2章到第11章，每一步都会令系统进入一个新的状态：

- 第一步后，你已准备就绪，可以安装Gentoo了
- 第二步后，你已为安装Gentoo准备好了网络连接
- 第三步后，你的硬盘已初始化完毕，等待Gentoo的安装
- 第四步后，你的安装环境已经准备好了，并准备chroot到一个新的系统环境中
- 第五步后，一些核心的软件包安装完毕，这在所有的Gentoo安装中都是相同的
- 第六步后，你已经编译好了你的Linux内核
- 第七步后，你已经写好绝大多数的Gentoo系统配置文件
- 第八步后，必要的系统工具（你可以从列表中选择）已经安装完毕
- 第九步后，你选择的启动引导程序已经安装配置好了，并且你已经登录了你新的Gentoo系统中
- 第十步后，你就可以在你的Gentoo linux系统中探索了

当你面临某个选择时，我们会尽力为你解释它的利弊。我们首先会介绍一个默认的选项，这在标题中会以“默认：”标识出来。剩下的则会被标注为：“备选：”千万*不要*认为默认选项是我们推荐的，这只是我们认为绝大多数用户会采用的。

有时候你也可以做一些可选的步骤。这样的步骤会被标注为“可选：”，当然这些步骤对于安装Gentoo来说就不是必须的。然而，有些可选的步骤依赖于你之前所做出的决定。我们将会在你需要做这样决定以及可选步骤出现的时候提醒你。

我有哪些选择？

你可以通过很多不同的方法来安装Gentoo。你可以下载我们的安装光盘安装，可以从已经安装好的另外一个发行版上安装，可以从一张可引导光盘上安装（比如Knoppix），也可以从网络启动环境或从一张恢复软盘上安装等等。

这份文档讲解了如何使用Gentoo安装光盘或者在某些情况下使用网络启动来安装Gentoo。这种安装方式假设你想要使用最新的软件包。如果你想要进行无网络安装，你应该阅读Gentoo 2008.0手册，那里面包含无网络安装指南。

请注意，如果你打算用GRP（Gentoo参考平台，一套供即时使用的预编译软件包），你*一定*要依照Gentoo 2008.0手册来安装。

关于其他安装方法请参考我们的其他安装指南。我们还提供了Gentoo安装技巧和窍门，也值得一读。如果你觉得本安装指南太过详细，你还可以使用我们文档资源中的可用于你的硬件架构的快速安装手册。

你也有这些选择：你可以从头编译整个系统，或使用Gentoo预编译软件包以在很短时间内便装好一个可用的系统。当然，你也可以使用折中的方案，从已完成一半的系统开始安装。

碰到问题？

如果你在安装过程中（或是在安装文档中）碰到问题，请到bug追踪系统检查这是否是已知的bug。如果没有，请创建一个bug报告，然后我们来解决它。大家不要害怕负责解决（你的）bug的开发人员，他们通常是不会吃人的。

需要注意的是，虽然你当前阅读的文档是针对某一特定的硬件架构的，但是它还是会包含对其他平台架构的引用。这是因为Gentoo手册中的绝大部分的源代码是对所有硬件架构都适用的（目的是减少重复劳动和节省有限的开发资源）。我们会努力把这种负面影响减少到最小以避免混淆。

如果你不能确定你的问题是否是一个使用者问题（尽管你已经仔细地阅读了文档但仍然出了错）还是一个软件问题（尽管我们已经仔细地测试了安装步骤/文档但仍然出了错），我们欢迎你到irc.freenode.net的#gentoo（英文）或是#gentoo-cn（中文）提问。当然，如果你有其他方面的关于Gentoo的问题，我们也欢迎:)

如果你有关于Gentoo的问题，请到我们的Gentoo文档查看常见问题。你也可以浏览我们论坛中的FAQs。如果你还是无法找到答案，那么就去irc.freenode.net中我们的#gentoo频道寻找吧，我们中的一些狂热者会一直呆在IRC里的:-)

2. 选择合适的安装方式

2.a. 硬件需求

介绍

在开始之前，我们首先列出安装Gentoo的硬件需求。

硬件需求

	最小光盘	LiveCD
CPU	i486或更新	i686或更新
内存	64MB	256MB
硬盘空间	1.5GB(不包括交换分区)	
交换分区	至少256MB	

2.b. Gentoo安装光盘

介绍

*Gentoo安装光盘*是*可启动*的光盘，它含有一个完整的Gentoo环境，允许你从光盘启动Linux。在启动过程中，它们会自动检测你的硬件并加载相应的驱动程序。这些光盘是由Gentoo开发者维护的。

所有的安装光盘都允许你启动、设置网络、初始化硬盘分区和从因特网上开始安装Gentoo。

Gentoo最小安装光盘

这个最小安装光盘名叫install-x86-minimal-<release>.iso。大小只有57MB。你可以用它安装Gentoo，但是*只能*用于有网络的环境。

Stage3

stage3是一个含有最小Gentoo环境的压缩包，适合按照手册继续安装Gentoo。以前，Gentoo手册介绍了使用3种stage的安装方法。虽然Gentoo仍然提供stage1和stage2，但在官方的安装指南中只使用stage3。如果你对stage1和stage2感兴趣，请阅读Gentoo FAQ中的如何使用stage1或stage2安装Gentoo?

Stage3不包含在LiveCD中，你可以从任何一个Gentoo官方镜像的releases/x86/autobuilds/current-stage3/目录里下载。

2.c. 下载刻录并从安装光盘启动

下载和刻录安装光盘

你已经选择使用了一种Gentoo安装光盘。我们从下载和刻录你选择的安装光盘开始。前面我们提到过几种安装光盘，那你怎样找到它们呢?

你可以从我们任何一个镜像下载。这些安装光盘位于releases/x86/autobuilds/current-iso/目录里。

在那个目录里你可以找到一些ISO文件，这些都是你可以在CD-R上刻录的光盘镜像。

如果你想知道文件是否在下载过程中损坏，你可以检查它的MD5校验和，并和我们提供的MD5校验和（如install-x86-minimal-<release>.iso.DIGESTS）进行比较。你可以使用Linux和Unix下的md5sum工具或者Windows下的md5sum来检查它的MD5校验和。

另外一种检查下载文件的完整性的方法是使用GnuPG来验证我们提供的数字签名（这个文件以.asc结尾）。下载签名文件。并获得公钥:

代码 3.1: 获得公钥

```
$ gpg --keyserver subkeys.pgp.net --recv-keys 17072058
```

现在验证签名:

代码 3.2: 验证数字签名

```
$ gpg --verify <签名文件> <下载的iso>
```

要刻录这些下载的ISO文件，你必须选择raw方式刻录。具体方法取决于你所选择的刻录软件。这里我们将讨论cdrecord和K3B;更多的信息可以在Gentoo FAQ里找到。

- 使用cdrecord，你只需简单的输入cdrecord dev=/dev/hdc <下载的iso 文件>（用你的CD-RW设备的路径来代替/dev/hdc）。
- 使用K3B，选择Tools > Burn CD Image。然后你可以在“Image to Burn”区域定位你的ISO文件。最后点击Start。

启动安装光盘

安装光盘刻录好后，就可以用来启动了。从光驱中取出所有的光盘，重启系统并进入BIOS。根据你的BIOS的不同一般是敲击DEL键，F1键或者ESC键就可以了。在BIOS里面，更改启动顺序以使光驱的启动在硬盘启动之前。这个选项一般在“CMOS Setup”里。如果你不这样做的话，你的系统重启后将只从硬盘启动，而忽略光驱。

现在把LiveCD放入光驱中并重新启动电脑。你会看到一个启动提示符。按回车键以使用默认选项来开始启动过程，或者用自定义的启动选项来启动，具体来说是指定一个内核并加上想要的启动选项，然后再按回车键。

指定一个内核? 是的，我们在安装光盘上提供了几个内核。默认的是gentoo。其他的内核对应特定硬件的需要，带-nofb的表示的禁用了用framebuffer。

下面你可以看到一个对已有内核的简述：

内核	描述
gentoo	默认的2.6内核带有对多CPU的支持
gentoo-nofb	与gentoo相同，但是不支持framebuffer
memtest86	检测内存错误

你也可以使用内核选项。他们代表了一些你可以激活或取消的设置。

硬件选项：

acpi=on
加载对ACPI的支持，在启动光盘的同时启动acpid后台程序。这个选项只有 在你的系统需要ACPI才能正常运行的情况下才需要。启用超线程的支持 这个选项不是必需的。

acpi=off
完全关闭ACPI。这在一些比较老的系统上有用，同时也是使用高级电源管 理（APM）的必要选项。这也将关闭对你的处理器的超线程的支持。

console=X
这个选项可以设置光盘的串口访问。第一个选项是设备，在x86上通常为 ttyS0，后面可以跟其他选项并用逗号分隔。默认选项为9600,8,n,1。

dmraid=X
这个选项用于向设备映射器RAID子系统传递参数。传递的参数必须包含 在引号内。

doapm
这个选项加载高级电源管理（APM）驱动支持。这需要你同时使用acpi=off 选项。

dopcmcia
这个选项加载了对PCMCIA和Cardbus硬件的支持，也使得pcmcia卡的 cardmgr能够在光盘启动的时候运行起来。这个选项只有在从 PCMCIA/Cardbus设备启动的时候才需要。

doscsi
这个选项加载对大部分SCSI控制器的支持。在大多数USB设备启动的时候 也需要这个选项，因为USB设备使用内核中的SCSI子系统。

sda=stroke
这个选项允许你对整个硬盘进行分区，即使你的BIOS不能处理大硬盘。 此选项仅使用在使用旧BIOS的机器上。使用时把sda替换为需要这个选项 的设备。

ide=nodma
此选项强制关闭内核中的DMA，一些IDE和CDROM驱动需要这个选项。如果 你的系统在读取IDE接口的光驱的时候遇到麻烦，可以尝试一下此选项。 此选项同时也将关闭hdparm的默认设置。

noapic
此选项关闭了在一些新主板上存在的高级可编程中断控制器。它在一些旧 的硬件上会引发一些问题。

nodetect
此选项将会关闭光盘会做的所有自动检测，包括设备检测和DHCP探测。这 在调试有问题的光盘或驱动的时候有用。

nodhcp
此选项将禁止在检测到的网卡上进行DHCP探测。这对仅有静态地址的网络 很有用。

nodmraid
关闭了设备映射RAID的支持，例如用于板载IDE/SATA RAID的控制器。

nofirewire
此选项关闭了Firewire模块加载。这个选项只有在你的Firewire硬件导致 光盘启动出现问题的时候才需要。

nogpm
此选项关闭了gpm控制台鼠标支持。

nohotplug
此选项关闭启动时对热插拔（hotplug）和冷插拔（coldplug）启动脚本的加 载。这个选项在调试失败的光盘和驱动的时候有用。

nokeymap
此选项禁止了键盘布局选择。

nolapic
此选项关闭了在单处理器内核上的本地APIC。

nosata
此选项关闭了对串行ATA（SATA）模块的加载。当你的系统的SATA子系统有 问题的时候，此选项有用。

nosmp
此选项关闭了在支持SMP（对称式多处理）的内核中的SMP功能。这个选项用 于调试在特定的驱动和主板上与SMP相关的问题。

nosound
此选项关闭了声音支持和音量的设置。这个选项在声音支持出现问题的系 统上有用。

nousb
此选项关闭了USB模块的自动加载。这个选项在调试USB问题时有用。

slowusb
这个选项在启动过程中增加一些额外的暂停，主要是为了一些慢速USB CDROM，例如IBM BladeCenter的。

卷/设备管理：

doevms
此选项打开了对IBM的可插拔EVMS（企业卷管理系统）的支持。这个选项与 lvm同时使用并不安全。

dolvm
此选项打开对Linux逻辑卷管理（lvm2）的支持。这个选项与evms同时使用 并不安全。

其他选项：

debug
启用调试代码。这个选项会在屏幕上显示很多数据，可能会很凌乱。

docache
此选项把光盘中整个运行部分缓存到内存中，你就可以umount /mnt/cdrom 然后mount另一个CDROM。此选项要求你至少有两倍于CDROM的可用内存。

doload=X
此选项使得初始内存盘加载任何列出的及其依赖的模块。把X替换成模 块名称。多个模块可以用一个逗号分隔的列表来指定。

dosshd
启动sshd，对于无人职守安装有用

passwd=foo
将foo设置为root用户密码；若使用dosshd必须使用此选项，否则我们会 将root密码设为随机值。

noload=X
此选项使得初始内存盘跳过那些指定的可能引起问题的模块。语法同 doload。

nonfs
启动是不启动portmap/nfsmount。

nox
此选项使得一个支持X的LiveCD不自动启动X，而是启动到命令行模式。

scandelay
此选项使光盘在启动过程的特定部分暂停10秒，以允许那些初始化比较慢的 设备能够被使用。

scandelay=X
此选项允许你在启动过程的特定部分指定暂停指定的秒数，以允许那些 初始化比较慢的设备能够被使用。把X替换成暂停的秒数。

注意： 本CD将会先检查“no*”选项，然后再检查“do*”选项，所以你可以以你指定的顺序来覆盖任意选项。

现在引导你的光盘，选择一个内核（如果你不喜欢默认的gentoo内核）和启动选项。例如：我们给你演示如何以dopcmcia内核参数启动gentoo内核：

代码 3.3: 启动一个安装光盘

```
boot: gentoo dopcmcia
```

然后你看到一个启动画面和一个进度条。如果你正在把Gentoo安装到一个使用非US键盘的电脑中，你必须立刻按下Alt-F1键切换到冗长模式下，然后按提示信息来做。如果10秒钟内没有选择，系统会以默认的键盘布局（US键盘）继续启动过程。一旦启动完成，你将会自动以“root”超级用户身份登录这个“Live” Gentoo Linux。在当前终端里你将会有个root提示符（#），你可以通过按Alt-F2、Alt-F3和Alt-F4切换到其他的终端，按Alt-F1回到你开始的那个终端。

现在从额外硬件配置继续

额外硬件配置

当安装光盘启动时，它会尝试检测所有的硬件设备和加载准确的内核模块来支持这些硬件。绝大多数情况下，它会做的很好。但是，有些情况下，它可能没有自动加载你需要的内核模块。如果PCI自动检测错过了你系统里的一些硬件，你需要手动为它们加载相应的内核模块。

在下一个例子中，我们尝试加载8139too模块（支持某种网卡）：

代码 3.4: 加载内核模块

```
# modprobe 8139too
```

可选：用户帐号

如果你打算为其他人提供你安装环境的访问权限，或者你不想以root帐号的身份（因为安全原因）使用irssi聊天的话，你需要创建必要的用户帐号，并改变root用户的密码。

使用passwd命令来修改root用户密码：

代码 3.5: 修改root用户密码

```
# passwd
New password: （输入新密码）
Re-enter password: （再次输入密码）
```

要创建用户帐号，首先输入他们的信息，再输入他的密码。我们可以使用useradd和passwd来完成。在下个的例子中，我们创建一个名为“john”的用户。

代码 3.6: 创建一个新用户

```
# useradd -m -G users john
# passwd john
New password: （输入john的密码）
Re-enter password: （再次输入john的密码）
```

你可以使用su从root用户切换到新建的用户：

代码 3.7: 切换用户

```
# su - john
```

可选：在安装时查看文档

如果你想在安装的时候查看Gentoo手册（不管是光盘中的还是在线的），请确认你已经创建好了一个用户帐号（请查看可选：用户帐号）。然后按Alt-F2打开一个新的终端并登录进入。

如果你想查看光盘里的文档，你可以立即运行links来阅读：

代码 3.8: 查看光盘里的文档

```
# links /mnt/cdrom/docs/html/index.html
```

不过，首选的还是在线查看Gentoo手册，因为它比光盘里提供的要新。你也可以使用links阅读它，但是必须在完成*配置网络*这一章后才行（否则你不能到因特网上查看文档）：

代码 3.9: 查看在线文档

```
# links http://www.gentoo.org/doc/zh_cn/handbook/handbook-x86.xml
```

你可以按Alt-F1来回到原来的终端。

可选：启动SSH服务

如果你想在安装Gentoo的过程中允许别的用户登录你的电脑（可能是因为别的用户愿意帮助你安装Gentoo，甚至是代你安装），你需要为他们创建用户帐号，甚至提供root用户密码（*只有*在你**完全信任**这位用户的情况下才行）。

执行下面的命令来启动SSH服务：

代码 3.10: 启动SSH服务

```
# /etc/init.d/sshd start
```

在使用sshd之前，你必须设置好你的网络。请从配置网络一章继续。

3. 配置网络

3.a. 自动网络检测

能够自动检测到么？

如果你的系统接入一个拥有DHCP服务器的以太网，那么很可能网络配置已经自动设置好了。如果是这样，你就可以利用到安装盘中的许多网络命令，如ssh，scp，ping，irssi，wget和links。

如果网络已经为你配置好了，/sbin/ifconfig命令就能列出lo以外的一些网卡的信息，如eth0：

代码 1.1: 一个正常工作的网络配置的/sbin/ifconfig命令输出

```
# /sbin/ifconfig
(……)
eth0      Link encap:Ethernet  HWaddr 00:50:BA:8F:61:7A
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::50:ba8f:617a/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1498792 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1284980 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1984 txqueuelen:100
          RX bytes:485691215 (463.1 Mb)  TX bytes:123951388 (118.2 Mb)
          Interrupt:11 Base address:0xe800
```

可选：配置网络代理

如果你是通过代理访问网络，就要在安装过程中设置代理信息。设置代理很容易：只要定义一个包含代理服务器信息的变量即可。

通常情况下只要将这个变量定义为代理服务器主机名。作为例子，我们假定代理名叫proxy.gentoo.org，端口为8080。

代码 1.2: 定义代理服务器

```
(如果代理传输HTTP数据)
# export http_proxy="http://proxy.gentoo.org:8080"
(如果代理传输FTP数据)
# export ftp_proxy="ftp://proxy.gentoo.org:8080"
(如果代理传输RSYNC数据)
# export RSYNC_PROXY="proxy.gentoo.org:8080"
```

如果代理要求用户名和密码，请使用以下格式的变量：

代码 1.3: 添加用户名/密码到代理变量

http://用户名:密码@proxy.gentoo.org:8080

测试网络

ping一下你的ISP提供的DNS服务器（可于/etc/resolv.conf中找到）和一个任选的网站，以确定数据包传到了网上、DNS域名解析正常，等等。

代码 1.4: 进一步测试网络

```
# ping -c 3 www.gentoo.org
```

如果现在你的网络可用了，就可以略过本章的余下部分，进入下一章预备磁盘。如果没有，请接下去阅读。

3.b. 自动网络配置

如果网络没有马上配置好，一些安装介质允许你使用net-setup（普通或无线网络），pppoe-setup（ADSL用户）或pptp（PPTP用户——x86、amd64、alpha、ppc和ppc64平台可用）来继续配置。

如果你的安装介质没有提供这些工具或者你的网络还是不能运行，请阅读手动配置网络。

- 普通以太网用户请继续阅读默认：使用net-setup
- ADSL用户请继续阅读备选：使用PPP
- PPTP用户请继续阅读备选：使用PPTP

默认：使用net-setup

如果网络没有自动配置好，最简单的配置方法是执行net-setup脚本：

代码 2.1: 执行net-setup脚本

```
# net-setup eth0
```

net-setup会问你一些关于你的网络环境的问题。全部回答完之后，你就能拥有一个正常工作的网络连接。按照前面叙述的方法测试一下网络连接。如果测试结果正常，恭喜！现在你已准备好开始安装Gentoo了。略过本章的余下部分并进入预备磁盘。

如果网络还不能配置好，请继续阅读手动配置网络。

备选：使用PPP

假如你通过PPPoE连接internet，安装盘（所有版本）中包含的ppp工具会让事情变得简单。使用所提供的pppoe-setup脚本来配置网络。它会提示你输入一些信息，包括连接到你的adsl调制解调器的网络设备、用户名和密码、DNS服务器的IP以及是否需要一个基本的防火墙。

代码 2.2: 使用ppp

```
# pppoe-setup
# pppoe-start
```

如果配置出错了，请仔细检查/etc/ppp/pap-secrets或/etc/ppp/chap-secrets，确保你在其中输入的用户名和密码无误，并确定使用了正确的以太网设备。如果网络设备不存在，必须加载适当的网络模块。这种情况下你应该继续阅读手动配置网络，我们在其中解释了如何加载适当的网络模块。

如果一切正常，请继续阅读预备磁盘。

备选：使用PPTP

如果你需要支持PPTP连接，可以使用我们的安装光盘中所提供的pptpclient。不过你必须首先确定配置没有问题。编辑/etc/ppp/pap-secrets或/etc/ppp/chap-secrets使其中包含正确的用户名/密码对：

代码 2.3: 编辑/etc/ppp/chap-secrets

```
# nano -w /etc/ppp/chap-secrets
```

然后如果有必要，调整一下/etc/ppp/options.pptp。

代码 2.4: 编辑/etc/ppp/options.pptp

```
# nano -w /etc/ppp/options.pptp
```

当做好了所有的配置，就请执行pptp（跟随一些不能在options.pptp中设置的选项）来连接服务器：

代码 2.5: 连接到一个“拨入”服务器

```
# pptp <服务器ip>
```

现在请继续阅读预备磁盘。

3.c. 手动配置网络

加载适当的网络模块

安装光盘在启动时，会尝试检测所有硬件设备并加载适当的内核模块（驱动程序）以支持你的硬件。绝大多数情况下，它都做得非常好。尽管如此，某些情况下它还是可能无法自动载入你需要的内核模块。

如果net-setup或pppoe-setup执行失败，那么很可能是没有一下子找到你的网卡。这意味着你将不得不手动加载适当的内核模块。

要找出我们为网络提供了哪些内核模块，请使用ls：

代码 3.1: 搜索所提供的模块

```
# ls /lib/modules/`uname -r`/kernel/drivers/net
```

如果找到了你的网卡的驱动程序，请使用modprobe来加载这个内核模块：

代码 3.2: 使用modprobe加载一个内核模块

```
（作为一个例子，我们载入pcnet32模块）
# modprobe pcnet32
```

现在用ifconfig检查一下是否检测到你的网卡了。成功检测到的网卡会输出类似下面这样的信息：

代码 3.3: 测试网卡是否可用，成功

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FE:FD:00:00:00:00
          BROADCAST NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

不过如果你得到如下错误信息，说明没有检测到网卡。

代码 3.4: 测试网卡是否可用，失败

```
# ifconfig eth0
eth0: error fetching interface information: Device not found
```

如果你的系统中存在多块网卡，分别名为*eth0*、*eth1*等等。确保你想要使用的那张网卡没有问题，并在阅读本文整篇文章时都记得使用它正确的名字。我们假定使用的是*eth0*这张网卡。

假如你现在拥有一张成功检测到的网卡，可以再试一试net-setup或pppoe-setup（现在应该能正常使用了）。不过对于用户中的喜欢刨根究底的人，我们会为你解释如何通过手动配置网络。

基于你的网络设置，请从以下列表中选择相应的章节继续：

- 使用DHCP自动获取IP
- 预备无线网络接入如果你有一张无线网卡
- 理解网络术语解释了你需要了解哪些关于网络方面的知识
- 使用ifconfig和route解释了如何手动设置你的网络

使用DHCP

DHCP（动态主机配置协议）可以让你自动获得网络信息（IP地址、子网掩码、广播地址、网关以及域名服务器等）。只有在你的网络中拥有一台DHCP服务器时才能获得（或者你的网络服务提供商提供DHCP服务）。执行dhcpcd使网络接口自动获得这些信息：

代码 3.5: 使用dhcpcd

```
# dhcpcd eth0
一些网络管理员要求你使用DHCP服务器所提供的主机名和域名。
这种情况下请用
# dhcpcd -HD eth0
```

如果工作正常（尝试ping一些互联网上的服务器，比如Google），那么你就完全准备好继续下一步了。略过本章的余下部分并继续阅读预备磁盘。

预备无线网络接入

注意： iwconfig命令仅在x86、amd64和ppc安装光盘中提供。不过在其他情况下，你仍可以按照linux-wlan-ng项目的介绍，来使无线扩展运作起来。

如果你正在使用一张无线（802.11）网卡，那么在继续任何一步之前你需要先配好无线设置。要查看当前的无线网卡设置，可以使用iwconfig。执行iwconfig会显示以下类似信息：

代码 3.6: 显示当前无线设置

```
# iwconfig eth0
```

```
eth0      IEEE 802.11-DS  ESSID:"GentooNode"
          Mode:Managed  Frequency:2.442GHz  Access Point: 00:09:5B:11:CC:F2
          Bit Rate:11Mb/s   Tx-Power=20 dBm   Sensitivity=0/65535
          Retry limit:16   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:25/10  Signal level:-51 dBm  Noise level:-102 dBm
          Rx invalid nwid:5901 Rx invalid crypt:0 Rx invalid frag:0 Tx
          excessive retries:237 Invalid misc:350282 Missed beacon:84
```

注意： 一些无线网卡的设备名可能为wlan0或ra0，而不是eth0。执行不带任何命令行参数的iwconfig以确定正确的设备名。

对大多数用户来说，可能只有两处设置需要重点修改，即ESSID（亦称无线网络名）或WEP密钥。如果前面列出的ESSID和AP地址正是你的无线AP的ESSID和地址，并且你不使用WEP，那么你的无线网络已工作正常。如果你需要修改你的ESSID或添加一个WEP密钥，可以参照下列命令：

注意： 如果你的无线网络使用的是WPA或者WPA2，你将需要使用wpa_supplicant。欲获取更多有关在Gentoo Linux里配置无线网络的信息，请阅读Gentoo手册中的无线网络一章。

代码 3.7: 修改ESSID和/或添加WEP密钥

```
（这里设置网络名为“GentooNode”）
# iwconfig eth0 essid GentooNode

（这里设置一个十六进制WEP密钥）
# iwconfig eth0 key 1234123412341234abcd

（这里设置一个ASCII密钥——以"s:"作为前缀）
# iwconfig eth0 key s:某密钥
```

然后可以使用iwconfig再次确认你的无线设置。一旦你配好无线设置，就可以继续配置下一节（理解网络术语）所述的IP级别的网络选项，或者可以使用前面所述的net-setup工具。

理解网络术语

注意： 如果你知道IP地址、广播地址、子网掩码和域名服务器，那你就可以略过本节并继续阅读使用ifconfig和route。

如果以上所做的全部失败，你将不得不手动配置你的网络。这其实一点也不难。不过，你需要熟悉一些网络术语，才能配置好网络令自己满意。读完本节之后，你将了解到什么是*网关*，*子网掩码*是作什么用的，*广播地址*是如何形成的，以及为什么需要*域名服务器*。

在一个网络中，所有主机都通过IP地址（互联网协议地址）来识别。一个这种地址是由四个0到255之间的整数组合成的。嗯，至少我们是这样认识它的。实际上，一个IP地址由32个比特（0和1）组成。让我们看一个例子：

代码 3.8: IP地址的一个例子

```
IP地址（整数）：    192.168.0.2
IP地址（位）：      11000000 10101000 00000000 00000010
                    -------- -------- -------- --------
                       192      168       0        2
```

在所有可访问到的网络里，这样的IP地址跟主机是一一对应的（比如你能够连接到的每台主机必须拥有一个唯一的IP地址）。为了区别一个网络内部和外部的主机，IP地址被分为两个部分：*网络*部分和*主机*部分。

*子网掩码*用来描述这种区分，它由一系列的1跟随着一系列的0组成。IP中映射在1上的部分为网络部分，另一部分为主机部分。照例，子网掩码可以描述为IP地址的形式。

代码 3.9: 区分网络/主机的例子

```
IP地址：          192       168       0          2
                11000000 10101000 00000000 00000010
子网掩码：      11111111 11111111 11111111 00000000
                  255       255       255        0
               +--------------------------+--------+
                        网络                  主机
```

也就是说，192.168.0.14仍是我们的例子网络中的成员，而192.168.1.2就不是了。

*广播*地址是一个跟你的网络在同一个网段的IP地址，只是它的主机部分全部为1。你的网络中所有的主机都监听这个IP地址。它真正的用途是用来广播数据包。

代码 3.10: 广播地址

```
IP地址：        192       168       0          2
              11000000 10101000 00000000 00000010
广播：        11000000 10101000 00000000 11111111
                 192       168       0        255
              +--------------------------+--------+
                        网络                  主机
```

为了能在互联网上冲浪，你必须知道是哪台主机在共享Internet连接。这台主机称为*网关*。由于它是一台普通主

机，所以它有一个普通的IP地址（比如192.168.0.1）。

前面我们陈述了每台主机拥有它自己的IP地址。为了能够通过域名（而不是IP地址）连到这台主机，你需要一种服务，以将一个域名（如*dev.gentoo.org*）翻译成一个IP地址（如*64.5.62.82*）。这种服务称为域名服务。你必须在/etc/resolv.conf中定义必要的*域名服务器*以享用这种服务。

某些情况下，你的网关也作为域名服务器提供服务。否则就必须加入ISP提供的域名服务器。

总结一下，继续下一步之前你需要了解下列知识：

网络名词	范例
你的IP地址	**192.168.0.2**
子网掩码	**255.255.255.0**
广播	**192.168.0.255**
网关	**192.168.0.1**
域名服务器	**195.130.130.5, 195.130.130.133**

使用ifconfig和route

设置网络包括三个步骤。首先我们使用ifconfig为自己设置一个IP地址。然后使用route设置到网关的路由。最后我们通过将域名服务器的IP写入/etc/resolv.conf来结束整个过程。

为了设置一个IP地址，你需要先获得你的IP地址、广播地址和子网掩码。然后执行以下命令，用你的IP地址、广播地址和子网掩码分别替代${IP地址}、${广播}和${子网掩码}。

代码 3.11: 使用ifconfig

```
# ifconfig eth0 ${IP地址} broadcast ${广播} netmask ${子网掩码} up
```

现在使用route设置路由。用你的网关IP地址代替${网关}：

代码 3.12: 使用route

```
# route add default gw ${网关}
```

现在用你喜欢的编辑器（我们的例子中使用nano）打开/etc/resolv.conf：

代码 3.13: 创建/etc/resolv.conf

```
# nano -w /etc/resolv.conf
```

现在使用下面的模板填写你的域名服务器。注意用适当的域名服务器地址替换${域名服务器1}和${域名服务器2}。

代码 3.14: /etc/resolv.conf模板

```
nameserver ${域名服务器1}
nameserver ${域名服务器2}
```

这就是全部了。现在通过ping一些互联网上的服务器（比如Google）来测试一下你的网络。如果一切正常，那就恭喜了。你现在已准备好开始安装Gentoo了。接下来请继续阅读预备磁盘。

4. 准备磁盘

4.a. 块设备介绍

块设备

我们要好好了解下Gentoo Linux以及普通Linux中有关磁盘方面的知识，包括Linux文件系统、分区和块设备。然后，一旦你熟悉了磁盘和文件的方方面面，我们将会指导你设置分区和文件系统，为你安装Gentoo Linux做好准备。

一开始我们先介绍*块设备*。最有名的块设备可能就是Linux系统中表示第一个IDE硬盘的/dev/sda。SCSI硬盘和Serial ATA硬盘都是/dev/sda。如果你正在使用内核里新的libata架构，即便IDE硬盘也会是/dev/sd*。如果你用的是旧的设备架构，你的第一个IDE硬盘将是/dev/hda。

上面介绍的块设备代表磁盘的抽象接口。用户程序可以使用这些块设备来与你的磁盘进行交互，而不用理会你的驱动器到底是IDE、SCSI还是其他的。程序可以把磁盘当作一系列连续的、可随机访问的512字节大小的块来访问。

分区

尽管在理论上可以使用一整块磁盘来安装你的Linux系统，但是在实际中几乎从不这样做。相反，整个磁盘块设备被分割成更小、更容易管理的块设备。在x86系统中，这些被称作*分区*。

分区有三种类型：*主分区*，*扩展分区*和*逻辑分区*。

*主分区*是把自己的信息储存在MBR（主引导记录）中的分区。由于MBR非常小（512个字节），所以仅可以定义4个主分区（例如，/dev/sda1到/dev/sda4）。

扩展分区是一种特殊的主分区（意味着扩展分区必须是4个可能的主分区之一），它包含着更多的分区。这种分区最初并不存在，但是由于4个主分区太少了，为了能划分更多的分区，在保持向后的兼容性的前提下扩展分区诞生了。

*逻辑分区*是在扩展分区内部的分区。它们的定义不在MBR中，而是在扩展分区中。

高级存储

此x86安装光盘提供了对EVMS和LVM2的支持。EVMS和LVM2提高了你的分区设置的灵活性。在安装指南中，我们将把精力集中在“常规”分区上，但是了解一下我们支持EVMS和LVM2也是有好处的。

4.b. 设计分区方案

默认分区方案

如果你没有兴趣为你的系统设计分区方案，你可以使用我们在本手册中使用的方案：

分区	文件系统	大小	说明
/dev/sda1	ext2	32M	启动分区
/dev/sda2	(swap)	512M	交换分区
/dev/sda3	ext3	剩余磁盘	根分区

如果你想知道一个分区应该多大，或者你需要多少个分区，继续阅读。否则请阅读使用fdisk来为你的磁盘分区来给你的磁盘分区。

多少和多大？

分区的数目主要根据你的系统环境来决定。例如，如果你有很多用户，你可能更希望把你的/home目录独立出来，因为这样做可以增加安全性，备份起来也更容易。如果你安装Gentoo用来做邮件服务器，你的/var目录应该独立出来，因为邮件都存储在/var目录中。选择一个好的文件系统将最大限度地提高你的系统性能。游戏服务器应该把/opt目录独立出来，因为大多数游戏服务端软件都安装在那。理由也与/home目录类似：安全与备份。你一定要使/usr 目录保持足够大：因为它将不仅包含大部分应用程序，而且Portage树本身就需要大约500M空间，这还不包括存储在其中的各种源代码包。

正如你所看到的，这完全取决于你想要达到什么目的。独立的分区或卷有以下优点：

- 你可以为每一个分区或卷选择最佳性能的文件系统
- 如果一个有问题的工具不停地向一个分区或卷写文件，也不会把你整个系统的可用空间都用完
- 如果需要，可以减少文件系统检测的时间，因为多个检测可以并行的进行（尽管这个优势在多磁盘系统上比多分区系统上更为明显）
- 通过对一些分区的进行只读、nosuid（忽略setuid位）、noexec（忽略可执行位）等方式挂载，安全性会大大增强

然而，多分区系统有一大缺点：如果配置的不合理，可能导致系统中一个分区有很大的空闲空间，而另一个没有空闲空间了。SCSI和SATA还有15个分区的限制。

作为分区的例子，我们向你示范一个用于演示的有20GB磁盘的笔记本（包含网站服务器、邮件服务器、gnome……）：

代码 2.1: 文件系统使用实例

```
$ df -h
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/sda5     ext3    509M  132M  351M  28% /
/dev/sda2     ext3    5.0G  3.0G  1.8G  63% /home
/dev/sda7     ext3    7.9G  6.2G  1.3G  83% /usr
/dev/sda8     ext3   1011M  483M  477M  51% /opt
/dev/sda9     ext3    2.0G  607M  1.3G  32% /var
/dev/sda1     ext2     51M   17M   31M  36% /boot
/dev/sda6     swap    516M   12M  504M   2% <not mounted>
（留作它用的未分配空间：2GB）
```

这里的/usr快满了（使用了83%），但是一旦所有的软件都安装好了，/usr目录就不会如此的增长了。尽管分配给/var目录的空间看似过多了，但是要记住，Portage默认使用这个分区来编译软件包。如果你想使你的/var目录保持一个合理的大小，如1GB，你需要更改/etc/make.conf 文件中的PORTAGE_TMPDIR来指定一个拥有足够的空闲空间的分区，用以编译诸如OpenOffice这样巨大的软件包。

4.c. 使用fdisk来为你的磁盘分区

下面来解释如何创建前面说明的那个示例分区布局，即：

分区	说明
/dev/sda1	启动分区
/dev/sda2	交换分区
/dev/sda3	根分区

根据你自己的喜好来改变分区布局。

查看当前分区布局

fdisk是一个流行且强大的工具，用来把你的磁盘划分为分区。启动你磁盘上fdisk程序（在示例中，我们使用/dev/sda）：

代码 3.1: 启动fdisk

```
# fdisk /dev/sda
```

一旦启动到fdisk中，你将看到如下提示：

代码 3.2: fdisk提示

```
Command (m for help):
```

键入p来显示你的磁盘的当前分区配置：

代码 3.3: 分区配置示例

```
Command (m for help): p

Disk /dev/sda: 240 heads, 63 sectors, 2184 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1   *         1        14    105808+  83  Linux
/dev/sda2            15        49    264600   82  Linux swap
/dev/sda3            50        70    158760   83  Linux
/dev/sda4            71      2184  15981840    5  Extended
/dev/sda5            71       209   1050808+  83  Linux
/dev/sda6           210       348   1050808+  83  Linux
/dev/sda7           349       626   2101648+  83  Linux
/dev/sda8           627       904   2101648+  83  Linux
/dev/sda9           905      2184   9676768+  83  Linux

Command (m for help):
```

这个磁盘配置包含了7个Linux文件系统（每个对应于列表中名为“Linux”的分区）及一个交换分区（列表中的“Linux swap”）。

删除所有分区

首先，我们将把磁盘上的所有分区删除。键入d来删除一个分区。例如，要删除存在的/dev/sda1：

代码 3.4: 删除分区

```
Command (m for help): d
Partition number (1-4): 1
```

该分区就会被列入删除计划。如果你键入p，它也不会再出现了，然而除非你保存了所做的修改，否则它并没有被真正删除。如果你犯了一个错误，想不保存修改并退出，立刻键入q并回车，你分区就不会被删除。

现在，假设你真的想删除你系统中的所有分区，键入p来显示分区列表，然后键入d和要删除的分区的数字，并重复此过程。最后，当分区表中什么也没有的时候你就可以结束了：

代码 3.5: 空的分区表

```
Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System

Command (m for help):
```

现在，内存中的分区表是空的，我们就可以创建分区了。我们将使用前面讨论过的那个默认的分区布局。当然，如果你不想要相同的分区方案就不要按照下面的指令来做！

创建启动分区

首先，我们创建一个小一点的启动分区。键入n创建一个新分区，然后键入p来选择一个主分区，接下来键入1选择第一个主分区。当提示输入第一个柱面的时候，敲回车键。当提示输入最后一个柱面的时候，输入+32M，来创建一个32M大小的分区，并设置它的启动标记：

代码 3.6: 创建启动分区

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
```

```
First cylinder (1-3876, default 1): （按回车）
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-3876, default 3876): +32M
```

现在，当你键入p时，你应该看到如下的分区输出：

代码 3.7: 已创建的启动分区

```
Command (m for help): p

Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1            1          14      105808+  83  Linux
```

我们需要把这个分区设置成可启动的。键入a来给分区添加启动标志，然后键入1。如果你再次按p键，你就会注意到，在“Boot”那一列有个*。

创建交换分区

我们现在来创建交换分区。键入n创建一个新分区，然后键入p来告诉fdisk你创建的是主分区。接着输入2来创建第2个主分区，在本例中是/dev/sda2。当提示输入第一个柱面的时候，直接敲回车。当提示输入最后一个柱面的时候，输入+512M来创建一个512MB大小的分区。在这之后，键入t来设置分区类型，键入2选择你刚刚创建的那个分区，然后再输入82把分区类型设置成“Linux Swap”。完成以上这些步骤之后，键入p，应该显示一个与下面类似的分区表：

代码 3.8: 创建交换分区后的分区列表

```
Command (m for help): p

Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *          1          14      105808+  83  Linux
/dev/sda2           15          81      506520   82  Linux swap
```

创建根分区

最后，我们来创建根分区。键入n创建一个新分区，然后键入p来告诉fdisk你创建的是主分区。接着输入3来创建第3个主分区，在本例中是/dev/sda3。当提示输入第一个柱面的时候，直接敲回车。当提示输入最后一个柱面的时候，单击回车把你磁盘上的剩余空间创建一个分区。完成以上这些步骤之后，键入p，应该显示一个与下面类似的分区表：

代码 3.9: 创建根分区后的分区列表

```
Command (m for help): p

Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *          1          14      105808+  83  Linux
/dev/sda2           15          81      506520   82  Linux swap
/dev/sda3           82        3876    28690200   83  Linux
```

保存分区布局

键入w来保存分区布局并退出fdisk。

代码 3.10: 保存并退出fdisk

```
Command (m for help): w
```

现在分区已经创建完毕，你可以继续创建文件系统。

4.d. 创建文件系统

介绍

你的分区已经创建完了，现在可以在上面安装文件系统了。如果你并不介意选择何种文件系统，而且乐意使用本手册中所使用的默认设置，请阅读在分区上应用文件系统。否则，继续阅读本文来了解可用的文件系统……

文件系统

Linux内核支持各种各样的文件系统。我们将介绍ext2、ext3、ReiserFS、XFS和JFS，因为它们是Linux系统中使

用最普遍的文件系统。

ext2是经考验证明可靠的Linux文件系统，但是没有元数据日志，这意味这在启动系统时的ext2文件系统的日常检查相当耗时。现在相当一部分的新一代的日志文件系统都可以非常迅速检查一致性，因此比那些非日志文件系统更受欢迎。当你启动系统碰巧遇到文件系统状态不一致时，日志文件系统不会在那里耽搁很长时间。如果你要在一个很小的硬盘（小于4G）上装Gentoo，那么你需要在创建ext2文件系统时预留足够的inode，执行此命令`mke2fs -T small /dev/<device>`.

ext3是ext2文件系统的带日志版本，提供了元数据日志模式以快速恢复数据。此外还提供了其他增强的日志模式，如完整数据日志模式和有序数据日志模式。它使用了HTree索引，在几乎所有的情况下都能保持高性能。简而言之，ext3是非常好及可靠的文件系统。如果你要在一个很小的硬盘（小于4G）上装Gentoo，那么你需要在创建ext2文件系统时预留足够的inode，执行此命令`mke2fs -T small /dev/<device>`。

JFS是IBM的高性能日志文件系统。JFS是一个轻量级的、快速的和稳定的基于B+树的文件系统，在很多情况下都有很好的表现。

ReiserFS是基于B+树的文件系统，它有着非常全面的性能，特别时在处理很多小文件的时候，虽然会占用多一点CPU。ReiserFS相比其他文件系统显得受维护的不够。

XFS是一种带元数据日志的文件系统，它有一个健壮的特性集，并且对可伸缩性进行了优化。XFS似乎对各种各样的硬件问题显得不够宽容。

在分区上应用文件系统

要在分区或卷上创建文件系统，对于每种可能的文件系统都有专门的工具。

文件系统	创建命令
ext2	mke2fs
ext3	mke2fs -j
reiserfs	mkreiserfs
xfs	mkfs.xfs
jfs	mkfs.jfs

例如，要使启动分区（本例中的/dev/sda1）为ext2和根分区（本例中的/dev/sda3）为ext3,应该这样做：

代码 4.1: 在分区上应用文件系统

```
# mke2fs /dev/sda1
# mke2fs -j /dev/sda3
```

现在在你新建的分区（或逻辑卷）上创建文件系统。

激活交换分区

mkswap是初始化交换分区的命令：

代码 4.2: 创建交换分区标志

```
# mkswap /dev/sda2
```

使用swapon命令来激活交换分区：

代码 4.3: 激活交换分区

```
# swapon /dev/sda2
```

使用上面提到的命令来创建和激活交换分区。

4.e. 挂载

现在你的分区都已经初始化了，并且安装了文件系统，是时候来挂载这些分区了。使用mount命令进行挂载。别忘记为每个你创建的分区建立所需的挂载目录。作为例子，我们来挂载根分区和启动分区：

代码 5.1: 挂载分区

```
# mount /dev/sda3 /mnt/gentoo
# mkdir /mnt/gentoo/boot
# mount /dev/sda1 /mnt/gentoo/boot
```

注意： 如果你希望/tmp目录在一个独立的分区上，确保在挂载之后修改它的权限：`chmod 1777 /mnt/gentoo/tmp`。这同样适用于/var/tmp目录。

我们还需要在/proc目录上挂载proc文件系统（内核的虚拟接口）。但是，我们首先需要把我们的文件放到分区上。

请继续阅读安装Gentoo安装文件。

5. 安装Gentoo安装文件

5.a. 安装一个Stage Tarball

正确设置日期 / 时间

在继续之前，你需要检查和更新系统日期 / 时间。未正确设置的时钟可能会在将来导致奇怪的结果！

确认当前日期 / 时间，请运行date：

代码 1.1: 确认日期 / 时间

```
# date
Fri Mar 29 16:21:18 UTC 2005
```

如果显示的日期 / 时间不正确，可以使用date MMDDhhmmYYYY命令（**MM**是月，**DD**是日，**hh**是时，**mm**是分，**YYYY**是年）来更新它。在这一步，你应该使用UTC时间。稍后你可以设置你自己的时区。举个例子，设置时间为2005年3月29日16时21分：

代码 1.2: 设置UTC日期 / 时间

```
# date 032916212005
```

做出你的选择

接下来，你要把*stage3* tarball安装到系统。你可以选择从网上下载它，或者如果你是使用Gentoo通用安装光盘或者LiveDVD引导系统的话，也可以从光盘里拷贝一个出来。如果你有通用安装光盘或者LiveDVD且光盘中有你需要的Stage文件的话，就直接用光盘里的吧，否则再从网上下载就是浪费带宽，因为它和网上下载的是一样的。大多数情况下，运行命令uname -m可以告诉你哪个stage文件才是你要下载的。

最小光盘和LiveCD不包含任何stage文件，不过LiveDVD里有。

5.b. 默认：使用从网上下载的Stage

下载Stage Tarball

进入Gentoo的挂载点，也就是装Gentoo的分区挂载的地方（很可能是/mnt/gentoo）：

代码 2.1: 进入Gentoo的挂载点

```
# cd /mnt/gentoo
```

根据你使用的安装介质不同，你有好几个工具可用来下载stage。如果有links，你可以马上连接到Gentoo镜像列表，然后选择一个离你最近的镜像站点来下载。

假如你没有可用的links，那你应该有lynx。如果你需要通过代理上网的，那么请export http_proxy和ftp_proxy这两个变量：

代码 2.2: 为lynx设置代理信息

```
# export http_proxy="http://proxy.server.com:port"
# export ftp_proxy="http://proxy.server.com:port"
```

我们现在假设你使用links。

进入releases/x86/autobuilds/目录里。你将会看到所有适合你的计算机体系结构的stage文件（它们也可能放在各个独立的子体系名称的子目录里）。选择一个，然后按D来下载。下载完以后，再按Q退出浏览器。

大多数PC用户应该使用**stage3-i686-<release>.tar.bz2** stage3文件。所有现代PC都是i686机器。如果你用的是一部旧机器，你可以检查Wikipedia上的i686兼容处理器列表。旧的处理器，比如Pentium，K5，K6或者Via C3以及其他类似的，需要普适性的**x86** stage3。**i486**之前的处理器不获支持。

代码 2.3: 用links连接镜像列表

```
# links http://www.gentoo.org/main/en/mirrors.xml

（如果links需要使用代理：）
# links -http-proxy proxy.server.com:8080 http://www.gentoo.org/main/en/mirrors.xml
```

确保你下载的是**stage3** tarball——使用stage1或stage2进行安装已经不再被支持。

如果你想校验所下载的stage tarball的完整性，使用md5sum然后把输出同镜像站上提供的MD5校验和对比。

代码 2.4: 校验一个stage tarball的完整性

```
# md5sum -c stage3-i686-<release>.tar.bz2.DIGESTS
stage3-i686-<release>.tar.bz2: OK
```

解开Stage Tarball

现在把你所下载的stage解压缩到系统里。我们使用tar命令来做，这是最简单的方法：

代码 2.5: 解开stage

```
# tar xvjpf stage3-*.tar.bz2
```

确保你使用了同样的参数（xvjpf）。x表示解开（*Extract*），v表示详细信息（*Verbose*）可以用来查看解压缩时发生了什么（可选参数），j表示使用bzip2解压缩，p表示保留权限（*Preserve permissions*），还有f表示我们要解开一个文件，而不是标准输入。

现在stage已经安装好，下面我们继续安装Portage。

5.c. 安装Portage

解开一个Portage快照

现在你得安装一个Portage的快照，它包含的一堆文件告诉Portage哪些软件可以安装，有哪些profile可用等等。

从网上下载和安装Portage快照

进入Gentoo文件系统的挂载点（很可能是/mnt/gentoo）：

代码 3.1: 进入Gentoo的挂载点

```
# cd /mnt/gentoo
```

打开links（或者lynx）然后到我们的Gentoo镜像列表。选择一个离你最近的镜像，打开snapshots/目录。然后选择最新的Portage快照（portage-latest.tar.bz2）并按D来下载它。

代码 3.2: 浏览Gentoo镜像列表

```
# links http://www.gentoo.org/main/en/mirrors.xml
```

现在按Q来退出浏览器。你现在已经有一个Portage快照保存在/mnt/gentoo里了。

如果你想校验所下载的快照的完整性，使用md5sum然后把输出和镜像站提供的MD5校验和比较。

代码 3.3: 校验Portage快照的完整性

```
# md5sum -c portage-latest.tar.bz2.md5sum
portage-latest.tar.bz2: OK
```

下一步，我们要把Portage快照解压缩到你的系统里。确保你使用的解压缩命令里最后一个参数是大写的C，而不是c。

代码 3.4: 解开Portage快照

```
# tar xvjf /mnt/gentoo/portage-latest.tar.bz2 -C /mnt/gentoo/usr
```

5.d. 配置编译选项

介绍

你可设置一些影响Portage行为的变量来优化Gentoo。这些变量都可作为环境变量来设置（使用export），但是它们不是永久的。为了保持你的设置，Portage为你提供了/etc/make.conf，一个Portage的配置文件。这就是我们现在要编辑的文件。

注意： 所有可能的变量都用注释形式罗列在/mnt/gentoo/usr/share/portage/config/make.conf.example里。要成功地安装Gentoo，你只需要设置下面提到的变量。

打开你喜欢的编辑器（在这个指南里我们使用nano），这样我们可以改变我们现在和之后将讨论的优化变量。

代码 4.1: 打开/etc/make.conf

```
# nano -w /mnt/gentoo/etc/make.conf
```

你很可能已经注意到了，make.conf.example文件的结构和一般的文件一样：注释行以"#"开头，其它行使用VARIABLE="content"的语法来定义变量。make.conf文件也使用相同的语法。其中的一些变量我们接下来讨论。

CFLAGS和CXXFLAGS

CFLAGS和CXXFLAGS变量分别定义了gcc C和C++编译器的优化标记。尽管我们通常都在这里定义，你也可以对每个软件单独定义这些优化标记以获得最好的性能。因为每个程序都是不同的。

在make.conf里你应该定义一些你认为可以使系统在*一般情况*下快速响应的优化标记。不要把实验性质的设置放到这个变量里来；过多的优化会使程序表现很差（崩溃，甚至更糟，不正常工作）。

我们不会解释所有可能的优化选项。如果你想搞清楚它们，请阅读GNU在线手册或者是gcc info页面（info gcc——只能在可工作的Linux系统中查看）。make.conf.example本身也包含了不少信息和范例；不要忘了也看看它。

第一个设置是-march=或者-mtune=标记，它指定了目标架构的名字。可能的选项 会在make.conf.example里有说明（以注释形式出现）。

第二个是-O标记（是大写的O，而不是数字零），它是指定gcc的优化级别的标记。可能的级别有s（为优化文件大小），0（0——不优化），1、2乃至3是针对速度的优化标记（每个级别都包含前一级的优化措施，并额外增加了一些）。-O2是推荐的默认设置。-O3已知当全局启用时会引起一些问题，所以我们推荐你还是用-O2。

另一个普遍使用的优化标记是-pipe（不同编译阶段通信使用管道而不是临时文件）。它对产生的代码没有任何影响，但是会使用更多的内存。在内存不多的系统里，gcc可能会被杀掉。如果是那样的话，就不要用这个标记。

使用-fomit-frame-pointer（它将不在寄存器里为不需要帧指针的函数保存帧指针）可能会在调试程序的时候造成严重后果！

在你定义CFLAGS和CXXFLAGS的时候，你需要把这些优化标记都合并起来。stage3文件里包含的你解压缩出来的默认值已经足够好了。下面这个例子仅仅是个例子：

代码 4.2: 定义CFLAGS和CXXFLAGS变量

```
CFLAGS="-O2 -march=i686 -pipe"
# 两个变量使用相同的设置
CXXFLAGS="${CFLAGS}"
```

注意： 你应该看一看编译优化指南以了解更多的信息，比如不同的编译选项将如何影响你的系统。

MAKEOPTS

通过使用MAKEOPTS你可以定义在安装软件的时候同时可以产生并行编译的数目。你的CPU数目加一是个不错的选择，但是这个准则并不永远都是完美的。

代码 4.3: 单CPU系统的MAKEOPTS

```
MAKEOPTS="-j2"
```

设置已准备好，让我们开始！

根据你的喜好更新/mnt/gentoo/etc/make.conf并保存（nano用户可以敲Ctrl-X）。你现在准备好可以继续安装Gentoo基本系统了。

6. 安装Gentoo基本系统

6.a. Chroot

可选：选择境像站点

为了快速下载源代码，建议你选择一个速度快的境像站点。Portage将在make.conf中查找并使用GENTOO_MIRRORS变量定义的境像列表。你也可以浏览我们的镜像列表来寻找一个或者多个离你最近的镜像（通常它们是最快的），不过我们提供了一个不错的工具来帮助你选择境像，它叫mirrorselect。

代码 1.1: 使用mirrorselect更新GENTOO_MIRRORS变量

```
# mirrorselect -i -o >> /mnt/gentoo/etc/make.conf
```

警告： 不要选择任何IPv6的境像。我们的stage目前还不支持IPv6。

另一个重要的设置就是make.conf里的SYNC设置。这个变量包含你更新Portage树（Portage下载和安装软件时需要用到的ebuild和脚本等信息的集合）时用到的rsync服务器信息。虽然你可以手动输入一个SYNC服务器地址，不过还是让mirrorselect来帮你完成更加的方便：

代码 1.2: 使用mirrorselect选择rsync镜像站点

```
# mirrorselect -i -r -o >> /mnt/gentoo/etc/make.conf
```

运行完mirrorselect以后，最好自己再检查一下/mnt/gentoo/etc/make.conf里的设置！

拷贝DNS信息

在我们进入新环境之前，还有一件事需要完成，那就是从/etc/resolv.conf拷贝DNS信息。这一步可以保证你在进入新的系统环境后还可以继续使用网络。/etc/resolv.conf包含了你网络里的域名服务器。

代码 1.3: 拷贝DNS信息

```
（参数"-L"是必须的，用来确保我们拷贝的不是一个符号链接）
# cp -L /etc/resolv.conf /mnt/gentoo/etc/
```

挂载/proc和/dev文件系统

将/proc文件系统挂载到/mnt/gentoo/proc，这样chroot后的环境里安装时也可以获取内核提供的相关信息，然后以bind方式挂载/dev文件系统。

代码 1.4: 挂载/proc和/dev

```
# mount -t proc none /mnt/gentoo/proc
# mount -o bind /dev /mnt/gentoo/dev
```

进入新的系统环境

现在所有的分区都已经被初始化，基本环境也已安装完毕，现在是到了该用*chroot*进入新安装环境的时候了。这意味着我们从当前安装环境（安装光盘或者其他安装介质）切换到你安装的系统里（也就是被初始化的分区）。

完成chroot有三步。首先我们用chroot把根文件系统从/（安装介质里）切换到/mnt/gentoo（在你的分区里）。然后我们使用env-update来建立新的环境，也就是创建新的环境变量。最后我们用source加载这些变量。

代码 1.5: chroot到新环境里

```
# chroot /mnt/gentoo /bin/bash
# env-update
>> Regenerating /etc/ld.so.cache...
# source /etc/profile
# export PS1="(chroot) $PS1"
```

祝贺你！你现在已经在你自己的Gentoo Linux环境里了。当然这离安装完成还有段时间，因为我们可以看到安装指南还剩下很多章节呢 :-)

6.b. 配置Portage

更新Portage树

你现在应该更新你的Portage树到最新版本。执行emerge --sync。

代码 2.1: 更新Portage树

```
# emerge --sync
(如果你在使用一个慢速终端比如一些帧缓冲或者是串口的控制台，你可以添加--quiet选项来加速这个过程:)

# emerge --sync --quiet
```

如果你网络前面的防火墙的配置使得rsync请求被阻挡的话，你可以使用emerge-webrsync下载和安装一个最新的portage快照。

如果系统警告你有一个新版本的Portage可用，你可以使用emerge --oneshot portage来更新它。

选择正确的Profile

首先，我们看一下一些小的定义。

profile是每个Gentoo系统的构造块。它不仅指定了USE、CFLAGS以及其他重要变量的默认值，它还把系统可用的软件版本锁定在某个范围。而这些都是由Gentoo的开发者来维护的。

以前，这个profile很少会有用户接触到。然而，有些情况下你可以决定需不需要修改profile。

你可以用下面的命令来查看当前使用的profile：

代码 2.2: 验证系统profile

```
# eselect profile list
Available profile symlink targets:
  [1]   default/linux/x86/10.0 *
  [2]   default/linux/x86/10.0/desktop
  [3]   default/linux/x86/10.0/server
```

默认的profile会提供给你一个基于2.6版本内核的Linux系统。这是默认推荐的，但是你也可以选择另外一个profile。

系统也为某些架构提供了desktop和server的子profile。运行eselect profile list来显示所有可用的profile。

看完了你的体系结构可用的profile之后，如果你愿意，你可以换一个profile。

代码 2.3: 切换profile

```
# eselect profile set 2
```

注意： 子profile developer是专为Gentoo Linux开发任务而准备的，而*不是*用来帮助构建一般性的开发环境的。

配置USE变量

USE是Gentoo为用户提供的最具威力的变量之一。很多程序通过它可以选择编译或者不编译某些可选的支持。例如，一些程序可以在编译时加入对gtk或是对qt的支持。其它的程序可以在编译时加入或不加入对于SLL的支持。有些程序甚至可以在编译时加入对帧缓冲的支持（svgalib）以取代X11（X服务器）。

大多数的发行版会使用尽可能多的支持特性编译它们的软件包，这既增加了软件的大小也减慢了启动时间，而这些还没有算上可能会涉及到的大量依赖性问题。Gentoo可以让你自己定义软件编译的选项，而这正是USE要做的事。

在USE变量里你可以定义关键字，它被用来对应相应的编译选项。比如，*ssl*会让程序在它编译时加入对它的支

持。*-X*会移除其对于X服务器的支持（注意前面的减号）。*gnome gtk -kde -qt3 -qt4*将会在你编译软件的时候添加对gnome（和gtk）的支持，并且移除对kde（和qt）的支持，这可以让你的系统尽可能多的为GNOME做优化。

默认的USE设置位于你profile的make.defaults文件里。你可以在符号连接/etc/make.profile所指向的目录和它所有的父目录里找到make.defaults文件。默认的USE设置是所有make.defaults文件里USE的集合。所有你放在/etc/make.conf里的USE都会根据默认设置重新计算。如果你添加了一些USE的设置，它会被增加到默认的列表里。如果你删除了一些USE设置（通过放一个减号到它前面），它将被从默认的列表里移除（如果它确实在默认列表里的话）。*绝对不*要改变/etc/make.profile目录里的任何东西，它会在你更新Portage的时候被覆盖掉！

关于USE的详解你可以在Gentoo手册的第二部分，USE标记里找到。对于USE标记的详细介绍可以查看你的系统里/usr/portage/profiles/use.desc。

代码 2.4: 查看可用的USE标记

```
# less /usr/portage/profiles/use.desc
（你可用用方向键来滚动，按'q'键退出）
```

作为一个例子，我们展示一个基于KDE的系统并带有DVD、ALSA以及光盘刻录支持的USE设置：

代码 2.5: 打开/etc/make.conf

```
# nano -w /etc/make.conf
```

代码 2.6: USE设置

```
USE="-gtk -gnome qt3 qt4 kde dvd alsa cdr"
```

可选：glibc Locales

在你的系统里可能只会用到一个或两个locale。你可用/etc/locale.gen来指定locale。

代码 2.7: 打开/etc/locale.gen

```
# nano -w /etc/locale.gen
```

下面这个例子中的locale同时支持英语（美国）和德语（德国）并带有字符集格式（比如UTF-8）的支持。

代码 2.8: 指定你的locale

```
en_US ISO-8859-1
en_US.UTF-8 UTF-8
de_DE ISO-8859-1
de_DE@euro ISO-8859-15
```

下一步我们运行locale-gen。它会产生所有你在/etc/locale.gen文件里指定的locale。

现在我们继续配置内核。

7. 配置内核

7.a. 时区

您首先需要选择您自己的时区，这样可以让系统知道它的位置在哪里。您可以在/usr/share/zoneinfo中找到您所在的时区，然后把它复制到/etc/localtime。请不要使用/usr/share/zoneinfo/Etc/GMT*下的时区，因为它们“名不副实”。例如，GMT-8事实上是GMT+8区。

代码 1.1: 设置时区信息

```
# ls /usr/share/zoneinfo
（假设您要用GMT）
# cp /usr/share/zoneinfo/GMT /etc/localtime
```

7.b. 安装源码

选择内核

Linux内核是所有发行版的核心。它位于用户程序和系统硬件之间。Gentoo提供给我们几个可选的内核源码。完整的清单参见Gentoo内核指南。

对于x86架构的系统来说，我们有gentoo-sources（包含增加了额外功能的补丁）。

选择你的内核源代码并使用emerge来安装。

代码 2.1: 安装内核源码

```
# emerge gentoo-sources
```

当您查看/usr/src时，您将会看到一个叫做linux的符号链接指向您安装的内核源码。在我们这个例子中，安装的源码指向gentoo-sources-2.6.30-r5。您的版本可能有所不同，所以请你记住这一点。

代码 2.2: 查看内核源码符号链接

```
# ls -l /usr/src/linux
lrwxrwxrwx    1 root    root    12 Oct 13 11:04 /usr/src/linux -> linux-2.6.30-r5
```

现在，我们开始配置和编译您的内核。您可以用genkernel来做这件事。这将会建立一个和安装光盘所用的内核类似的通用内核。不过我们将首先说明如何手动配置一个内核，因为这是优化您系统环境的最佳方法。

如果您希望手动配置您的内核，点击默认：手动配置。如果您希望使用genkernel，您可以阅读备选：使用genkernel。

7.c. 默认：手动配置

介绍

手动配置内核经常被Linux使用者认为是最困难的步骤。事实并非如此——但是当您手动配置几次内核之后，您就不会再觉得它有多么难了。:)

然而，一件事情*是*真的：在手动配置内核之前，您必须了解您的系统。您可以安装pciutils（emerge pciutils），用其中的lspci来了解您需要的大部分信息。您现在可以在chroot的环境中运行lspci。您可以忽略任何*pcilib*的警告。（类似于pcilib: cannot open/sys/bus/pci/devices）。此外，您也可以在非chroot的环境执行lspci。结果相同。您还可以运行lsmod来查看安装光盘使用了哪些内核模块。（这也是个不错的提示，它可以教你该选择哪些模块）。

现在进入到您的内核目录并且执行make menuconfig。这将会启动一个基于ncureses的配置菜单。

代码 3.1: 开始menuconfig

```
# cd /usr/src/linux
# make menuconfig
```

您将会看到一些配置条目。首先我们将会列出一些您必须启用的选项（否则Gentoo将不能正常运行或者根本不能运行）。

必须启用的选项

确保启动您的系统所必需的驱动（比如SCSI控制器……）被编译*进*内核而不是作为模块加入的。否则您的系统将完全不能启动。

现在选择正确的处理器类型：

代码 3.2: 选择正确的处理器类别

```
Processor type and features --->
  (Change according to your system)
  (Athlon/Duron/K7) Processor family
```

现在进入File Systems并且选择您使用的文件系统。请*不要*把它们编译成模块，否则您的Gentoo系统将不能挂载您的分区。同时您也要启用Virtual memory和/proc file system的支持。

代码 3.3: 选择需要的文件系统

```
File systems --->
  Pseudo Filesystems --->
    [*] /proc file system support
    [*] Virtual memory file system support (former shm fs)

#-#-#-#-#  choice  #-#-#-#-#
（依照你系统的需要启用下面所列出的一个或是多个选项）
  <*> Reiserfs support
  <*> Ext3 journalling file system support
  <*> JFS filesystem support
  <*> Second extended fs support
  <*> XFS filesystem support
```

如果您在使用PPPoE接入Interne或者您在使用拨号的调制解调器，您需要下面的选项：

代码 3.4: 选择PPPoE驱动

```
Device Drivers --->
  Networking Support --->
    <*> PPP (point-to-point protocol) support
    <*>   PPP support for async serial ports
    <*>   PPP support for sync tty ports
```

两个压缩选项不会造成什么错误，不过它们不是必需的。PPP over Ethernet选项也不是必需的，只在使用ppp并被配置成使用核心PPPoE时才会用到它。

如果您需要它，请不要忘记在内核中包含对您的网卡的支持。

如果您拥有支持HyperThreading（tm）的Intel CPU，或者您有多个CPU，您需要 激活"Symmetric multi-processing support"：

代码 3.5: 启用SMP支持

```
Processor type and features  --->
  [*] Symmetric multi-processing support
```

注意： 在多核心系统中，处理器的数目相当于核心的数目。

如果你有多于4GB的内存，你需要启用"High Memory Support (64G)"。

如果您使用USB输入装置（比如键盘或者鼠标）那么不要忘记支持它们：

代码 3.6: 启用USB接口的输入设备支持

```
Device Drivers --->
  [*] HID Devices--->
    <*>   USB Human Interface Device (full HID) support
```

如果你想要对你的笔记本的PCMCIA支持，也不要忘了在系统中允许PCMCIA card bridge。

代码 3.7: 启用PCMCIA支持

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->
  PCCARD (PCMCIA/CardBus) support  --->
    <*> PCCard (PCMCIA/CardBus) support
（如果您需要使用老式的PCMCIA卡，选择16位。大多数人需要。）
    <*>   16-bit PCMCIA support
    [*]   32-bit CardBus support
（选择相关的bridges）
    --- PC-card bridges
    <*> CardBus yenta-compatible bridge support (NEW)
    <*> Cirrus PD6729 compatible bridge support (NEW)
    <*> i82092 compatible bridge support (NEW)
    <*> i82365 compatible bridge support (NEW)
    <*> Databook TCIC host bridge support (NEW)
```

当您完成了内核配置之后，请点击编译与安装.

编译与安装

既然现在您的内核已经配置成功了，那么就是时候编译并且安装它了。退出配置界面并且开启编译进程：

代码 3.8: 编译内核

```
# make && make modules_install
```

当内核编译完成后，复制内核镜像到/boot。您可以给内核任意命名，然后记住它。因为在您配置您的系统引导程序的时候您需要用到它。记得用您内核的名字和版本来替代kernel-2.6.30-gentoo-r5。

代码 3.9: 安装内核

```
# cp arch/i386/boot/bzImage /boot/kernel-2.6.30-gentoo-r5
```

现在请从内核模块一节继续安装。

7.d. 备选：使用genkernel

如果您在阅读这部分，那么您选择了使用genkernel脚本来配置您自己的内核。

现在您的内核源码树已经安装了，是时候用genkernel脚本自动编译您的内核了。genkernel是使用类似安装光盘中的内核配置来配置内核的。这表明当您用genkernel建立内核时，您的系统在启动时候将会如同安装光盘那样识别您所有的硬件。因为genkernel不需要手动配置内核，所以它对于那些不想自己编译特定内核的用户来说是一个理想的解决方案。

现在让我们来看看如何使用genkernel吧。首先，安装genkernel：

代码 4.1: 安装genkernel

```
# emerge genkernel
```

然后，复制安装光盘上的内核配置文件到genkernel搜索配置文件的默认位置：

代码 4.2: 复制安装光盘的配置文件

```
# zcat /proc/config.gz > /usr/share/genkernel/arch/x86/kernel-config
```

现在执行genkernel all编译您的内核源码。请注意，genkernel编译出的内核支持几乎所有硬件，编译需要一段很长的时间。

另外需要注意的是，如果您的启动分区没有使用ext2或者ext3文件系统，您必须使用genkernel --menuconfig all来手动配置您的内核，把您所使用的文件系统编译进内核。（*不能*编译为模块！）。EVMS2和LVM2的用户很可能也需要加上--evms2和--lvm2参数。

代码 4.3: 运行genkernel

```
# genkernel all
```

一旦genkernel运行完成，一个包括全部模块和initrd的内核将被建立。在后面配置引导程序时我们将会用到这个内核和initrd。请记下内核和initrd的名字，因为您将在配置引导程序的时候用到他们。initrd将会在启动真正的系统前自动识别硬件（如同安装光盘一样）。

代码 4.4: 查看内核和initrd的名字

```
# ls /boot/kernel* /boot/initramfs*
```

7.e. 内核模块

配置模块

您应该在/etc/modules.autoload.d/kernel-2.6中列出您需要自动加载的模块。如果您愿意，您也可以加上模块的选项。

要查看所有可用的模块，运行如下的find命令。不要忘记把“<kernel version>”替换成你刚编译好的内核版本：

代码 5.1: 查看所有可用的模块

```
# find /lib/modules/<kernel version>/ -type f -iname '*.o' -or -iname '*.ko' | less
```

例如，要自动加载3c59x.ko模块，编辑kernel-2.6文件然后写入模块的名字。

代码 5.2: 编辑/etc/modules.autoload.d/kernel-2.6

```
# nano -w /etc/modules.autoload.d/kernel-2.6
```

代码 5.3: /etc/modules.autoload.d/kernel-2.6

```
3c59x
```

现在请从配置您的系统一章来继续您的安装。

8. 配置系统

8.a. 文件系统信息

fstab是什么？

在Linux系统下，系统所用到的所有分区都必须在/etc/fstab文件中指明。这个文件包含了这些分区的挂载点（在系统目录树中的位置）、挂载方法和特殊挂载选项（是否自动挂载，是否可以用户挂载等）。

创建/etc/fstab

/etc/fstab使用一种特殊语法格式。每行都包含六个字段。这些字段之间由空白键（空格键，tab键，或者两者混合使用）分隔。每个字段都有自己的含意：

- 第一个字段是对**分区**的描述，也就是设备文件的路径
- 第二个字段是分区**挂载点**，也就是分区应该挂载到的地方
- 第三个字段给出分区所用的**文件系统**
- 第四个字段给出的是挂载分区时mount命令所用的**挂载选项**。由于每个文件系统都有自己的挂载选项，我们建议你阅读mount手册（man mount）以获得所有挂载选项的列表。多个挂载选项之间是用逗号分隔的。
- 第五个字段是给dump使用的，用以决定这个分区是否需要**dump**。一般情况下，你可以把该字段设为0（零）。
- 第六个字段是给fsck使用的，用以决定系统非正常关机之后文件系统的检查顺序。根文件系统应该为1，而其它的应该为2（如果不需要文件系统自检的话可以设为0）。

重要： Gentoo系统默认的/etc/fstab文件*不是有效的fstab文件*。你必须创建自己的/etc/fstab。

代码 1.1: 打开/etc/fstab

```
# nano -w /etc/fstab
```

让我们看看/boot分区的挂载选项是怎么写的。这仅仅是个例子，如果你没有或者不能创建/boot，请不要复制它。

在我们默认的x86分区例子中，/boot一般 为/dev/sda1分区，且采用*ext2*文件系统。 这个分区在引导过程中需要自检，因此我们这样写：

代码 1.2: /etc/fstab中/boot行的一个例子

```
/dev/sda1   /boot     ext2    defaults        1 2
```

为了提高系统的安全性，一部分用户不希望/boot分区自动挂载。这些用户应该用noauto替换defaults。这就表示

用户每次使用该分区时，需要手动挂载。

增加符合你分区方案的规则，为你的光驱（当然，如果你有其他分区或者驱动器，也为它们加上）添加挂载规则。

现在就参考以下*例子*创建你的/etc/fstab：

代码 1.3: /etc/fstab的一个完整例子

```
/dev/sda1   /boot        ext2    defaults,noatime     1 2
/dev/sda2   none         swap    sw                   0 0
/dev/sda3   /            ext3    noatime              0 1

/dev/cdrom  /mnt/cdrom   auto    noauto,user          0 0
```

auto选项可以使mount猜测文件系统（推荐对于可移动设备采用这个选项，因为它们可能采用很多不同的文件系统），而user选项使得非root用户可以挂载光驱。

为了提高性能，大部分用户会添加noatime挂载选项。由于不记录该分区中文件的访问时间（一般来说你并不需要知道它），这个选项能够提高系统速度。

请再次确认你的/etc/fstab文件是正确的，保存并退出，继续下面的内容。

8.b. 网络信息

主机名、域名等

用户必须要做的事情之一就是命名自己的机器。尽管这看上去很容易，但是*很多*用户觉得为他们的Linux机器起一个合适的名字是很难的。为了加快事情的进度，你应该知道你所命名的所有名字都是可以在今后重新修改的。因此，你可以简单命名你的系统为tux，域名为homenetwork。

代码 2.1: 设定主机名

```
# nano -w /etc/conf.d/hostname

（将HOSTNAME的变量值设定为主机名）
HOSTNAME="tux"
```

第二，*如果*你需要一个域名，在/etc/conf.d/net中设定。只有你的ISP或者网络管理员说你需要一个域名，或者你有一个DNS服务器但是没有DHCP服务器的时候，你才需要域名。如果你的网络是DHCP分配IP，那么你不需要理会DNS和域名的问题。

代码 2.2: 设定域名

```
# nano -w /etc/conf.d/net

（设定dns_domain的变量值为你的域名）
dns_domain_lo="homenetwork"
```

注意： 如果你选择不设定域名，你可以去掉登录界面上的这条信息“This is hostname.(none)”。你只需要修改/etc/issue，把字符串.\O从该文件里删掉即可。

如果你有一个NIS域（如果你不知道这是什么，就说明你没有），你也需要定义一个：

代码 2.3: 设定NIS域名

```
# nano -w /etc/conf.d/net

（设定nis_domain的变量值为你的NIS域名）
nis_domain_lo="my-nisdomain"
```

注意： 如果想知道更多关于DNS和NIS配置的信息，可以看/etc/conf.d/net.example当中的例子。当然，你也可以安装openresolv来帮助设置DNS、NIS。

配置你的网络

在准备说“嘿，我们已经配置过网络”之前，你应该记得在开始安装Gentoo之初所设置的网络配置是仅仅为了安装而设置的。现在你所要设置的是Gentoo系统的永久网络配置。

注意： 更多关于网络配置的详细信息，包括网卡绑定、网桥、802.1Q VLANs和无线网络在内的高级配置会在Gentoo网络配置这一部分介绍.

/etc/conf.d/net当中收集了所有的网络信息。尽管这个文件采用直接易懂的语法，如果你还是因为觉得不够直观而完全不知道如何手动进行网络配置的话，请不用担心，我们将一一解释。在/etc/conf.d/net.example中有一个详细注释过的例子，它涵盖了许多种类不同的配置。

系统默认使用DHCP。如果使用DHCP的话，你需要安装一个DHCP客户端。这个将在稍后的安装必要的系统工具部分介绍。但是不要忘记安装一个DHCP客户端。

如果你需要配置你的网络连接，不管是因为你是需要指定DHCP选项还是你根本不想采用DHCP，请使用你喜欢的编辑器（在这个例子中用的是nano）打开/etc/conf.d/net：

代码 2.4: 打开/etc/conf.d/net准备编辑

```
# nano -w /etc/conf.d/net
```

你会看到以下的文件:

代码 2.5: 默认的/etc/conf.d/net

```
# This blank configuration will automatically use DHCP for any net.*
# scripts in /etc/init.d.  To create a more complete configuration,
# please review /etc/conf.d/net.example and save your configuration
# in /etc/conf.d/net (this file :]!).
```

为了输入你自己的IP地址，子网掩码和网关，你需要设置config_eth0和routes_eth0：

代码 2.6: 手动为eth0设置IP信息

```
config_eth0=( "192.168.0.2 netmask 255.255.255.0 brd 192.168.0.255" )
routes_eth0=( "default via 192.168.0.1" )
```

如果你使用DHCP，请定义一下config_eth0：

代码 2.7: 让eth0自动获得IP地址

```
config_eth0=( "dhcp" )
```

请阅读/etc/conf.d/net.example以得到所有选项的列表。如果你需要设定特殊的DHCP选项，请参考你的DHCP客户端的手册页。

如果你有多个网络接口，那么重复之前对于config_eth1、config_eth2等的操作步骤。

现在可以保存配置并且退出，继续下面的安装和配置。

在启动时自动启用网络

为了在启动时自动激活网络接口，你必须添加这些到default运行级别。

代码 2.8: 添加net.eth0到默认的运行级别

```
# rc-update add net.eth0 default
```

如果你有多个网络接口，你需要为他们创建合适的net.eth1、net.eth2等启动脚本。你可以用ln来做这个。

代码 2.9: 创建额外的启动脚本

```
# cd /etc/init.d
# ln -s net.lo net.eth1
# rc-update add net.eth1 default
```

记下网络信息

现在你需要告诉Linux有关你的网络的信息。这需要在/etc/hosts文件中定义，它将帮助你将那些无法被域名解析器解析的主机名解析成IP地址。你需要定义你自己的系统。如果你不想启用内部DNS系统的话，你也需要定义内部网络上的其它系统。

代码 2.10: 打开/etc/hosts

```
# nano -w /etc/hosts
```

代码 2.11: 填入网络信息

```
（这里定义的是现在的系统）
127.0.0.1       tux.homenetwork tux localhost

（定义你网络上的其它系统。如果你要用这种方式进行定义的话,它们必须有静
态IP。）
192.168.0.5     jenny.homenetwork jenny
192.168.0.6     benny.homenetwork benny
```

保存并且退出编辑器，继续下面的过程。

如果你没有PCMCIA，你可以跳过以下内容进入系统信息了。 PCMCIA用户应该读一下以下关于PCMCIA的内容。

可选：启用PCMCIA

PCMCIA用户首先应该安装pcmciautils软件包。

代码 2.12: 安装pcmciautils

```
# emerge pcmciautils
```

8.c. 系统信息

Root密码

首先我们键入以设置root密码：

代码 3.1: 设置root密码

```
# passwd
```

系统信息

Gentoo使用/etc/rc.conf来做通用的、系统级的配置。打开/etc/rc.conf并好好读读这个文件中的注解：）

代码 3.2: 打开/etc/rc.conf

```
# nano -w /etc/rc.conf
```

当你完成对/etc/rc.conf的配置后，保存并退出。

正如你所看到的，为了帮助你完成必要变量的配置，这个文件有丰富的注释信息。你可以让你的系统使用unicode并定义你的默认编辑器和你的显示管理器（比如gdm或者kdm）。

Gentoo用/etc/conf.d/keymaps来处理键盘设置。编辑它就可以设置你的键盘。

代码 3.3: 打开/etc/conf.d/keymaps

```
# nano -w /etc/conf.d/keymaps
```

KEYMAP这个变量要特别注意。如果你选择了错误的KEYMAP，在你敲击键盘的时候会有奇怪的结果。

完成/etc/conf.d/keymaps的配置之后，保存并退出。

Gentoo使用/etc/conf.d/clock来设置时钟选项。根据你的需要来编辑它。

代码 3.4: 打开/etc/conf.d/clock

```
# nano -w /etc/conf.d/clock
```

如果你机器上的钟不用UTC，你需要在文件钟加上CLOCK="local"。否则，你的时钟就有可能出现偏差。

完成对/etc/conf.d/clock的配置后，保存并且退出。

请继续阅读安装必要的系统工具。

9. 安装必要的系统工具

9.a. 系统日志工具

因为有一些工具提供给用户的功能比较类似，它们就没有包含在*stage3*当中。现在就是你选择安装哪一个的时候了。

你首先需要决定的就是系统日志工具。Unix和Linux在日志记录功能方面有良好的传统——如果你愿意的话你可以把系统发生的所有事件都记录到日志文件中。这些功能就是通过*系统日志工具*来完成的。

Gentoo提供了多种系统日志工具可供选择。这当中有sysklogd（传统的系统日志守护进程），syslog-ng（一个高级系统日志工具），metalog（一个可以灵活配置的系统日志工具）。Portage内或许还有其他的系统日志工具——我们的可用软件包数量是以天为单位在增加的。

如果你打算使用sysklogd或者syslog-ng你很可能会随后希望安装logrotate，因为这些系统日志工具并没有提供系统日志文件的滚动功能。

要安装你所选择的系统日志工具，你可以用emerge命令安装它，并使用rc-update将它加入default运行级别。以下就是一个安装syslog-ng的例子。当然你要把它换成你的系统日志工具：

代码 1.1: 安装一个系统日志工具

```
# emerge syslog-ng
# rc-update add syslog-ng default
```

9.b. 可选：Cron守护进程

接下来你可以选择cron守护进程。尽管这是可选的并且不是系统所必须的，但是最好能够安装一个。那么，什么是cron守护进程呢？cron守护进程执行预定的命令。如果你需要执行一些有规律（例如每天、每周或者每月）的命令，这就会非常有用。

Gentoo提供了三个可选的cron守护进程：dcron、fcron和vixie-cron。安装这其中一个的方法和安装一个系统日志工具的方法类似。但是，dcron和fcron需要额外的配置命令，即crontab /etc/crontab。如果你不知道应该选择哪个，那么就用vixie-cron好了。

我们对无网络安装只提供了vixie-cron。如果你希望安装其它的cron守护进程，那你只能等待稍后安装了。

代码 2.1: 安装一个cron守护进程

```
# emerge vixie-cron
# rc-update add vixie-cron default
（只有在使用dcron或fcron时需要） # crontab /etc/crontab
```

9.c. 可选：文件索引

如果你想索引你的系统文件使得你能够使用locate工具很快定位它们，你需要安装sys-apps/slocate。

代码 3.1: 安装slocate

```
# emerge slocate
```

9.d. 文件系统工具

根据你所使用的文件系统的不同，你需要安装必须的文件系统工具（用于检查文件系统完整性、创建额外的文件系统等）。请注意管理ext2/ext3文件系统的工具（e2fsprogs）已经做为系统的一部分被安装了。

以下的表格列出了特定文件系统所需要安装的工具。

文件系统	工具	安装命令
XFS	xfsprogs	emerge xfsprogs
ReiserFS	reiserfsprogs	emerge reiserfsprogs
JFS	jfsutils	emerge jfsutils

如果你是EVMS用户，你还需要安装evms：

代码 4.1: 安装EVMS工具

```
# USE="-gtk" emerge evms
```

USE="-gtk"参数会避免安装一些相关依赖的基于GTK的程序。如果你希望使用evms的图形工具，你可以重新编译evms。

9.e. 网络工具

如果你不需要任何其它网络相关的工具（例如ppp或dhcp客户端）可以跳过这部分内容继续进入配置引导程序

可选：安装一个DHCP客户端

如果你需要Gentoo为你的网卡自动获得IP地址，你需要安装dhcpcd（或者任何其它的DHCP客户端——参见网络模块当中的DHCP客户端列表）。如果你不安装的话，你安装完Gentoo系统后可能会无法连接网络。

代码 5.1: 安装dhcpcd

```
# emerge dhcpcd
```

可选：安装PPPoE客户端

如果你需要ppp来连接网络，你需要安装它。

代码 5.2: 安装ppp

```
# emerge ppp
```

现在进入配置引导程序部分。

10. 配置引导程序

10.a. 做出您的选择

介绍

现在，您已经配置并编译好了内核，必需的一些系统配置文件也已经就位，是时候安装一个叫做*引导程序*的程序来“引燃”您的内核并启动系统了。

对于x86，Gentoo Linux提供了GRUB和LILO。

但在安装bootloader之前，我们要告诉您如何配置framebuffer（前提当然是您想使用它）。在framebuffer的帮助下，您系统的Linux命令行将拥有（有限的）图形特性（例如使用Gentoo提供的漂亮的bootsplash图片）。

可选：Framebuffer

*如果*您已经为您的内核加上了framebuffer支持（或者您使用genkernel的默认内核配置），您就可以在bootloader配置文件中加入一个video内核参数来激活它。

首先，您需要知道您使用的framebuffer设备。您应该使用uvesafb作为*VESA驱动*.

video语句用来控制framebuffer的显示参数。此语句中需指定欲使用的framebuffer驱动以及你想启用的控制语句。/usr/src/linux/Documentation/fb/uvesafb.txt文件里列出了所有变量。最常用的选项是：

控制选项	描述
ywrap	假设显卡支持显存回卷操作（也就是说显存里的数据填充到尽头的时候会接着从起始处继续）
mtrr:n	设置MTRR寄存器。 n 可以是： 0 - 禁用 1 - 不使用cache 2 - write-back 3 - write-combining 4 - write-through
模式	设定分辨率，颜色深度和刷新率。 例如，1024x768-32@85对应的分辨率是1024x768，32位色深和85Hz的刷新率。

最后这个参数的内容可能类似这样video=uvesafb:mtrr:3,ywrap,1024x768-32@85。将它记下来；很快您将用到它。

接下来要继续安装GRUB*或*LILO。

10.b. 默认：使用GRUB

了解GRUB的术语

理解GRUB最重要的在于熟悉它如何表述硬盘驱动器和分区。在GRUB中，您的Linux分区/dev/sda1将很可能都被称为(hd0,0)。请注意hd0,0两边的括号——它们是必须加上的。

硬盘是从0而不是从“a”开始，分区从0而不是1开始。要明确的是，hd设备只指硬盘，而不包括atapi-ide设备，例如cdrom播放器和刻录机。同样的标识构造方式也适用于SCSI驱动器。（正常情况下，除非BIOS被配置为从SCSI驱动器引导，否则SCSI硬盘的标识数字会比IDE的高。）当您要求BIOS从一个不同的硬盘引导时（例如您的主IDE接口上的从盘），*那个*硬盘就会被视为hd0。

假设您有一个硬盘是/dev/sda，还有两个分别是/dev/sdb和/dev/sdc，那么/dev/hdd7将会被GRUB标识为(hd1,6)。听起来这有些不合常理，确实如此。但正如我们即将见到的，GRUB为您提供了一个tab补全机制，这将大大方便您的操作。即使您拥有许多的硬盘和分区，而且对GRUB的数字标识方案不甚了解，也不用怕。

我们已经有了一些感觉了，是时候安装GRUB了。

安装GRUB

要安装GRUB，首先让我们emerge它：

代码 2.1: 安装GRUB

```
# emerge grub
```

尽管现在已经安装完GRUB，我们仍需要为其写一个配置文件，并将其安置到硬盘的主引导记录中，使它能自动引导您新创建的内核。您可以使用nano（或其他可用的编辑器）来创建配置文件/boot/grub/grub.conf：

代码 2.2: 创建/boot/grub/grub.conf

```
# nano -w /boot/grub/grub.conf
```

现在我们将详细地写一个grub.conf。您在下面的内容中可以找到两个可能的针对本指南中前面章节给出的分区方式例子的grub.conf。我们只详细地评述第一个grub.conf。确保您使用*您自己的*内核镜像文件名，以及如果有的话，*您自己的*initrd镜像文件名。

- 第一个grub.conf是为未使用genkernel来构建内核的用户准备的
- 第二个grub.conf则是为使用genkernel来构建内核的用户准备的

注意： Grub根据BIOS来决定设备名。如果你改变了你的BIOS设定，你的设备字母和序号也会改变。比如如果你改变了你的设备的启动顺序，你就需要改变你的grub配置。

注意： 如果您使用JFS来作为root文件系统，您就*必须*在kernel那行添加上“ro”参数，因为JFS文件系统在其允许被加载为可读写状态前需要使用这个参数来重放它的日志。

代码 2.3: 不使用genkernel的用户的grub.conf

```
# 默认选择哪个列表来引导。0表示第一个， 1表示第二个，以此类推。
default 0
# 引导默认列表前等待多少秒
timeout 30
# 使用漂亮、“臃肿”的spalsh图像来增加一点趣味:)
# 如果您没有安装显卡，请将这行注释掉
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title Gentoo Linux 2.6.30-r5
# 内核镜像（或者操作系统）所在分区
root (hd0,0)
```

```
kernel /boot/kernel-2.6.30-gentoo-r5 root=/dev/sda3

title Gentoo Linux 2.6.30-r5 (rescue)
# 内核镜像（或者操作系统）所在分区
root (hd0,0)
kernel /boot/kernel-2.6.30-gentoo-r5 root=/dev/sda3 init=/bin/bb

# 接下来的四行只有在您与Windows系统进行双启动的情况下才需要。
# 本例中，windows系统位于/dev/sda6。
title Windows XP
rootnoverify (hd0,5)
makeactive
chainloader +1
```

代码 2.4: 使用genkernel的用户的grub.conf

```
default 0
timeout 30
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title Gentoo Linux 2.6.30-r5
root (hd0,0)
kernel /boot/kernel-genkernel-x86-2.6.30-gentoo-r5 root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev/sda3
initrd /boot/initramfs-genkernel-x86-2.6.30-gentoo-r5

# 只有在双启动的情况下才需要以下内容
title Windows XP
rootnoverify (hd0,5)
makeactive
chainloader +1
```

如果您使用与此不同的分区方案和 / 或内核镜像，请相应地进行调整。无论怎样，请确保紧跟在GRUB设备（例如(hd0,0)）后面的是相对于挂载点的路径，而不是根目录。换句话说，(hd0,0)/grub/splash.xpm.gz的实际目标是/boot/grub/splash.xpm.gz，因为(hd0,0)就是/boot分区。

另外，如果您选择使用不同的分区方案并且没有将/boot置于一个独立的分区，上述示例代码中/boot前缀实际上是*必需*的。如果您使用我们所建议的分区方案，则不一定非要使用/boot前缀，但即便加了它，由于/boot分区里有一个boot符号链接指向当前目录，也可以正常工作。简单的说，无论您是否定义了一个独立的/boot分区，上述例子应该都能正常工作。

如果您需要传递任何其他的参数给内核，简单地在内核命令后面加上它们就可以了。我们已经传递了一个参数（root=/dev/sda3或real_root=/dev/sda3），但您也可以加上其他的，例如我们前面讨论过的为framebuffer加上video参数。

如果你的引导程序配置文件包含real_root参数，请使用real_rootflags参数来设置根文件系统的挂载选项。

如果您使用的是2.6.7或更高版本的内核，并且您使用硬盘跳线来使主板的BIOS能正确处理大硬盘，那么在启动内核时您还需要附加sda=stroke的参数。请把sda替换成需要这个参数的设备。

genkernel用户需知道他们的内核使用与安装光盘相同的启动参数。例如，若您有SCSI设备时，您就需要添加doscsi这个内核启动参数。

现在请保存grub.conf文件并退出。您仍需要将GRUB安装到MBR（主引导记录），以便重启时GRUB能自动执行。

GRUB的开发者推荐使用grub-install。尽管如此，如果在某些情况下grub-install无法正常工作，您依然可以选择进行手动安装GRUB。

从默认：使用grub-install安装GRUB或者备选：使用手动指令安装GRUB继续。

默认：使用grub-install安装GRUB

为了安装GRUB，您将需要执行grub-install命令。尽管如此，当我们处于chroot的环境时，grub-install并不能正常的工作。我们还需要创建一个/etc/mtab，在里面列出所有已加载的文件系统。幸运的是，有一个简单的方法来完成这个任务——将/proc/mounts拷贝成/etc/mtab，如果您没有创建一个独立的boot分区，请排除rootfs行。下面的命令在两种情况下都可以正常工作：

代码 2.5: 创建/etc/mtab

```
# grep -v rootfs /proc/mounts > /etc/mtab
```

现在我们就可以用grub-install来安装GRUB了：

代码 2.6: 执行grub-install

```
# grub-install --no-floppy /dev/sda
```

如果您还有更多与GRUB相关的问题，请查阅GRUB FAQ或者GRUB手册。

从重启系统章节继续。

备选：使用手动指令安装GRUB

您可以通过输入grub来开始配置GRUB。呈现在您面前的将是grub命令提示符grub>。现在，您需要输入正确的命令

来将GRUB引导记录安装到您的硬盘中。

代码 2.7: 启动GRUB shell

```
# grub --no-floppy
```

注意： 如果您的系统中没有任何的软盘驱动器，在上面的命令后面加上--no-floppy选项，防止grub检测（实际上不存在的）软驱。

在此示例配置中，我们想让我们安装的GRUB可以从boot分区/dev/sda1读取信息，同时我们将GRUB引导记录安装到硬盘的MBR（主引导记录）中，这样当打开电脑时您首先可以看到的就是GRUB引导的界面。当然，如果您安装时没有按照示例配置来做，请相应的更改一些命令。

GRUB的tab自动补全机制可以在GRUB中使用。例如，如果您输入“root (”之后按下TAB键，一个设备列表就会呈现在您面前（例如hd0）。如果您输入“root (hd0,”之后按下TAB键，系统又将返回一个已有的分区列表，以便您选择（例如hd0,0）。

利用tab补全，设置GRUB将不会那么困难。现在我们继续设置GRUB，如何？ :-)

代码 2.8: 将GRUB安装到硬盘主引导记录中

```
grub> root (hd0,0)      (指定您的/boot目录所在分区)
grub> setup (hd0)       (将GRUB安装到硬盘主引导记录)
grub> quit              (退出GRUB shell)
```

注意： 如果您想将GRUB安装到某一个分区而不是硬盘主引导记录中，您需要调整setup命令，使其指向正确的分区。例如，如果您要将GRUB安装到/dev/sda3，对应的命令应该是setup (hd0,2)。但是只有少数用户需要这么做。

如果您还有更多与GRUB相关的问题，请查阅GRUB FAQ或者GRUB手册。

从重启系统章节继续。

10.c. 备选：使用LILO

安装LILO

LILO，是LInuxLoader的简称，它是一个可靠且确实能担当重任的Linux的bootloader。尽管如此，它缺少一些GRUB具有的特性（这正是目前GRUB之所以流行的原因）。LILO仍被人们使用的原因在于：在一些系统中，GRUB不能工作而LILO可以。当然，也有一些用户是因为熟悉它而仍坚持使用它。无论是什么原因，Gentoo对两个bootloader都提供了支持，看到这里显然您已经选择了使用LILO。

安装LILO是件轻而易举的事；只需emerge就好了。

代码 3.1: 安装LILO

```
# emerge lilo
```

配置LILO

要配置LILO，您必须创建/etc/lilo.conf文件。打开一个您喜欢的编辑器（为保持一致性，在本手册中我们使用nano）然后创建它。

代码 3.2: 创建/etc/lilo.conf

```
# nano -w /etc/lilo.conf
```

前面章节中我们已经提醒您记下您所创建的内核镜像文件名。在接下来的示例lilo.conf中，我们使用的是示例的分区方案。这里有两个独立的部分：

- 一个是为那些不使用genkernel建立内核的用户准备的
- 另一个是为那些使用genkernel建立内核的用户准备的

请确保您使用的是*您自己的*内核镜像的文件名，以及（如果用到的话）*您自己的*initrd镜像的文件名。

注意： 如果您使用JFS来作为root文件系统，您就*必须*在每个引导项目的后面添加上形如append="ro"的一行内容，因为JFS文件系统在其允许被加载为可读写状态前需要使用这个参数来重放它的日志。

代码 3.3: 示例/etc/lilo.conf

```
boot=/dev/sda             # 将LILO安装到硬盘主引导扇区
prompt                    # 给用户选择其他引导项目的机会
timeout=50                # 引导默认引导项目前等待5秒钟
default=gentoo            # 当等待结束后引导“gentoo”项目

# 给不使用genkernel的用户的
image=/boot/kernel-2.6.30-gentoo-r5
  label=gentoo            # 我们为此引导项目起的名字
  read-only               # 以只读root状态开始。此处不要改变！
  root=/dev/sda3          # 定位root文件系统

image=/boot/kernel-2.6.30-gentoo-r5
```

```
    label=gentoo.rescue     # 我们为此引导项目起的名字
    read-only               # 以只读root状态开始。此处不要改变!
    root=/dev/sda3          # 定位root文件系统
    append="init=/bin/bb"   # 启动Gentoo的静态急救shell

  # 给使用genkernel的用户的
  image=/boot/kernel-genkernel-x86-2.6.30-gentoo-r5
    label=gentoo
    read-only
    root=/dev/ram0
    append="init=/linuxrc ramdisk=8192 real_root=/dev/sda3"
    initrd=/boot/initramfs-genkernel-x86-2.6.30-gentoo-r5

  # 下边的两行只有在您与Windows系统进行双启动的情况下才需要
  # 在本例中，Windows位于/dev/sda6
  other=/dev/sda6
    label=windows
```

注意： 如果您使用了与此不同的分区方案和/或者内核镜像，请进行相应的调整。

如果您需要为内核传递任何额外的参数，可以在相应的启动项目里添加一个append语句。例如，我们加上video参数来启用framebuffer：

代码 3.4: 使用append来添加内核参数

```
image=/boot/kernel-2.6.30-gentoo-r5
  label=gentoo
  read-only
  root=/dev/sda3
  append="video=uvesafb:mtrr,ywrap,1024x768-32@85"
```

如果您使用的是2.6.7或更高版本的内核，并且您使用硬盘跳线来使主板的BIOS能正确处理大硬盘，那么在启动内核时您还需要附加sda=stroke的参数。请把sda替换成需要这个参数的设备。

genkernel用户需知道他们的内核使用与安装光盘相同的启动参数。例如，若您有SCSI设备时，您需添加doscsi这个内核启动参数。

现在让我们保存文件并退出。要最终完成，您还得运行/sbin/lilo，这样LILO才能将/etc/lilo.conf应用到您的系统（也就是将其自身安装到硬盘中）。请记住每当你安装了一个新内核或改变了菜单的任何内容时，您也需要再运行一次/sbin/lilo。

代码 3.5: 完成LILO的安装

```
# /sbin/lilo
```

如果您还有更多与LILO有关的问题，请参阅它的wikipedia页面。

现在请继续阅读重启系统。

10.d. 重启系统

退出chroot的环境，卸载所有已挂载的分区。然后输入您已经等待多时的那个魔术般的命令：reboot。

代码 4.1: 卸载所有分区和重启

```
# exit
cdimage ~# cd
cdimage ~# umount /mnt/gentoo/boot /mnt/gentoo/dev /mnt/gentoo/proc /mnt/gentoo
cdimage ~# reboot
```

当然您别忘了取出可引导光盘，否则重启后引导的将是这张光盘而不是您的新Gentoo系统。

当重启进入到您新装的Gentoo系统之后，请阅读结束Gentoo的安装来完成剩下的工作。

11. 结束Gentoo的安装

11.a. 用户管理

为日常使用添加一个用户

在Unix/Linux系统中，用root进行工作是一件*危险*的事情，应该尽量避免。因此我们*强烈*推荐您为日常使用添加一个普通用户。

用户所属的组定义了其可以执行的活动。下表中列出了许多您可能希望使用的重要组：

组	描述
audio	允许使用声音设备
cdrom	允许直接使用光驱
floppy	允许直接使用软驱

games	允许运行游戏
portage	允许以普通用户的身份执行emerge --pretend
usb	允许使用USB设备
plugdev	允许挂载和使用可拔插设备，例如摄像头和U盘
video	允许使用视频采集设备和硬件加速
wheel	允许使用su

例如，要创建一个名为john的用户，并使其隶属于wheel组，users组和audio组，请先用root用户登录（只有root有权限创建用户），然后执行useradd命令：

代码 1.1: 添加一个日常使用的用户

```
Login: root
Password: (您的root密码)

# useradd -m -G users,wheel,audio -s /bin/bash john
# passwd john
Password: （输入john用户使用的密码）
Re-enter password: （再次输入以便进行校验）
```

有时，普通用户可能需要以root身份执行一些任务，这时您可以使用su -来临时取得root权限。另外一种方法就是使用sudo包，如果配置正确，使用它将会非常安全。

11.b. 磁盘清理

移除tarball

现在您的gentoo已经结束了安装并完成了重启，如果一切顺利，您就可以将下载下来的stage3 tarball和Portage快照文件从硬盘里清除掉了。请记住它们是被下载到您的/目录中的。

代码 2.1: 移除stage3 tarball

```
# rm /stage3-*.tar.bz2*
```

代码 2.2: 移除Portage快照文件

```
# rm /portage-latest.tar.bz2*
```

12. 下一步该做什么？

12.a. 文档

恭喜！您现在拥有了一个可以运转的Gentoo系统。但接下来往何处去？现在您拥有什么样的选择？先探索些什么？Gentoo为它的用户提供了大量的可能性，因此也有大量已经提供文档（少量没有）的特性。

您无疑应该先看一下Gentoo手册里标题为在Gentoo中工作的那一章节，那里解释了如何保证您随时用上最新的软件，如何安装更多的软件，什么是USE标记，以及Gentoo Init系统是如何工作的等等。

如果您有兴趣对系统进行优化以使其适合桌面应用，或者您想了解如何将您的系统配置成一个完整可用的桌面系统，请参阅我们所提供的全面的Gentoo桌面文档资源。另外，我们也提供本地化指南供您参考，利用它您可以使您的系统看起来更加本地化。

我们还提供了一份值得您阅读的Gentoo安全手册

您如果想得到一份我们已有的可用文档的列表，请查看我们提供的文档资源页面。

12.b. 线上Gentoo

当然也非常欢迎您加入我们的Gentoo论坛或者我们建立的Gentoo IRC 频道中的任意一个。

我们也有一些邮件列表是面对所有用户开放的。加入列表的方法已包含在前述页面中。

现在，我们将保持安静，以便让您享受您的安装过程。:)

B. 使用Gentoo

1. Portage入门

1.a. 欢迎使用Portage

Portage可能是Gentoo在软件包管理方面最瞩目的创新。它所具有的高度灵活性以及其他大量的特性，使其经常被认为是Linux中最好用的软件包管理工具。

Portage完全用Python和Bash写成。由于它们均为脚本语言，所以Portage对于用户而言也就是完全可见的。

绝大部分用户将通过emerge工具来使用Portage。本章节的目的不是在重复emerge的man page里已经提供的内容，如果要完整的了解emerge工具提供的选项，请参考其man page里给出的相应说明。

代码 1.1: 获取emerge的帮助信息

```
$ man emerge
```

1.b. Portage树

Ebuilds

当我们谈到（软件）包的时候，我们通常指的是Portage树为Gentoo用户提供的包的名称。Portage树是*ebuilds*文件的集合，这些文件包含了Portage管理工具维护软件（安装，搜索，查询， ...）时所需要的所有信息，并被默认的放置在/usr/portage目录中。

每当您要求Portage对系统中的软件包执行一些操作的时候，它会以系统中的ebuilds文件作为基础。因此您最好定期更新系统Portage树中的ebuild文件，这样Portage才知道新加入了哪些软件，哪些软件发布了安全更新，等等。

更新Portage树

通常我们使用rsync命令来更新Portage树，它是一个快速的增量性的文件传输工具。另外emerge命令为rsync命令提供了一个相当简单的前端。

代码 2.1: 更新Portage树

```
# emerge --sync
```

如果由于防火墙的限制，您无法进行rsync操作，您仍然可以通过下载我们每天汇总的Portage树快照来更新本地的Portage树。emerge-webrsync命令会自动将最新的Portage快照下载并安装到您的系统中。

代码 2.2: 运行emerge-webrsync

```
# emerge-webrsync
```

1.c. 软件包维护

软件的查询

要在Portage树中查找指的的软件，您可以使用emerge命令里内建的查找功能。默认情况下，emerge --search会返回那些名字与您指定（无论完整或部分）的名字匹配的软件包。

例如，要查找那些名字中包含“pdf”字样的软件：

代码 3.1: 查找名字包含pdf的软件包

```
$ emerge --search pdf
```

而如果要通过软件的描述来查找相应的软件，您可以使用--searchdesc （或者简写为 -S） 开关：

代码 3.2: 查找与pdf相关的软件包

```
$ emerge --searchdesc pdf
```

当您观察输出结果的时候，会注意到它提供了大量信息。因为相关的区域已经进行了明确的标注，我们就不再进一步介绍各个部分的含义：

代码 3.3: 'emerge --search'返回结果的例子

```
*  net-print/cups-pdf
      Latest version available: 1.5.2
      Latest version installed: [ Not Installed ]
      Size of downloaded files: 15 kB
      Homepage:    http://cip.physik.uni-wuerzburg.de/~vrbehr/cups-pdf/
      Description: Provides a virtual printer for CUPS to produce PDF files.
      License:     GPL-2
```

软件的安装

每当您找到一个感兴趣的软件，您可以轻松地用emerge命令安装它。例如，要安装gnumeric：

代码 3.4: 安装gnumeric

```
# emerge gnumeric
```

因为许多应用程序互相依赖，试图安装其中任何一个，可能将会要求同时安装其所依赖的其他相关软件。无需担心，Portage可以很好地处理软件包间的依赖问题。当您安装某个软件的时候，您可以加上--pretend开关，以便观察Portage*将会*同时安装哪些软件。例如：

代码 3.5: 假装安装gnumeric

```
# emerge --pretend gnumeric
```

当您要求Portage安装一个软件的时候，必要时它会自动从internet下载相应的源代码包并默认将其保存在/usr/portage/distfiles目录中，然后将源代码包解压缩，编译并安装软件。如果您仅仅希望Portage将源代码包下载到本地而不安装它们，在emerge命令后加上--fetchonly的选项：

代码 3.6: 下载gnumeric的源代码包

```
# emerge --fetchonly gnumeric
```

查找已安装软件的文档

许多软件包中包含有自己的文档，有些时候doc的USE标记决定了软件包中的自带文档是否会被安装到本地。您可以通过emerge -vp <软件包名称>命令来检查是否存在doc USE标记。

代码 3.7: 检查是否存在doc的USE标记

```
(当然alsa-lib只是一个例子。)
# emerge -vp alsa-lib
[ebuild  N    ] media-libs/alsa-lib-1.0.14_rc1  -debug +doc 698 kB
```

最好的启用doc的方式是在/etc/portage/package.use里对想要启用的包单独启用，这样你就能只获得你想要的软件文档。全局启用此标记已知会导致循环依赖。欲了解更多信息请看USE标志一章。

当一个软件包安装结束后，它的文档通常会存放在/usr/share/doc目录下以软件包名命名的子目录中。您也可以通过equery工具来列出此软件安装后生成的所有文件，这个工具来自于app-portage/gentoolkit 包。

代码 3.8: 定位包的文档

```
# ls -l /usr/share/doc/alsa-lib-1.0.14_rc1
total 28
-rw-r--r--  1 root root  669 May 17 21:54 ChangeLog.gz
-rw-r--r--  1 root root 9373 May 17 21:54 COPYING.gz
drwxr-xr-x  2 root root 8560 May 17 21:54 html
-rw-r--r--  1 root root  196 May 17 21:54 TODO.gz

(或者使用equery来定位您感兴趣的文件:)
# equery files alsa-lib | less
media-libs/alsa-lib-1.0.14_rc1
* Contents of media-libs/alsa-lib-1.0.14_rc1:
/usr
/usr/bin
/usr/bin/alsalisp
(部分输出内容)
```

软件的移除

当您想把一个软件包从系统中移除的时候，使用emerge --unmerge命令。命令执行完成后，*除了*那些在安装软件包后您修改过的配置文件，Portage将会移除此软件包安装到您系统中的所有文件。保留这些修改过的配置文件是为了便于您今后再次使用同一个软件包。

不过，您需要**特别注意**的是：Portage将*不会*检查您要删除的包是否仍被其他的包依赖。但是当您要删除一个可能破坏您系统的重要的软件包时，它还是会给予警告。

代码 3.9: 从系统中删除gnumeric

```
# emerge --unmerge gnumeric
```

当您从系统中移除一个软件包时，之前那些为了满足其依赖关系而自动被一并安装的软件包将会被保留。如果想让Portage找出那些现在可以移除的相关软件包，可以使用emerge的--depclean功能。稍后我们将谈到这点。

更新您的系统

要保持您系统在最佳状态（更不用说安装那些最新的安全更新），您需要定期的更新您的系统。由于Portage只能检查本地Portage树中已有的ebuilds文件，因此您首先应该更新您的Portage树。当您的Portage树更新后，您可以用emerge --update world命令来更新系统。在下一个例子里，我们还会使用--ask开关来控制Portage显示它要更新的软件包列表，并让您决定是否继续更新。

代码 3.10: 更新您的系统

```
# emerge --update --ask world
```

Portage接下来会查找您已经安装的软件包是否有更新版本，但它只会核对那些您已经*明确地*安装过的软件包(也就是在/var/lib/portage/world文件中列出的包），并不会完整去的检查与这些软件产生依赖关系的包是否也需要更新。如果您想更新系统中的*每个软件包*，那就加上--deep参数：

代码 3.11: 更新整个系统

```
# emerge --update --deep world
```

因为有时开发者会为那些您没有明确要求安装（但被其他软件包依赖而装入系统中）的包推出安全更新，我们也推荐偶尔运行一下这个命令。

每当您改变了系统中任何的USE标记后，您可能需要加入--newuse选项。这样Portage将会检查这个USE标记的变动

是否导致需要安装新的软件包或者将现有的包重新编译。

代码 3.12: 执行完整更新

```
# emerge --update --deep --newuse world
```

Meta软件包

Portage树中的一些软件包并没有包含任何实际的内容，而只是用来安装一系列软件包的集合。例如，kde包就是一个包含了一系列与KDE相关的互相依赖的软件包的集合，您可以通过安装它来在系统中搭建起一个完整的KDE环境。

如果您试图从系统中移除一个这样的软件包的集合体，只是单纯地使用emerge --unmerge命令并不能完成您的要求，原因在于这些包的依赖关系仍然保留在系统中。

不用担心，Portage也提供了移除孤立依赖的软件包的功能，但由于软件包间的依赖关系是动态的，您首先需要充分地更新您的整个系统，包括更改USE标记设定而导致的变化。在这之后您可以运行emerge --depclean来移除那些完全没有被其他包依赖的软件包（“orphaned dependencies”）。移除之后你需要重新编译那些曾经与刚刚移除的这些包动态连接过的应用程序，因为实际上这些程序不需要那些包。

所有这些可以用以下三个命令来实现：

代码 3.13: 移除孤立依赖的软件包

```
# emerge --update --deep --newuse world
# emerge --depclean
# revdep-rebuild
```

revdep-rebuild工具由gentoolkit包提供；使用前别忘了首先emerge它：

代码 3.14: 安装gentoolkit包

```
# emerge gentoolkit
```

1.d. 当Portage抱怨的时候...

有关SLOT，Virtual，分支，体系结构和Profile

正如我们之前指出的那样，Portage是一个非常强大并支持许多特性的软件包管理工具，而这是其他类似工具所欠缺的。为了理解这一点，我们为您粗略地解释一些Portage的面貌。

通过使用Portage，一个软件的不同版本可以共存于一个系统中。其他发行版倾向于直接在软件包名字中包含版本号（例如freetype 和 freetype2），Gentoo的Portage使用一种我们称之为*SLOT*的技术来实现这种并存。一个ebuild为它自身的版本声明了一个确切的SLOT。具有不同SLOT的同一软件的Ebuild可以共存于同一个系统中。例如，上例中那个freetype包就拥有不同的ebuilds，里面分别有SLOT="1"和SLOT="2"的标志。

有一些不同的软件包提供了类似的功能。比如metalogd，sysklogd和syslog-ng都是系统日志记录工具。那些依赖于“系统日志记录工具”的程序并不能随便的依赖于其中之一，比如metalogd，因为其他的系统日志工具可能也是很好的选择。好在Portage允许使用*虚拟包*：每一个系统日志记录工具都可以提供virtual/syslog包，因此应用程序们可以设定成仅仅依赖于virtual/syslog即可。

Portage树中的软件可以存在于不同的分支中。您的系统默认只会接受那些Gentoo认为稳定的软件包。绝大多数新提交的软件会被添加到测试分支里。这意味着在此软件被标示为稳定版前需要进行更多的测试。尽管您可以看到那些软件的ebuilds已经加入Portage树，在它们未被加入稳定分支前Portage将不会安装它们。

有些软件只在某几个体系结构上可用。或者在其他体系结构中还不能运行，或者仍需要对其进行更多的测试，或者将软件提交到Portage树中的开发者还不能确定这个软件能否运行于其他体系结构。

每一个Gentoo安装都依附于一个确定的profile，此文件里除了其他信息外还包含了一个正常工作的系统需要的软件包的列表。

被阻挡的包

代码 4.1: Portage关于被阻挡的包的警告(使用 --pretend参数)

```
[blocks B     ] mail-mta/ssmtp (is blocking mail-mta/postfix-2.2.2-r1)
```

代码 4.2: Portage关于被阻挡的包的警告(不使用 --pretend参数)

```
!!! Error: the mail-mta/postfix package conflicts with another package.
!!!        both can't be installed on the same system together.
!!!        Please use 'emerge --pretend' to determine blockers.
```

Ebuilds文件中包含了特定的字段，里面为Portage提供了此软件的各种依赖关系的信息。总计有两种可能的依赖关系：一种是编译依赖，在DEPEND区域进行声明;另一种是“运行时”依赖，在RDEPEND区域中进行声明。如果上述两种依赖关系中任何一个明确指明某个实体或者虚拟包（译注：可能已安装和正要安装）与要安装的包*不*相容的时候，就会阻挡软件的安装。

为了使安装得以继续进行，您可以选择不安装这个软件包，或者先将发生冲突的包卸载。例如，在我们给出的这个例子中，您可以选择不安装postfix，或者先卸载ssmtp。

你也可能会遇到某些特定版本的包被屏蔽的情况，比如<media-video/mplayer-bin-1.0_rc1-r2。在这种情况

下，升级到一个更新的版本就能解决问题。

也有可能两个需要安装的包互相阻挡。这种少见的情况下，您应该明确自己为什么需要同时安装它们。绝大多数时候您只需要安装它们之中的一个就可以了。如果不是这样，请您到Gentoo的bug跟踪系统中提交一个bug，以便我们找出解决方案。

被屏蔽的包

代码 4.3: Portage关于被屏蔽的包的警告

```
!!! all ebuilds that could satisfy "bootsplash" have been masked.
```

代码 4.4: Portage关于被屏蔽的包的警告——原因

```
!!! possible candidates are:

- gnome-base/gnome-2.8.0_pre1 (masked by: ~x86 keyword)
- lm-sensors/lm-sensors-2.8.7 (masked by: -sparc keyword)
- sys-libs/glibc-2.3.4.20040808 (masked by: -* keyword)
- dev-util/cvsd-1.0.2 (masked by: missing keyword)
- games-fps/unreal-tournament-451 (masked by: package.mask)
- sys-libs/glibc-2.3.2-r11 (masked by: profile)
```

当您想安装一个对于您系统不可用的软件包。您会收到类似这样的屏蔽错误提示。您应该试着安装那些对于您系统可用的程序或者等待那些不可用的包被置为可用的。通常一个软件包被屏蔽的原因在于：

- **~arch keyword** 意味着这个软件没有经过充分的测试，不能进入稳定分支，请等待一段时间后在尝试使用它。
- **-arch keyword**或**-* keyword** 意味着这个软件不能工作在您机器的体系结构中。如果您确信它能工作那么请到我们的bugzilla网站提交一个bug报告。
- **missing keyword** 意味着这个软件还没有在您机器的体系结构中进行过测试。您可以咨询相应体系结构移植小组是否能对它进行测试，或者您自己为他们进行这样的测试并将您得到的结论提交到我们的bugzilla网站。
- **package.mask** 意味着这个软件被认为是损坏的，不稳定的或者有更严重的问题，它被故意标识为“不应使用”。
- **profile** 意味着这个软件不适用于您的profile。安装这样的应用软件可能会破坏您的系统，或者只是不能与您使用的profile相兼容。

依赖缺失

代码 4.5: Portage关于依赖缺失的警告

```
emerge: there are no ebuilds to satisfy ">=sys-devel/gcc-3.4.2-r4".

!!! Problem with ebuild sys-devel/gcc-3.4.2-r2
!!! Possibly a DEPEND/*DEPEND problem.
```

这表示您正尝试安装的应用程序依赖于您的系统不可用的另外一些软件包。请到bugzilla查看是否有此问题的记录，如果没有查找到相关信息的话请提交一个报告。除非您的系统混用了不同分支，否则这类问题不应该发生，若发生了那就是一个bug。

意指不明的Ebuild名称

代码 4.6: Portage对于意指不明的Ebuild名称的警告

```
!!! The short ebuild name "aterm" is ambiguous.  Please specify
!!! one of the following fully-qualified ebuild names instead:

    dev-libs/aterm
    x11-terms/aterm
```

您要安装的应用程序对应有多个同名的包。您需要同时指定类别的名称。Portage会列出所有可供选择的名称匹配的包。

循环依赖

代码 4.7: Portage关于循环依赖问题的警告

```
!!! Error: circular dependencies:

ebuild / net-print/cups-1.1.15-r2 depends on ebuild / app-text/ghostscript-7.05.3-r1
ebuild / app-text/ghostscript-7.05.3-r1 depends on ebuild / net-print/cups-1.1.15-r2
```

两个（或多个）您想安装的包由于循环依赖而不能安装。这很可能源于Portage树中的bug。请等一段时间后重新sync再尝试安装。您也可以去bugzilla看看是否已经有此问题的报告，或者提交一个关于它的报告。

下载失败

代码 4.8: Portage关于下载失败的警告

```
!!! Fetch failed for sys-libs/ncurses-5.4-r5, continuing...
(...)
!!! Some fetch errors were encountered.  Please see above for details.
```

当Portage下载指定软件的源代码失败时，它会尝试继续安装其它（若适用）的应用程序。源代码下载失败可能源于镜像服务器没有正确同步，也可能因为ebuild文件给出了错误的下载地址。那些保存源代码的服务器也可能因为某些原因当机。

一小时后重试一次看看问题是否仍然存在。

系统Profile保护

代码 4.9: Portage关于profile中保护的包的警告

```
!!! Trying to unmerge package(s) in system profile. 'sys-apps/portage'
!!! This could be damaging to your system.
```

您要求移除系统核心软件包中的一个。它是您的profile中所列出的必需的软件，因此不能从系统中移除。

Digest验证失败

有时当您试图emerge一个软件包时，它会失败并给出以下信息：

代码 4.10: Digest验证失败

```
>>> checking ebuild checksums
!!! Digest verification failed:
```

这是Portage树中出现了错误的迹象，通常这是由于开发者在向Portage树提交一个软件包时出错造成的。

当digest校验失败的时候，请*不要*尝试自己去为此软件包重新产生digest。使用ebuild foo manifest并不能修复问题，反而几乎肯定会使问题变得更糟。

取而代之的应该是等待一至两个小时以便让开发者来修复Portage树。一般来说错误很有可能马上就会被开发者注意到，但Portage树的修复也需要一点点时间。当您等待的时候，到Bugzilla看看是否已经有人报告了这个问题。如果没有，那就为那个损坏的包提交一个bug报告吧。

一旦在Bugzilla上看到此问题已经修复，您只需要重新sync就可以下载下来那些修复后的digest。

重要： 但值得注意的是：这并*不*意为着您可以短时间内多次重复sync您的portage树（对于同一个rsync服务器）。正如（当您运行emerge --sync时）rsync策略所指出的那样，那些短时间内过于频繁进行多次sync的用户将会被更新服务器禁止访问（一般是将你的IP添加到禁止名单并保留一段指定的时间后才解除）。实际上，最好等到您的下次计划更新的日子再sync，因为这样您不会使rsync服务器过载而影响其他用户的正常使用。

2. USE标记

2.a. USE标记是什么？

USE标记的指导思想

你在安装gentoo（或者是其他发行版，甚至于其他特定操作系统）的时候，你要依据你工作的环境做出选择。服务器跟工作站的组织结构不同，游戏机跟3D工作站也会不一样。

不单只是选择你想要安装的包时如此，选择某一个包需要的特性时同样如此。如果你不需要OpenGL，为什么还要颇费周折的安装OpenGL并在其他包中加入对OpenGL的支持？如果你不想用KDE，而且软件包没有KDE也能完美运行，为什么还要在编译这些包的时候加入KDE支持？

为了帮用户判断什么需要安装或激活，什么不需要；我们希望用户能用简单的方式设定他们自己的环境。这能促使用户判断他真正需要的东西，并让Portage（我们的包管理系统）做出有用的决定的过程变得简单。

USE标记的设定

我们来具体看看USE标记。每一个标记都是代表对某特定概念的支持和依赖关系信息的关键字。如果你设定了某个USE标记，Portage会明白你选择了支持这个关键字（所代表的概念）。当然这同时也改变了这个包的依赖关系信息。

让我们看一个示例：关键字kde。如果你的USE变量里面没有这个关键字，所有具有*可选*KDE支持的包在编译时都*不会*编译KDE支持。所有具有*可选*KDE依赖关系的包在安装时都*不会*（做为一个依赖关系而）安装KDE库。如果你设定了kde关键字，这些包在安装时都*会*编译KDE支持，而且KDE库也会（作为一个依赖关系而）被安装。

通过正确设定关键字，你会得到一个根据你的需要而定制的系统。

有哪些USE标记？

USE标记分两类：*全局*和*局部*USE标记。

- *全局*USE标记适用范围是整个系统，可以被许多包使用。这就是大多数人眼里的USE标记。
- *局部*USE标记只被单个包用来做该包特有的决定。

当前可用的全局USE标记列表可以在网上或者本机的/usr/portage/profiles/use.desc文件里找到。

当前可用的局部USE标记列表可以在本机的/usr/portage/profiles/use.local.desc 文件里找到。

2.b. 使用USE标记

声明永久USE标记

希望您已经意识到了USE标记的重要性，现在我们就让你了解怎样声明USE标记。

就像前面提到的，所有USE标记都声明在USE变量里面。为了让用户能方便地查找和选择USE标记，我们提供了一份*默认*的USE设定。这些设定是我们觉得Gentoo用户通常都要用到的USE标记的集合。这个默认设置在make.defaults文件——你的profile的一部分——里声明。

你的系统使用的profile是符号链接/etc/make.profile所指向的目录。每个profile叠加于某个更大的profile之上，最终的结果是这些profile的并集。初始profile是base profile（/usr/portage/profiles/base）。

让我们看看2004.3 profile的默认设定：

代码 2.1: 2004.3 profile的累积make.defaults USE标记

```
(这个例子是base，default-linux，default-linux/x86和
default-linux/x86/2004.3的设定的并集)
USE="x86 oss apm arts avi berkdb bitmap-fonts crypt cups encode fortran f77
     foomaticdb gdbm gif gpm gtk imlib jpeg kde gnome libg++ libwww mad
     mikmod motif mpeg ncurses nls oggvorbis opengl pam pdflib png python qt
     quicktime readline sdl spell ssl svga tcpd truetype X xml2 xmms xv zlib"
```

就像你看到的那样，这个变量已经包括了非常多的关键字。**不**要通过修改make.defaults文件里的USE变量来满足你的需要：在升级Portage的时候，这个文件将会被破坏（被覆盖）。

要改变这个默认设置，你需要在USE变量里添加或移去关键字。这是通过在/etc/make.conf里定义USE全局变量来实现的。在这个变量里，添加你需要的额外的USE标记，或者移去你不需要的USE标记。后者可通过在标记前面加个负号（“-”）前缀来实现。

例如，要移除对KDE和QT的支持，并添加对ldap的支持，可以在/etc/make.conf 里声明USE如下：

代码 2.2: /etc/make.conf里关于USE设置的一个例子

```
USE="-kde -qt3 -qt4 ldap"
```

为单个包声明USE标记

如果你想要为一个（或者几个）程序而不是系统范围内声明一个USE标记，你需要创建/etc/portage目录（如果没有这个目录的话），然后编辑/etc/portage/package.use文件。通常这是一个文件，不过它也可以是一个目录；请看man portage以获得更多信息。下面的例子假设package.use是一个文件。

比如说，如果你不想全局的启用berkdb支持，但是你想把它应用到mysql，你可以这样：

代码 2.3: /etc/portage/package.use示例

```
dev-db/mysql berkdb
```

你当然也可以直接为某一个程序*禁用*USE标记。比如说，如果你不想要PHP的java支持：

代码 2.4: /etc/portage/package.use第二个示例

```
dev-php/php -java
```

声明临时USE标记

有时，你只想暂时改变一个USE设置。你可以仅仅把USE变量声明成一个环境变量，而不必两次修改/etc/make.conf。但是要记住，当你重新emerge或者升级这个程序的时候（不管是单独地还是作为系统升级的一部分），你的修改都会丢失！

下面的例子我们将在安装seamonkey的时候暂时性地从USE设置中移去java标记。

代码 2.5: 将USE作为一个环境变量使用

```
# USE="-java" emerge seamonkey
```

优先级

当然，我们需要一个明确的先后次序来决定何处的USE设定优先级较高。你肯定不希望在定义了USE="-java"之后，因为某个有更高优先级的设定而导致java仍然被使用。USE设定的优先级顺序是（由低到高）：

1. make.defaults里面的USE默认设定
2. 用户在/etc/make.conf里面的USE设定
3. 用户在/etc/portage/package.use里面的USE设定
4. 作为环境变量的USE设定

运行emerge --info可以看到Portage识别的最终的USE设定。它会列出Portage使用的所有相关变量（包括USE变量）。

代码 2.6: 运行emerge --info

```
# emerge --info
```

在你的整个系统上应用新的USE标记

如果你已经修改了你的USE标记，而且你想用新USE标记更新你的系统，可以使用emerge 的--newuse选项：

代码 2.7: 重新构建你的系统

```
# emerge --update --deep --newuse world
```

然后运行Portage的depclean来移除已经安装到你的旧系统里但是在新USE标记中被废除的条件依赖关系。

警告： 运行emerge --depclean是一项危险的操作，必须小心。请反复检查要删除的包的列表里确定没有你仍然需要的包。下面这个例子里，我们添加了-p选项来depclean——只列出这些包而不删除他们。

代码 2.8: 删除不需要的包

```
# emerge -p --depclean
```

depclean完成之后，运行revdep-rebuild来重新构建那些动态链接到由可能已经删除的包提供的公共对象的程序。revdep-rebuild是gentoolkit包的一部分，不要忘了先emerge它。

代码 2.9: 运行revdep-rebuild

```
# revdep-rebuild
```

这些都完成之后，你的系统就已经应用上了新的USE标记设定。

2.c. 包特有的USE标记

查看可用USE标记

让我们以seamonkey来作例子，看看它接收什么USE标记。我们可以以--pretend和--verbose为选项执行emerge来查看：

代码 3.1: 查看使用的USE标记

```
# emerge --pretend --verbose seamonkey
These are the packages that I would merge, in order:

Calculating dependencies ...done!
[ebuild   R   ] www-client/seamonkey-1.0.7  USE="crypt gnome java -debug -ipv6
-ldap -mozcalendar -mozdevelop -moznocompose -moznoirc -moznomail -moznopango
-moznoroaming -postgres -xinerama -xprint" 0 kB
```

emerge并不是做这件事的唯一工具。事实上，我们有一个专门的包信息工具叫equery，它属于gentoolkit包。所以首先请安装gentoolkit：

代码 3.2: 安装gentoolkit

```
# emerge gentoolkit
```

现在以uses为参数执行equery来查看指定包的USE标记。例如：gnumeric包：

代码 3.3: 用equery来查看当前使用的USE标记

```
# equery --nocolor uses =gnumeric-1.6.3 -a
[ Searching for packages matching =gnumeric-1.6.3... ]
[ Colour Code : set unset ]
[ Legend        : Left column  (U) - USE flags from make.conf  ]
[               : Right column (I) - USE flags packages was installed with ]
[ Found these USE variables for app-office/gnumeric-1.6.3 ]
 U I
- - debug   : Enable extra debug codepaths, like asserts and extra output.
             If you want to get meaningful backtraces see
             http://www.gentoo.org/proj/en/qa/backtraces.xml.
- - gnome   : Adds GNOME support
+ + python  : Adds support/bindings for the Python language
- - static  : !!do not set this during bootstrap!! Causes binaries to be
                    statically linked instead of dynamically
```

3. Portage特性

3.a. Portage特性

Portage有几个附加的特性，它们能够令您的Gentoo之旅更加愉快。这些特性中的大多数依赖于某些能够提高性能、可靠性、安全性等的软件工具。

为了打开或者关闭某一Portage特性您需要编辑/etc/make.conf中的FEATURES变量，这个变量包含不同的特性关键字，用空格分开。在一些情况下您可能还需要额外的安装被这个特性所依赖的工具。

并不是所有Portage所支持的特性都在这里列出。完整的概述，请查阅make.conf手册页。

代码 1.1: 查阅make.conf手册页

```
$ man make.conf
```

查看特性的默认设置，运行emerge --info并且查找FEATURE变量或者用grep显示它：

代码 1.2: 找出已经被设定的特性

```
$ emerge --info | grep FEATURE
```

3.b. 分布式编译

使用distcc

distcc是一个分布式编译程序，可以把编译任务分配给同一网络中的不同机器，这些机器的配置不必完全相同。distcc客户端发送所有必须的信息给所有可利用的distcc服务器（运行distccd的机器）。这样它们每一个都能为客户端编译一部分源码。所导致的效果就是更短的编译时间。

您可以在Gentoo Distcc文档里找到更多的关于distcc的信息（包括如何让它在Gentoo上工作）。

安装distcc

Distcc使用一个图形化监视器来监视您的机器发送出去的编译工作。如果您使用GNOME，那么将“gnome”放入您的USE设置中。但是，如果您不使用GNOME而仍然希望使用这个监视器，那么就把“gtk”放进您的USE设置中。

代码 2.1: 安装distcc

```
# emerge distcc
```

激活Portage支持

添加distcc到/etc/make.conf之中的FEATURES变量。然后，设定您合意的MAKEOPTS变量。一个众所周知的参数是"-jX"，其中X为运行distccd主机（包含当前主机）的CPU数目加上一，但是您也可能用其他的数字来得到更好的结果。

现在运行distcc-config并输入已有的DistCC服务器。作为一个简单例子，我们假设已有的DistCC服务器是192.168.1.102（当前主机）、192.168.1.103和192.168.1.104（两个远端服务器）：

代码 2.2: 配置Distcc，使用已有的三个DistCC服务器

```
# distcc-config --set-hosts "192.168.1.102 192.168.1.103 192.168.1.104"
```

当然，也不要忘了运行distccd系统服务：

代码 2.3: 开始Distcc系统服务

```
# rc-update add distccd default
# /etc/init.d/distccd start
```

3.c. 缓冲编译结果

关于ccache

ccache是一个快速的编译器缓存。当您编译一个程序的时候，它会缓存中间的结果。这样，不论什么时候您重新编译同一个程序，编译所需要得时间将被大大缩短。对于普通的编译来说，这可以提高编译速度5到10倍。

如果您对ccache的工作机制有兴趣，请访问ccache主页.

安装ccache

要安装ccache，只需要运行emerge ccache:

代码 3.1: 安装ccache

```
# emerge ccache
```

激活Portage支持

打开/etc/make.conf并添加ccache到FEATURES变量。然后添加一个新的变量CCACHE_SIZE并设置它为"2G":

代码 3.2: 在/etc/make.conf中设定CCACHE_SIZE

```
CCACHE_SIZE="2G"
```

要检查ccache是否运行，只需让它提供给您它的统计数据。因为Portage使用一个不同的ccache主目录，您需要设定CCACHE_DIR变量：

代码 3.3: 查看ccache统计数据

```
# CCACHE_DIR="/var/tmp/ccache" ccache -s
```

/var/tmp/ccache是Portage的默认ccache主目录；为了修改这个设置，您可以设定/etc/make.conf中的CCACHE_DIR参数。

不过，如果您运行ccache，它使用的默认目录是${HOME}/.ccache。这就是为什么当您查询（Portage）ccache统计数据的时候您需要设定CCACHE_DIR参数的原因。

非Portage C编译中使用ccache

如果您需要在非Portage编译中使用ccache，添加/usr/lib/ccache/bin到您PATH参数里靠前的位置（在/usr/bin之前）。这一点可以通过编辑在您用户主目录中的.bash_profile文件来实现。使用.bash_profile是定义PATH参数的一个方式.

代码 3.4: 编辑.bash_profile

```
PATH="/usr/lib/ccache/bin:/opt/bin:${PATH}"
```

3.d. 二进制包支持

创建预编译包

Portage支持用预编译包安装。尽管Gentoo本身并不提供预编译包（除GRP快照之外），Portage依然能够处理预编译包。

如果某个包已经被安装在您的系统上，您可以用quickpkg来创建一个预编译包。也可以用带有--buildpkg或--buildpkgonly选项的emerge命令。

如果您希望Portage为您所安装的每一个单独的包创建预编译包，在FEATURES中添加buildpkg参数.

预编译包的更多扩展支持可以用catalyst得到。关于catalyst的更多信息请参阅Catalyst FAQ.

安装预编译包

尽管Gentoo并不提供，但是您可以自己建立一个“中心仓库”来存放预编译包。如果您希望使用这个仓库，您需要设定PORTAGE_BINHOST参数使Portage能够知道它。例如，如果预编译包在ftp://buildhost/gentoo上：

代码 4.1: 在/etc/make.conf中设定PORTAGE_BINHOST参数

```
PORTAGE_BINHOST="ftp://buildhost/gentoo"
```

当您需要安装预编译包的时候，在emerge命令后的--getbinpkg选项旁加入--usepkg选项。前者让emerge命令从预定的服务器上下载预编译包，后者让emerge首先试图安装预编译包，如果预编译包不存在，那么才下载并编译源码。

例如：用预编译包安装gnumeric

代码 4.2: 安装gnumeric的预编译包

```
# emerge --usepkg --getbinpkg gnumeric
```

关于emerge的预编译包的更多信息请参阅emerge手册页:

代码 4.3: 阅读emerge手册页

```
$ man emerge
```

3.e. 下载文件

并行下载

当您在编译一系列软件的时候，Portage能在编译某一个包的同时下载编译列表中后面的包的源码，这样就缩短了安装时间。如果您希望使用这种功能，添加“parallel-fetch”到您的FEATURES变量里。

Usersfetch

当您以root身份运行Portage，FEATURES="userfetch"可以让Portage在下载源码包的时候放弃root特权。这是一个小小的安全性的提高。

4. 初始化脚本

4.a. 运行级别

启动你的系统

当你启动系统时，会发现有很多的文字信息输出。如果注意观察的话，会发现每次启动时这些文字信息都是相同的。所有这些动作的顺序我们称之为*启动顺序*，而且它们基本上是被静态定义的。

首先，你的引导程序会把你在引导程序配置文件中定义的内核镜像加载到内存中，之后它就告诉CPU可以运行内核了。当内核被加载且运行后，内核会初始化所有内核专有的结构体和任务，并开启init进程。

然后，这个进程确保所有的文件系统（在/etc/fstab中定义的）都已被挂载且能使用。接着，该进程会执行位于/etc/init.d下的一些脚本，这些脚本会启动一些你需要的服务，以使你能获得一个成功启动的系统。

最终，当所有的脚本执行完毕，init将激活终端（大多情况下只是激活虚拟终端，可以使用Alt-F1、Alt-F2等来访问），并把一个叫agetty的特殊进程附于其上。这个进程会确保你可以通过运行login从这些终端登录到你的系统中。

初始化脚本

现在init不会随机的执行/etc/init.d下的脚本。甚至，它不会运行/etc/init.d下所有的脚本，它只会执行那些需要被执行的脚本。这个是由/etc/runlevels目录决定的。

首先，init会运行所有/etc/runlevels/boot目录下的符号链接所指向的/etc/init.d目录下的脚本。通常，它会按照字母顺序执行这些脚本，但是有些脚本中含有依赖关系，意味着系统要在执行另一个脚本之后才能运行此脚本。

当/etc/runlevels/boot目录所引用的脚本都执行完毕后，init将继续运行/etc/runlevels/default目录下的符号链接所指向的脚本。同样它们会按照字母顺序执行这些脚本，除非一个脚本有依赖关系，那样的话现有次序就会被改变以使启动顺序更加合理。

Init进程是如何工作的

当然init自己不会决定所有的启动顺序。它需要一个配置文件来指定它的工作流程。这个配置文件就是/etc/inittab。

如果你还记得我们刚刚描述的启动顺序，你会记得init首先做的是挂载所有的文件系统。这个功能其实是在/etc/inittab这个配置文件中定义好的。如下：

代码 1.1: /etc/inittab中系统初始化行

```
si::sysinit:/sbin/rc sysinit
```

这一行告诉init必须运行/sbin/rc sysinit来初始化系统。/sbin/rc脚本是负责系统初始化的，所以你可能会说init它本身并没做太多的事情——它只是把初始化系统任务交给了另一个进程。

接下来，init会执行所有在/etc/runlevels/boot目录下的具有符号链接的脚本。这是由下面这行定义的：

代码 1.2: 系统初始化，承上

```
rc::bootwait:/sbin/rc boot
```

同样，rc脚本将完成必要的工作。注意：给rc脚本的参数（*boot*）和要用的/etc/runlevels的子目录是一样的。

现在init进程将检查配置文件来判断应该运行哪个*运行级别（runlevel）*。它是通过读取/etc/inittab中的下面这行来决定的：

代码 1.3: 系统默认运行级别行

```
id:3:initdefault:
```

在这个例子（绝大部分的Gentoo用户将使用）里，*运行级别*id为3。根据这个信息，init进程会检查该执行什么来启动*运行级别3*：

代码 1.4: 运行级别的定义

```
l0:0:wait:/sbin/rc shutdown
l1:S1:wait:/sbin/rc single
l2:2:wait:/sbin/rc nonetwork
l3:3:wait:/sbin/rc default
l4:4:wait:/sbin/rc default
l5:5:wait:/sbin/rc default
l6:6:wait:/sbin/rc reboot
```

定义了级别3的行将再次调用rc脚本来启动服务（现在参数为*default*）。请再次注意：rc使用的参数和/etc/runlevels下的子目录是一样的。

当rc执行完毕后，init将会决定哪些虚拟终端需要被激活以及每个终端需要运行什么样的命令：

代码 1.5: 虚拟终端定义

```
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:12345:respawn:/sbin/agetty 38400 tty2 linux
c3:12345:respawn:/sbin/agetty 38400 tty3 linux
c4:12345:respawn:/sbin/agetty 38400 tty4 linux
```

```
c5:12345:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux
```

什么是运行级别？

你已经看到init使用一种数字的方式来决定需要激活的运行级别。*运行级别*表示你系统运行的状态，它包含了你进入或退出一个运行级别时需要执行的一组脚本（运行级别脚本或者*初始化脚本*）。

在Gentoo中定义了七种运行级别：三个内部运行级别和四个用户自定义运行级别。这些内部运行级别分别叫做*sysinit*、*shutdown*和*reboot*，它们所做的就如同像它们的名字那样：初始化系统、关闭系统和重启系统。

用户定义的运行级别都在/etc/runlevels目录下有同名的子目录：boot、default、nonetwork和single。运行级别boot会启动所有其他运行级别必须要使用到的系统服务。其余的三个运行级别的不同之处主要在于它们要启动的服务：default是用来日常工作用的；nonetwork是在无网络的情况下使用的；还有single是用户修复系统时用的。

使用初始化脚本

实际上rc进程调用的脚本都称为*初始化脚本*。每个在/etc/init.d下的脚本都可以在执行时带上以下参数，如：*start*、*stop*、*restart*、*pause*、*zap*、*status*、*ineed*、*iuse*、*needsme*、*usesme*或者*broken*。

要启动、停止或者重启一个服务（和所有依赖于它的服务），应该用参数start、stop和restart。示例如下：

代码 1.6: 启动Postfix服务

```
# /etc/init.d/postfix start
```

注意： 只有那些*需要*给定服务的服务会被暂停或重启。而其他依赖（*使用*但不需要）于它的服务则不会被暂停或重启。

如果你想要停止一个服务，但是不想影响依赖于它的服务，可以用pause这个参数：

代码 1.7: 停止Postfix服务但又保持相关依赖着的服务继续运行

```
# /etc/init.d/postfix pause
```

如果你要查看一个服务的运行状态（如：启动、停止、暂停……），你可以使用参数status：

代码 1.8: 查看postfix服务的运行状态

```
# /etc/init.d/postfix status
```

如果状态信息告诉你服务正在运行，但是你知道它实际上没有运行，这种情况下你可以使用参数zap将状态信息重设为“停止”：

代码 1.9: 重设postfix的状态信息

```
# /etc/init.d/postfix zap
```

有些时候我们也需要知道某个服务的依赖性，这时可以使用参数iuse或者ineed。使用ineed你可以查看这个服务正常工作真正必要的服务，而另一个方面iuse将会显示这个服务可能使用到的所有服务，但这并不一定是正常工作所必需的。

代码 1.10: 列出Postfix服务所依赖的所有必要服务列表

```
# /etc/init.d/postfix ineed
```

同样的，我们也可以知道哪些服务需要这个服务（needsme）或者哪些服务可以使用这个服务（usesme）：

代码 1.11: 列出哪些服务需要Postfix服务

```
# /etc/init.d/postfix needsme
```

最后，我们还可以知道这个服务需要但又缺少的依赖关系：

代码 1.12: 列出Postfix所缺少的依赖关系

```
# /etc/init.d/postfix broken
```

4.b. 使用rc-update

什么是rc-update？

Gentoo的初始化系统使用依赖关系树（dependency-tree）来决定什么服务会首先被启动。因为这是个很乏味的工作，我们不想让我们的用户去手动来完成它，所以我们创建了简化运行级别和初始化脚本的管理工具（*rc-update*）。

使用rc-update你可以从一个运行级别中添加或删除初始化脚本。rc-update工具会自动调用/sbin/depscan.sh脚本来重新创建依赖关系树。

添加和删除服务

在Gentoo的安装过程中你已经添加初始化脚本到“default”运行级别。那时你可能还不清楚“default”是干什么的，但是现在你应该知道了。rc-update脚本需要由第二个参数来决定其行为：*add*、*del*或者是*show*。

要添加或删除一个初始化脚本，只需要给rc-update *add*或者*del*参数，并随后跟上初始化脚本和运行级别。如下：

代码 2.1: 从default级别中删除Postfix服务

```
# rc-update del postfix default
```

rc-update -v show命令将会显示出所有已存在的初始化脚本，并列出它们在哪个运行级别中运行：

代码 2.2: 获得初始化脚本的信息

```
# rc-update -v show
```

你也可以运行rc-update show（没有-v参数）来只查看已经启用的初始化脚本和他们的运行级别。

4.c. 配置服务

我们为什么需要额外的配置？

初始化脚本有时候是很复杂的。因此我们不想让用户自己直接编辑初始化脚本，这样将更容易出错。然而能够配置这样的一个服务也挺重要。比如：你可能想给某个服务本身添加更多的选项。

把配置信息独立于脚本之外存放的另一个原因是，你不用担心升级初始化脚本会覆盖掉你之前所做的改动。

/etc/conf.d目录

Gentoo提供了一个简单的方法来配置这样的一个服务：每一个可以配置的初始化脚本在/etc/conf.d里有一个文件。比如：apache2的初始化脚本（叫做/etc/init.d/apache2）有一个配置文件叫/etc/conf.d/apache2，它包含了Apache 2服务器启动时你要给它的选项：

代码 3.1: 在/etc/conf.d/apache2中定义的变量

```
APACHE2_OPTS="-D PHP5"
```

这样的一个配置文件包含了变量，且只含有变量（就如同/etc/make.conf一样），可以使得配置服务非常简便。它还允许我们提供更多有关这个变量的信息（以注释形式）。

4.d. 撰写初始化脚本

需要这样吗？

当然不是，自己写一个初始化脚本通常情况下不是必需的，因为Gentoo给已提供的服务准备了一个立刻就能使用的初始化脚本。但是，你可能没有通过Portage来安装一个服务，在这种情况下你通常将需要创建一个初始化脚本。

如果服务软件提供者提供的初始化脚本不是明确地专为Gentoo而写的，请不要使用它，因为Gentoo的初始化脚本和其他发行版的初始化脚本是不兼容的！

布局

一个初始化脚本的基本布局如下所示：

代码 4.1: 初始化脚本的基本布局

```
#!/sbin/runscript

depend() {
   (依赖关系信息)
}

start() {
   (启动服务所必需的命令)
}

stop() {
   (停止服务所必需的命令)
}

restart() {
   (重启服务所必需的命令)
}
```

所有的初始化脚本都*需要*定义函数start()函数。其他所有的部分都是可选的。

依赖关系

在这里我们可以定义两种依赖关系：use和need。我们之前提到过，依赖关系中need比use要更加严格。紧跟着依赖关系类型之后你需要输入你所依赖的服务，或者*虚拟*依赖关系。

一个*虚拟*依赖关系是由一个服务提供的，但它不仅仅只有那个服务提供。我们自己定制的初始化脚本可以依赖于一个系统日志服务，但是可能我们有很多可用的系统日志程序（如：metalogd、syslog-ng、sysklogd……）。你不可能同时需要所有的系统日志程序（因为没有一个系统会同时安装和运行所有的系统日志程序），我们确保所有的这些服务都提供了一个虚拟依赖关系。

让我们来看看postfix服务的依赖关系信息：

代码 4.2: Postfix的依赖关系信息

```
depend() {
  need net
  use logger dns
  provide mta
}
```

就像你看到的，postfix服务信息如下：

- 需要（虚拟）依赖关系net（比如可由/etc/init.d/net.eth0提供）
- 使用到的（虚拟）依赖关系logger（比如可由/etc/init.d/syslog-ng提供）
- 使用到的（虚拟）依赖关系dns（比如可由/etc/init.d/named提供）
- 提供（虚拟）依赖关系mta（所有的mail服务器一样）

控制执行顺序

在一些情况下你可能并不需要某一个服务，但是，你可能需要这个服务在系统中其他某*可能*存在的服务之前（或之后）启动（注意这个条件——这不再是依赖关系了）*以及*运行在同一个运行级别（注意这个条件——只牵扯到同一运行级别的服务）。你可以使用before或者after设置用来提供这个信息。

下面我们来看一下Portmap服务的配置情况：

代码 4.3: Portmap服务的depend()函数

```
depend() {
  need net
  before inetd
  before xinetd
}
```

你也可以使用“*”来代替相同运行级别的所有服务，虽然这种做法不是很明智。

代码 4.4: 让初始化脚本在它所属的运行级别中第一个运行

```
depend() {
  before *
}
```

如果你的服务需要写入本地磁盘，则需要localmount服务。如果它在/var/run目录下放置文件，例如一个pid文件（pidfile），那么我们需要它在bootmisc之后运行。示例如下：

代码 4.5: depend()函数示例

```
depend() {
  need localmount
  after bootmisc
}
```

标准函数

紧随函数depend()，你还需要定义函数start()。它包含了初始化该服务所需要的所有命令。我们推荐使用函数ebegin和eend函数来通知用户要发生的事情：

代码 4.6: start()函数示例

```
start() {
  ebegin "Starting my_service"
  start-stop-daemon --start --exec /path/to/my_service \
    --pidfile /path/to/my_pidfile
  eend $?
}
```

--exec和--pidfile都应该用在start和stop函数中。如果这个服务并没有创建一个pidfile，那么如果可能的话就用--make-pidfile创建一个，不过你应该进行必要的测试以确保可行。否则，就不要使用pid文件（pidfile）。你也可以添加--quiet选项到start-stop-daemon中，但是通常我们不建议使用它，除非这个服务输出的信息极端冗长。使用--quiet选项的服务如果启动失败，那么可能会给调试带来困难。

注意： 我们要确保--exec选项实际上调用的是一个服务，而不仅仅是一个启动和退出服务的shell脚本——那是初始化脚本应该做的事。

如果你需要更多的start()函数示例，可以阅读/etc/init.d目录下的初始化脚本源代码。

你还可以定义其他的函数：stop()和restart()。但你不是一定要定义这些函数，因为如果我们使用start-stop-daemon时我们的初始化系统可以足够智能地填充这些函数。

虽然你不必创建一个stop函数，但下面还是给出了示例：

代码 4.7: stop()函数示例

```
stop() {
  ebegin "Stopping my_service"
  start-stop-daemon --stop --exec /path/to/my_service \
    --pidfile /path/to/my_pidfile
  eend $?
}
```

如果你的服务运行了一些其他语言的脚本（例如：bash、python、或者是perl），并且这个脚本之后改了名字（例如：foo.py改为foo），那么你需要给start-stop-daemon添加--name选项。同时，你必须要明确说明改过的名字。在这个例子中，一个服务启动了foo.py脚本，之后该脚本改名为foo：

代码 4.8: 一个启动foo脚本的服务

```
start() {
  ebegin "Starting my_script"
  start-stop-daemon --start --exec /path/to/my_script \
    --pidfile /path/to/my_pidfile --name foo
  eend $?
}
```

关于start-stop-daemon更多的信息你可以从man手册中得到：

代码 4.9: 获得start-stop-daemon的man帮助

```
$ $ man start-stop-daemon
```

Gentoo初始化脚本的语法是基于Bourne Again Shell(bash)的，所以你可以自由地在你的初始化脚本中使用bash兼容的结构。

添加自定义选项

如果你想要初始化脚本支持比我们当前所看到的更多的选项，你可以添加选项名称到opts变量中，然后创建一个与选项同名的函数。示例：支持一个名为restartdelay的选项：

代码 4.10: 支持restartdelay选项

```
opts="${opts} restartdelay"

restartdelay() {
  stop
  sleep 3    #在再次启动前等待3秒钟
  start
}
```

服务配置变量

要支持/etc/conf.d里的配置文件，你不需要作任何操作：当你的初始化脚本执行的时候，下面的文件将会被自动source（也就是说可以使用里面的变量）：

- /etc/conf.d/<你的初始化脚本>
- /etc/conf.d/basic
- /etc/rc.conf

而且，如果你的初始化脚本提供了一个虚拟依赖关系（如：net），那么和这个依赖关系相关的文件（如：/etc/conf.d/net）也会被source。

4.e. 改变运行级别的行为

谁可能从此获益？

许多笔记本用户会遇到这样的情形：在家时你需要启动net.eth0，而在路上你就不需要启动net.eth0（因为没有网络可用）。使用Gentoo，你可以根据你的意愿来改变运行级别的行为。

例如：你可以创建另一个可以启动的“default”运行级别，并分配其他的一些初始化脚本给它。然后你可以在启动时选择你想要运行哪个“default”运行级别。

使用softlevel

首先，给你的另一个“default”运行级别创建一个运行级别目录。作为一个例子我们创建offline运行级别：

代码 5.1: 创建一个运行级别目录

```
# mkdir /etc/runlevels/offline
```

下面我们添加必要的初始化脚本到新创建的运行级别中。示例：如果你需要一个与你当前的default运行级别一模一样的副本，但不包括net.eth0：

代码 5.2: 添加必要的初始化脚本

```
（复制所有在default运行级别中的服务到offline运行级别中）
# cd /etc/runlevels/default
# for service in *; do rc-update add $service offline; done
（删除在offline运行级别中不需要的服务）
# rc-update del net.eth0 offline
（显示在offline运行级别中活动的服务）
# rc-update show offline
（部分输出示例）
             acpid | offline
      domainname | offline
           local | offline
        net.eth0 |
```

虽然net.eth0已经从offline运行级别中被移除了，udev仍然尝试启动任何它检测到的设备并启动对应的服务。因此，你将需要把每一个你不想启动的网络服务（还有其他任何udev可能启动的设备的服务）加入/etc/conf.d/rc，如下所示。

代码 5.3: 在/etc/conf.d/rc里禁用设备启动的服务

```
RC_COLDPLUG="yes"
（下面指定你不想自动启动的服务名称）
RC_PLUG_SERVICES="!net.eth0"
```

注意： 欲知更多有关设备启动的服务的信息，请看/etc/conf.d/rc文件中的注释。

现在可以编辑你的引导程序的配置文件并且添加一个offline运行级别的条目。示例：在/boot/grub/grub.conf中：

代码 5.4: 增加一个offline运行级别的条目

```
title Gentoo Linux Offline Usage
  root (hd0,0)
  kernel (hd0,0)/kernel-2.4.25 root=/dev/hda3 softlevel=offline
```

至此，所有的设置都好了。如果你启动你的系统并在启动时选择新添的条目，将会使用运行级别offline而不是default。

使用bootlevel

使用bootlevel完全类似于softlevel。仅仅不同的是在这里你要定义另一个“boot”运行级别而不是另一个的“default”运行级别。

5. 环境变量

5.a. 环境变量？

什么是环境变量

环境变量是一个具有特定名字的对象，它包含了一个或者多个应用程序所将使用到的信息。许多用户（特别是那些刚接触Linux的新手）发现这些变量有些怪异或者难以控制。其实，这是个误会：通过使用环境变量，你可以很容易的修改一个牵涉到一个或多个应用程序的配置信息。

重要的例子

下表罗列了一些Linux系统使用的变量并说明了它们的用处。在表格后面将列举一些变量例值。

变量	说明
PATH	这个变量包含了一系列由冒号分隔开的目录，系统就从这些目录里寻找可执行文件。如果你输入的可执行文件（例如ls、rc-update或者emerge）不在这些目录中，系统就无法执行它（除非你输入这个命令的完整路径，如/bin/ls）。
ROOTPATH	这个变量的功能和PATH相同，但它只罗列出超级用户（root）键入命令时所需检查的目录。
LDPATH	这个变量包含了一系列用冒号隔开的目录，动态链接器将在这些目录里查找库文件。
MANPATH	这个变量包含了一系列用冒号隔开的目录，命令man会在这些目录里搜索man页面。
INFODIR	这个变量包含了一系列用冒号隔开的目录，命令info将在这些目录里搜索info页面。
PAGER	这个变量包含了浏览文件内容的程序的路径（例如less或者more）。
EDITOR	这个变量包含了修改文件内容的程序（文件编辑器）的路径（比如nano或者vi）。
KDEDIRS	这个变量包含了一系列用冒号隔开的目录，里面放的是KDE相关的资料。
CONFIG_PROTECT	这个变量包含了一系列用*空格*隔开的目录，它们在更新的时候会被Portage保护起来。

CONFIG_PROTECT_MASK	这个变量包含了一系列用*空格*隔开的目录，它们在更新的时候不会被Portage保护起来。

下面你可以找到所有这些变量定义的范例：

代码 1.1: 定义范例

```
PATH="/bin:/usr/bin:/usr/local/bin:/opt/bin:/usr/games/bin"
ROOTPATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin"
LDPATH="/lib:/usr/lib:/usr/local/lib:/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
MANPATH="/usr/share/man:/usr/local/share/man"
INFODIR="/usr/share/info:/usr/local/share/info"
PAGER="/usr/bin/less"
EDITOR="/usr/bin/vim"
KDEDIRS="/usr"
CONFIG_PROTECT="/usr/X11R6/lib/X11/xkb /opt/tomcat/conf \
                /usr/kde/3.1/share/config /usr/share/texmf/tex/generic/config/ \
                /usr/share/texmf/tex/platex/config/ /usr/share/config"
CONFIG_PROTECT_MASK="/etc/gconf"
```

5.b. 全局变量的定义

/etc/env.d目录

Gentoo采用了/etc/env.d目录来集中定义全局变量。在这个目录里，你会发现很多类似00basic、05gcc等等这样的文件，它们包含了文件名中提到的应用程序需要的变量。

举个例子，当你安装gcc时，一个名为05gcc的文件就会被ebuild所创建，里面包含了如下一些变量：

代码 2.1: /etc/env.d/05gcc

```
PATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
ROOTPATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
MANPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/man"
INFOPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/info"
CC="gcc"
CXX="g++"
LDPATH="/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
```

其他的发行版会让你到/etc/profile或者其他地方修改和添加这些变量的定义。而Gentoo为用户（还有为Portage）提供了更加便捷的方式来维护和管理环境变量，以后你不再需要把精力放在那些众多的包含环境变量的文件身上了。

比如，当你更新完gcc的时候，/etc/env.d/05gcc也会被同时更新，而不需要你手工来完成。

这不仅对Portage有益，作为用户，你也是受益者。有时候你需要设置某个系统范围的环境变量。我们拿http_proxy变量来做例子，为了避免把/etc/profile搞乱，你只要新建一个文件（/etc/env.d/99local）然后添加你的定义：

代码 2.2: /etc/env.d/99local

```
http_proxy="proxy.server.com:8080"
```

通过使用同一个文件来定义你所有的变量，你对如何定义自己的变量有了个大概的了解。

env-update脚本

/etc/env.d中的好几个文件都定义了PATH变量。这并没有错：当你运行env-update的时候，它会在更新环境变量之前把这些定义都追加到PATH里，因此对于软件包（或者用户）来说将会很容易地设置他们自己的环境变量，而不影响到现有变量的值。

env-update脚本会根据/etc/env.d里文件的字母顺序来附加变量的值。这些文件名必须要以两位数字开头。

代码 2.3: env-update所用的更新顺序

```
       00basic        99kde-env       99local
    +-------------+----------------+-------------+
PATH="/bin:/usr/bin:/usr/kde/3.2/bin:/usr/local/bin"
```

变量并不总是被串联起来，只有下列变量才会被串联：KDEDIRS、PATH、LDPATH、MANPATH、INFODIR、INFOPATH、ROOTPATH、CONFIG_PROTECT、CONFIG_PROTECT_MASK、PRELINK_PATH以及PRELINK_PATH_MASK。对于（/etc/env.d里的文件中按照字母顺序排列后）其他所有变量，最新定义的值才会被使用到。

当你运行env-update的时候，它会在文件/etc/profile.env里（会被/etc/profile使用）创建所有的环境变量。它也会从变量LDPATH中获取信息用来建立/etc/ld.so.conf。这些完成以后，它将运行ldconfig来重建动态链接器需要的文件/etc/ld.so.cache。

如果你想在运行env-update后立即看到效果，执行下面的命令来更新你的环境。自己安装过Gentoo的用户可能已经记住了这个安装指南中提到过的命令：

代码 2.4: 更新环境

```
# env-update && source /etc/profile
```

注意： 上面的命令只会更新你当前终端里的环境变量、*新*控制台以及它们的子程序。因此，假如你正在X11里工作，你要么在每一个你打开的终端里输入source /etc/profile，要么重新启动X，这样所有新的终端才能引用到新的变量。如果你使用了登录管理器，登陆成root然后输入/etc/init.d/xdm restart。如果不是这样，你需要注销然后重新登录回X这样才能产生使用新变量值的子程序。

重要： 你在定义其他变量时不能使用shell变量。这意味着这样的定义FOO="$BAR"（此处$BAR是另外一个变量）是不允许的。

5.c. 本地变量的定义

特定用户

你并不是一直都想定义全局变量。比如你想把/home/my_user/bin和当前目录（你当前所在的目录）添加到PATH变量中，但又不想让其他用户的PATH变量中也有这个。如果你想定义一个本地变量，可以使用~/.bashrc或者~/.bash_profile：

代码 3.1: 在~/.bashrc中扩展PATH做为本地使用

```
（冒号后面不跟任何目录就代表了当前目录）
PATH="${PATH}:/home/my_user/bin:"
```

当你重新登录的时候，你的PATH变量将被更新。

特定会话

有时候甚至需要更加严格的定义。你可能要使用一个你临时创建的目录里面的程序，而又不想输入它的路径或者为此短时间内内修改~/.bashrc。

在这种情况下，你只需要在当前会话中使用export来定义PATH变量。只要你不注销，PATH变量将保持这个临时的设置。

代码 3.2: 定义特定会话中的环境变量

```
# export PATH="${PATH}:/home/my_user/tmp/usr/bin"
```

C. 使用Portage

1. 文件和目录

1.a. Portage文件

配置指南

Portage附带了一个默认的配置文件/etc/make.globals。当你打开它时，你就会发现所有的Portage配置选项都是通过变量来控制的。Portage读取的是什么变量，这些变量分别又是什么意思，下面将详细解释。

因为许多配置命令在不同的架构下是不同的，Portage也有相应的各组默认配置文件，它们是你的profile的一部分。你的profile是/etc/make.profile这个链接文件指向的目录；Portage的配置选项是在你的profile以及所有被继承的profile中的make.defaults文件中设定的。我们稍后将详细解释profile及/etc/make.profile目录。

如果打算改变配置变量，*不要*变更/etc/make.globals或是make.defaults。而应该修改/etc/make.conf，它比前面的几个文件有更高的优先级。你会发现还有一个/usr/share/portage/config/make.conf.example文件。顾名思义，它仅仅是一个例子而已——Portage并不读取这个文件。

你也可以将一个Portage的配置变量定义为环境变量，但我们并不推荐这样做。

Profile特定的信息

我们已经见到过/etc/make.profile目录了。不过，这并不是一个真正的目录，而是一个指向profile的符号链接，默认情况下是一个位于/usr/portage/profiles里的目录，虽然你也可以在其他地方创建自己的profile并指向他们。这个符号链接指向的profile就是你的系统所使用的。

一个profile包含了Portage需要的与架构相关的信息，比如该profile对应的system包含的软件包的列表，以及对这个profile来说不能运行的（或者被屏蔽掉）的软件列表，等等。

用户特定的配置

当你需要变更Portage安装软件的行为时，你需要做的就是编辑/etc/portage 中的文件。我们*强烈建议*你使用/etc/portage中的文件而*不是*通过修改环境变量来变更这些行为！

在/etc/portage目录中，你可以创建下列文档：

- package.mask它列出了你永远不希望Portage安装的软件包。
- package.unmask它列出了本来Gentoo的开发者不建议安装的，但是你希望能安装的软件包。

- package.keywords它列出了还未被确认适合你的系统或架构，但是你希望能安装的软件包。
- package.use它列出了你希望某些特定软件包使用的而不是整个系统使用的USE标记。

这些并不需要一定是文件；它们也可以是有包含单个软件包信息文件的目录。更多关于/etc/portage目录的信息及你能创建的文件的完整列表可以在Portage的手册页中找到：

代码 1.1: 阅读Portage的手册页

```
$ man portage
```

改变Portage文件和目录的位置

先前提到的配置文件不能保存在其他地方——Portage总是会在这些特定的位置搜索配置文件。不过Portage还用了许多其他的位置来满足不同的目的：编译、保存源代码、保存portage树。

所有的这些目的都有众所周知的默认位置，不过你可以根据你自己的喜好通过/etc/make.conf来改变它们。本章中的其他部分将解释Portage使用哪些特定的位置存放它们以及怎样改变它们在你的文件系统中的存放地点。

这篇文档并不是一份全面的参考。如果你需要100%范围的说明，请参看Portage和make.conf的手册页：

代码 1.2: 阅读Portage和make.conf的手册页

```
$ man portage
$ man make.conf
```

1.b. 储存文件

Portage树

Portage树的默认位置是/usr/portage。这是PORTDIR变量定义的。当你把Portage树保存在其他位置（通过改变这个变量），不要忘了相应地改变符号链接/etc/make.profile指向的位置。

如果你改变了PORTDIR变量，你可能也需要改变下面几个变量，因为它们不会知道PORTDIR变量的改变。这是Portage处理这些变量的方式导致的：PKGDIR、DISTDIR、RPMDIR。

预编译二进制包

虽然Portage并不默认使用预编译的二进制包，但却对其有多方面的支持。当你要求Portage使用预编译的二进制包时，它就会在/usr/portage/packages中寻找它们。这个位置是通过PKGDIR变量定义的。

源代码

程序的源代码默认保存于/usr/portage/distfiles。这个位置是通过DISTDIR变量定义的。

Portage数据库

Portage将你系统的状态（装了哪些软件包，什么文件属于哪个软件包……）保存在/var/db/pkg。*不要*手动改变这些文件！它可能破坏Portage对你系统的了解。

Portage缓存

Portage缓存（包括修改时间、虚拟包、依赖关系树信息……）储存在/var/cache/edb。这个位置就是一个缓存：如果你没有正在运行portage相关的程序，你就可以清空它。

1.c. 编译软件

Portage的临时文件

Portage的临时文件默认保存在/var/tmp。这是通过PORTAGE_TMPDIR变量定义的。

如果你改变了PORTAGE_TMPDIR变量，你也需要改变下列的变量，因为它们不会知道PORTAGE_TMPDIR变量的改变。这是由于PORTAGE处理这些变量的方式导致的：BUILD_PREFIX。

编译目录

Portage为每一个它所安装的软件包在/var/tmp/portage里创建特定的编译目录。这一位置是通过BUILD_PREFIX变量定义的。

Live文件系统位置

默认情况下，Portage将所有的文件安装到当前文件系统（/）里。但是你可以通过改变环境变量ROOT来改变它。这在你想要创建一个新的编译镜像时是很有用的。

1.d. 日志特性

Ebuild日志

Portage能为每一个ebuild建立日志文件，但只有当PORT_LOGDIR变量设定的位置是portage可写的才行。默认情

况下，这个变量没有设定。如果你没有设定PORT_LOGDIR，你就不会收到当前日志系统报告的任何编译日志，然而你可能收到一些来自新的elog机制的日志。如果你定义了PORT_LOGDIR并且你也使用elog，你就将收到编译日志以及elog保存的任何日志，就像下文解释的一样。

Portage通过elog对日志记录提供精确的控制：

- PORTAGE_ELOG_CLASSES：这是你设置什么信息被记入日志的地方。你可以使用任何用空格分隔的info、warn、error、log和qa的组合。
  - info：记录下ebuild打印的“einfo”信息
  - warn：记录下ebuild打印的“ewarn”信息
  - error：记录下ebuild打印的“eerror”信息
  - log：记录下一些ebuild中的“elog”信息
  - qa：记录下ebuild打印的“QA Notice”信息
- PORTAGE_ELOG_SYSTEM：这是用来选择处理日志信息的模块的。如果留空，则日志记录就被取消。你可以使用任何用空格分隔开的save、custom、syslog、mail、save_summary和mail_summary的组合。你必须至少选择一个模块以使用elog。
  - save：表示将每一个软件包的日志保存在$PORT_LOGDIR/elog中，或者是在$PORT_LOGDIR 没有配置的情况下保存在/var/log/portage/elog 目录下面。
  - custom：将所有的信息传递给用户在$PORTAGE_ELOG_COMMAND中定义的命令，这将在随后讨论。
  - syslog：把所有的信息发送给已安装的系统日志软件。
  - mail：把所有的信息传递给用户在$PORTAGE_ELOG_MAILURI中定义的邮件服务器；这将在随后讨论。这一elog的邮件特性需要>=portage-2.1.1。
  - save_summary： 和save类似，不过它把所有的信息保存在$PORT_LOGDIR/elog/summary.log里，或者/var/log/portage/elog/summary.log里，如果$PORT_LOGDIR没有定义的话。
  - mail_summary： 和mail类似，不过它会在emerge结束时把所有的信息在一个邮件里发送出去。
- PORTAGE_ELOG_COMMAND：这个仅仅在custom模块被激活时使用。在这里你指定一个命令来处理日志信息。注意，你可以利用两个变量：${PACKAGE}是软件包的名字和版本，而${LOGFILE}是日志文件的绝对路径。这里是一个可能的用法：
  - PORTAGE_ELOG_COMMAND="/path/to/logger -p '\${PACKAGE}' -f '\${LOGFILE}'"
- PORTAGE_ELOG_MAILURI：这包含了mail模块的设定，如地址、用户、密码、邮件服务器、端口。默认配置是 “root@localhost localhost”。
- 这里是一个使用smtp服务器的例子，基于用户名和密码认证并使用特定端口(默认端口是25)：
  - PORTAGE_ELOG_MAILURI="user@some.domain username:password@smtp.some.domain:995"
- PORTAGE_ELOG_MAILFROM：允许你配置日志邮件的“from”地址；如果没有配置的话，默认是“portage”。
- PORTAGE_ELOG_MAILSUBJECT：允许你为日志邮件生成一个主题行。注意你可以使用两个变量：${PACKAGE}将会显示软件包的名子和版本，而${HOST}则是运行Portage的主机的FQDN。
- 这里是一个可能的应用：
  - PORTAGE_ELOG_MAILSUBJECT="package \${PACKAGE} was merged on \${HOST} with some messages"

重要： 如果你使用Portage-2.0.*中的enotice，你必须完全移除enotice，因为它和elog不兼容。

2. 通过变量来配置

2.a. Portage的配置

如前面所述，你可以在/etc/make.conf中定义许多变量来配置Portage。请参考make.conf的手册页以获得更详细和完整的信息：

代码 1.1：阅读make.conf的手册页

```
$ man make.conf
```

2.b. 构建相关的选项

配置和编译器选项

当Portage构建应用程序时，他会将下面变量的内容传给编译器和configure脚本：

- CFLAGS和CXXFLAGS定义了编译器编译C和C++程序所使用的参数。
- CHOST为程序的configure脚本定义了build和host信息。
- MAKEOPTS参数被传递给make命令，通常用来设定编译时并行处理线程的数量。更多关于make的参数可在make的手册页中找到。

USE变量也会在配置和编译的过程中被用到，不过这在前面的章节中已经详细解释过了。

安装相关选项

当Portage在安装了一个某软件的新版本后，会把属于旧版本的已经过时的文件从你的系统中删除。在此之前

Portage会给用户5秒钟时间撤销此操作，这5秒钟由CLEAN_DELAY变量定义。

你可以通过设定EMERGE_DEFAULT_OPTS变量以使emerge每次运行时都使用某些特定的选项。一些常用的选项是--ask，--verbose，--tree，等等。

2.c. 配置文件的保护

Portage的保护位置

Portage一般会用新文件覆盖掉旧文件，除非这些旧文件位于*受保护的*位置里。这些受保护的位置由CONFIG_PROTECT变量设定，通常是配置文件的位置。在设置时，多个目录间用空格分开。

当受保护的位置里的文件有新版本时，新文件被保存前将会被更名，同时用户会被告知此新配置文件的存在。

你可以利用emerge --info来获得目前CONFIG_PROTECT的设置：

代码 3.1: 获得CONFIG_PROTECT的设定

```
$ emerge --info | grep 'CONFIG_PROTECT='
```

更多关于Portage配置文件保护的信息可以查看emerge手册页的CONFIGURATION FILES部分：

代码 3.2: 更多关于Portage配置文件保护的信息

```
$ man emerge
```

解除对某些目录的保护

你可以通过变量CONFIG_PROTECT_MASK来解除对受保护目录下某些子目录的保护。

2.d. 下载的选项

服务器的地址

当你的系统上没有所需要的信息和数据时，Portage就会从网络上获取数据。含有这些信息和数据的服务器的位置定义在以下变量中：

- GENTOO_MIRRORS定义了一些提供源代码（distfiles）的服务器地址。
- PORTAGE_BINHOST定义了一个特定的服务器地址，此服务器可为你的系统提供预先编译好的软件包。

第3个变量设定rsync服务器的地址，此服务器用来更新你的Portage树：

- SYNC定义了一个特定的服务器，Portage用此服务器获取Portage树。

GENTOO_MIRRORS和SYNC变量可以由程序mirrorselect自动设定。首先你要emerge mirrorselect。更多信息参考mirrorselect的在线帮助：

代码 4.1: 更多关于mirrorselect的信息

```
# mirrorselect --help
```

如果你需要使用代理服务器，你可以用http_proxy、ftp_proxy和RSYNC_PROXY变量来设定它们。

获取源代码的命令

当Portage需要获取源代码时，默认使用wget来下载。你可以通过定义FETCHCOMMAND变量来改变它。

Portage支持断点续传。默认是使用wget来实现，但你可以通过定义RESUMECOMMAND来变量来改变它。

确保你的FETCHCOMMAND和RESUMECOMMAND把源代码存放到正确的位置。在这些变量中你应该用\${URI}和\${DISTDIR}分别指定源代码和distfiles的位置。

你可以对不同的传输协议使用不同的设置，相关的参数有FETCHCOMMAND_HTTP、FETCHCOMMAND_FTP、RESUMECOMMAND_HTTP和RESUMECOMMAND_FTP等等。

Rsync的设置

你不能改变Portage用来更新Portage树所使用的rsync命令，但你可以设定一些与rsync命令相关的变量：

- PORTAGE_RSYNC_OPTS定义了一些在sync过程中使用的变量，变量之间用空格分开。不要做任何改变除非你*十分清楚*会有什么后果。注意某些必要的选项总会起作用，即使PORTAGE_RSYNC_OPTS的设置是空的。
- PORTAGE_RSYNC_EXTRA_OPTS可以用于在同步过程中设置附加选项。选项之间用空格分开。
  - --timeout=<number>： 这里设定rsync连接超时的秒数。此变量默认为180秒。如果用户是使用拨号上网或者一台处理速度慢的电脑，这个值需要设到300或者更高。
  - --exclude-from=/etc/portage/rsync_excludes： 这个参数指定一个文件，内容是rsync在更新过程中需要忽略的软件包和/或者类别的列表。这个例子指定的是/etc/portage/rsync_excludes。请参考使用Portage树的子集以了解此文件的格式。
  - --quiet： 减少屏幕输出

- --verbose： 在屏幕上输出完整的文件列表
- --progress： 为每个文件显示一个进度条

- PORTAGE_RSYNC_RETRIES定义了rsync程序试图连接到由SYNC变量所指定的服务器的次数，默认为3。

更多有关这些参数的信息请阅读man rsync.

2.e. Gentoo的设置

分支的选则

你可以用ACCEPT_KEYWORDS变量改变默认的系统分支。此变量的默认设置为你的电脑体系结构的稳定分支。更多关于Gentoo分支的信息可以在下一章节找到。

Portage的特性

你可以通过FEATURES变量激活Portage的某个特性。这些已在前面的章节中讨论过，如Portage的特性.

2.f. Portage的行为

资源管理

通过PORTAGE_NICENESS变量，你可以增加或减少Portage运行时的nice值。PORTAGE_NICENESS变量的值会被*加到*目前的nice值上。

更多关于nice值的信息，请参考nice的手册页：

代码 6.1: 更多关于nice的信息

```
$ man nice
```

输出行为

NOCLOR变量的默认值为“false”，如果你想禁用Portage的彩色输出，请把这个变量设为“true”。

3. 使用多个软件分支

3.a. 使用一个分支

稳定的分支

Gentoo提供了一个ACCEPT_KEYWORDS变量来定义您系统所使用的软件分支。默认情况下，系统会选择您的体系结构的稳定软件分支。例如x86。

我们推荐您只使用默认的稳定软件分支。不过，如果您不是那么注重稳定性，并且愿意向http://bugs.gentoo.org提交一些bug报告来帮助和完善Gentoo，请继续阅读下面的内容。

测试的分支

如果您想用最新版本的软件，您可以考虑转向使用测试分支。要让Portage转而使用测试分支的软件，您只需在您的体系结构名称前加上一个~符号。

顾名思义，“测试分支”就是带有测试性质的。如果一个包正处于测试中，这代表软件的开发人员认为它虽然已经具有了相当的功能但还没有经过完全的测试。使用这样的软件，您当然可能会第一个发现它的bug，并可以提交一个bug报告来通知相关的开发者。

要小心的是，您可能会遇到不稳定性、不完美的软件包处理（例如错误或者缺失的依赖关系）、过于频繁的更新（导致大量的编译）和损坏的包等问题。如果您还不是很清楚Gentoo的工作方式以及如何去解决这些问题，我们推荐您还是使用稳定且测试过分支。

例如，要选择针对x86体系结构的测试分支，请修改/etc/make.conf文件并设定如下内容：

代码 1.1: 设定ACCEPT_KEYWORDS变量

```
ACCEPT_KEYWORDS="~x86"
```

之后如果您更新系统，您将会发现有*大量*的包需要更新。要提醒您注意的是：您使用测试分支更新系统后，再想转回使用官方的稳定分支将不是一件容易的事情（当然您使用系统的备份的情况除外）。

3.b. 混合使用稳定和测试分支

package.keywords文件的位置

您可以让Portage使用某些软件的测试分支中的版本，对于系统的其他软件则使用稳定分支。要实现这样的目的，您需要在/etc/portage/package.keywords文件里加入那些软件包的名字及其所属分类的名称。您也可以建立一个同名文件夹，并在里面建立的文件里加入上述内容。例如，要使用gnumeric属于测试分支中的版本：

代码 2.1: /etc/portage/package.keywords中针对gnumeric设置，一行完整内容

```
app-office/gnumeric ~x86
```

测试特定版本

如果您希望Portage使用测试分支中某软件的特定版本，但后续版本不再这么做，你可以在package.keywords里加入相应的版本号来实现这个目的。在此情况下您*必须*使用 = 运算符。您也可以通过使用<=，<，>或>=运算符来指定一个要使用的版本范围。

任何情况下，如果您添加了版本号，您*必须*使用一个运算符；如果您忽略了版本号，您就*不能*使用运算符。

在如下的例子中，我们要求Portage接受版本号为1.2.13的gnumeric：

代码 2.2: 允许gnumeric的测试版本

```
=app-office/gnumeric-1.2.13 ~x86
```

3.c. 使用被屏蔽的包

package.unmask文件的位置

Gentoo的开发者们**不**支持您使用这个文件。如果您要使用它请千万小心。开发者们将不会回应有关于package.unmask和 / 或package.mask的支持请求。您已经被提醒过了。

当一个包被Gentoo的开发者们屏蔽，但你不考虑package.mask文件（默认保存于/usr/portage/profiles目录下）里所陈述的原因，仍然想使用它的话，请在/etc/portage/package.unmask文件（如果是一个文件夹，就在此文件夹下的文件中）中加入与package.mask里那行*一模一样*的内容。

比如说，软件包=net-mail/hotwayd-0.8被屏蔽了，但是您想解除这个屏蔽，需要做的只是在package.unmask中加入相同的一行内容就可以了。

代码 3.1: /etc/portage/package.unmask

```
=net-mail/hotwayd-0.8
```

package.mask文件的位置

当您希望Portage忽略一个特定的软件包或者一个软件包的特定版本，您可以在/etc/portage/package.mask文件（或者此目录下的一个文件）中加入一行适当的内容来屏蔽它。

例如，当您不希望Portage安装比gentoo-sources-2.6.8.1更新版本的内核时，您只需要在package.mask文件里加入下面一行内容：

代码 3.2: /etc/portage/package.mask的例子

```
>sys-kernel/gentoo-sources-2.6.8.1
```

4. Portage附加工具

4.a. dispatch-conf

dispatch-conf是一个帮助合并._cfg0000_<名称>文件的工具。._cfg0000_<名称>文件是由Portage在它要覆盖被CONFIG_PROTECT变量所保护的某个目录里的文件时建立的。

使用dispatch-conf能够在合并配置文件并升级更新的同时保持所有更新记录。dispatch-conf以RCS版本管理系统或是补丁的方式来保存配置文件间的差别。这意味着如果你在升级配置文件犯下错误时，你可以随时退回到你的配置文件的之前版本。

使用dispatch-conf，你可以保持配置文件原来的样子，或者使用新的配置文件，你还可以编辑当前文件或交互式地合并更新。除此之外，dispatch-conf还有一些很棒的特性：

- 可以自动合并仅有注释变更的文件；
- 可自动合并仅有空白符数量的不同的文件；

确定你先编辑了/etc/dispatch-conf.conf并创建了archive-dir变量设定的目录。

代码 1.1: 运行dispatch-conf

```
# dispatch-conf
```

当运行dispatch-conf的时候，程序会带你把每个改变了的配置文件挨个过一边。按u来用新配置文件更新（替换）现在的配置文件，然后继续处理下一个。按z来删除新配置文件，然后继续处理下一个。当处理完所有的配置文件之后，dispatch-conf就会退出。你也可以随时按q来退出。

更多信息，请查阅dispatch-conf手册页。它会告诉你交互式的合并新旧配置文件，编辑新配置文件，检查两个文件间的差异等等。

代码 1.2: 阅读dispatch-conf手册页

```
$ man dispatch-conf
```

4.b. etc-update

你也可以使用etc-update来合并配置文件。它不像dispatch-conf那样简单易用，功能也少，但是它也能提供交互式合并功能并且能自动合并一些简单的改变。

不过，和dispatch-conf不同的是，etc-update *不*保留你的配置文件的旧版本。一旦你更新了文件，旧版本就永远丢失了。所以要非常小心，因为使用etc-update与使用dispatch-conf相比 *明显*的不安全。

代码 2.1: 运行etc-update

```
# etc-update
```

在整合简单直观的更动后，系统会提示你一个需要更新的受保护的文件列表。在最底下会提示你可选的操作选项：

代码 2.2: etc-update选项

```
Please select a file to edit by entering the corresponding number.
              (-1 to exit) (-3 to auto merge all remaining files)
                           (-5 to auto-merge AND not use 'mv -i'):
```

如果你输入-1，etc-update将直接退出且不执行任何变更。如果你输入-3或-5，所有列出的配置文件将被更新的版本覆盖。因此先选出无需自动升级的配置文件非常重要，而具体步骤也很简单，只需要输入在该配置文件左边显示的数字就可以了。

我们选择配置文件/etc/pear.conf作为范例：

代码 2.3: 更新指定的配置文件

```
Beginning of differences between /etc/pear.conf and /etc/._cfg0000_pear.conf
[...]
End of differences between /etc/pear.conf and /etc/._cfg0000_pear.conf
1) Replace original with update
2) Delete update, keeping original as is
3) Interactively merge original with update
4) Show differences again
```

现在你可以看到这两个文件之间的差别。如果你认为升级的配置文件可以正确无误的投入使用，输入1。如果你认为升级的配置文件是不必要的，或者它也没有提供任何新的或有用的信息，输入2。如果你想交互地升级你当前的配置文件，输入3。

这里我们就不再赘述交互性整合的详细过程。出于完整性的考虑，我们将列出在整合两个文件时可以用到的所有的命令。你将看到来自新旧文件的两行内容和一个提示符，在提示符这里你可以输入以下命令：

代码 2.4: 用交互方式配置时可以使用的命令

```
ed:     编辑并使用两种版本，每一个版本都加上一个不同的标题头。
eb:     编辑并使用两个版本。
el:     编辑并使用左边的版本。
er:     编辑并使用右边的版本。
e:      编辑一个新的版本。
l:      使用左边的版本。
r:      使用右边的版本。
s:      包含相同的行。
v:      包含相同的行，并列出来源。
q:      退出。
```

当你完成重要的配置文件的更新后，余下的其它配置文件你就可以采用自动更新的方法了。当无法再找到任何可更新的配置文件时etc-update将退出。

4.c. quickpkg

利用quickpkg可以对系统中已安装的包进行打包归档。这些归档文件可以作为预编译包使用。运行quickpkg非常简单：只要加上你想要制作的软件包的名字就可以了。

例如，要打包curl,arts,procps

代码 3.1: quickpkg的使用范例

```
# quickpkg curl arts procps
```

预编译包会保存在$PKGDIR/All（默认为/usr/portage/packages/All)。指向这些包的符号链接保存在$PKGDIR/<category>中。

5. 改造Portage树

5.a. 使用Portage树的一个子集

排除软件包/类别

你可以有选择地更新特定的类别/软件包并且忽略其他类别/软件包。我们通过让emerge --sync在执行rsync的时

候排除个别类别/软件包来实现这个功能。

你需要在/etc/make.conf中赋予--exclude-from变量一个文件名，该文件应包含你想要排除的软件包。

代码 1.1: 在/etc/make.conf中指定包含排除软件包条目的文件

```
PORTAGE_RSYNC_EXTRA_OPTS="--exclude-from=/etc/portage/rsync_excludes"
```

代码 1.2: 在/etc/portage/rsync_excludes中设定排除所有游戏

```
games-*/*
```

然而要注意的是这可能导致依赖性问题，因为新的、允许安装的软件包可能依赖于那些新的、但被排除的软件包。

5.b. 添加非官方Ebuild

定义一个Portage Overlay目录

可以让你的Portage使用官方Portage树里面没有的ebuild。创建一个新的目录（比如/usr/local/portage）用以存放第三方ebuild。请在新目录中使用跟官方Portage树一样的目录结构！

然后在/etc/make.conf中定义PORTDIR_OVERLAY变量，使它指向刚才创建的目录。现在，每当你用到Portage的时候，这些ebuilds也会被同时计算在内，并且当你下次运行emerge --sync时，不会删除/覆盖这些ebuild。

使用多个Overlay仓库

对于开发多个Overlay的许多高级用户而言，他们会在更新Portage树之前测试软件包或是想使用来源广泛的非官方ebuild。app-portage/gentoolkit-dev软件包提供了gensync工具，帮助你的overlay仓库时时都处于最新的状态。

可以用gensync一次更新所有的仓库，也可以选择更新其中的几个。每个仓库会有一个.syncsource文件于/etc/gensync/配置目录下，其内容包含仓库的存放位置、名字、ID等。

假设你有两个额外的仓库，分别名为java（存放开发中的java ebuild）和entapps（存放在家中为你给公司开发的应用程序）。那么可以通过以下命令来更新这些仓库：

代码 2.1: 使用gensync更新仓库

```
# gensync java entapps
```

5.c. 非Portage维护的软件

Portage环境下使用自己维护的软件

尽管Portage已经提供了某些软件，但是某些情况下你还是想自己配置、安装和维护这些软件，而不是希望Portage为你自动完成这些过程。典型的例子是内核源码和nvidia驱动程序。你可以配置一下Portage让它知道某个软件包已经通过手动安装到系统中了。这个过程叫做*注入*”，Portage是通过/etc/portage/profile/package.provided文件来实现这个功能的。

举个例子，如果你想告诉Portage已经手动安装了gentoo-sources-2.6.11.6，那么把下面一行添加到/etc/portage/profile/package.provided文件里：

代码 3.1: package.provided范例

```
sys-kernel/gentoo-sources-2.6.11.6
```

D. Gentoo网络配置

1. 新手上路

1.a. 新手上路

注意： 本文假定你已经配置好了你的内核，包括你的硬件的模块，并且你知道你的硬件（译注：这里指网卡）的接口名。我们同时假定你已经设置了eth0，当然它也可以是eth1、wlan0等等。

注意： 本文档要求你正在运行的是baselayout-1.11.11或更高版本。

要开始配置你的网卡，你首先需要告诉Gentoo RC系统你的网卡。这可以通过在/etc/init.d目录里建立一个指向net.lo的名叫net.eth0的符号链接来实现。

代码 1.1: 建立符号连接net.eth0指向net.lo

```
# cd /etc/init.d
# ln -s net.lo net.eth0
```

现在Gentoo的RC系统知道了这个接口。它还需要知道怎么来配置这个接口。所有的网络接口都在/etc/conf.d/net文件里设置。下面是一个设置DHCP和静态地址的简单配置：

代码 1.2: /etc/conf.d/net文件的一个示例

```
# DHCP
config_eth0=( "dhcp" )

# 使用CIDR形式表示的静态IP
config_eth0=( "192.168.0.7/24" )
routes_eth0=( "default via 192.168.0.1" )

# 使用netmask形式表示的静态IP
config_eth0=( "192.168.0.7 netmask 255.255.255.0" )
routes_eth0=( "default via 192.168.0.1" )
```

注意： 如果你没有指定，DHCP是默认选项。

注意： CIDR代表无级别Internet域路由（Classless InterDomain Routing）。一开始，IPv4地址被归类为A，B或者C类。最初的分类系统没有考虑到Internet如此流行，会有耗尽IP的危险。CIDR允许一个IP地址可以表达多个IP地址。除了以一个斜线跟着一个数字结束外，CIDR IP地址跟普通IP地址是一样的；例如：192.168.0.0/16。CIDR是RFC 1519定义的。

现在我们已经配置好了接口，可以用下面的命令启动、停止它

代码 1.3: 启动和停止网络脚本

```
# /etc/init.d/net.eth0 start
# /etc/init.d/net.eth0 stop
```

重要： 我们建议你在/etc/conf.d/rc文件里设置RC_VERBOSE="yes"。这样你可以在网络出现故障的时候，得到关于故障的更多信息。

现在已经成功地启动并停止了你的网络接口，你可能希望在Gentoo启动的时候启动它。这样做就可以了：（最后那个“rc”命令通知Gentoo启动当前runlevel中还没有被启动的脚本）

代码 1.4: 配置网络接口以便在启动的时候装载它

```
# rc-update add net.eth0 default
# rc
```

2. 高级配置

2.a. 高级配置

config_eth0变量是一个网络接口配置的核心。它包含了一组用来配置接口的高级指令列表（我们用eth0作为示例）。指令列表中的每一条命令都是顺序执行的。该列表中只要至少有一条命令正常工作，则这个网络接口就被认为是可以使用的。

这里列出了一些可用的指令。

命令	描述
null	不做任何事
noop	如果网络接口已经开启并且也绑定好地址的话，则中止配置过程，配置成功。
一个IPv4或是IPv6的地址	绑定地址到接口上
dhcp、adsl或apipa（或者一个第三方模块提供的自定义命令）	运行提供这些命令的模块，例如dhcp这个命令会运行一个提供DHCP功能的模块，它可能是dhcpcd、dhclient或者pump中的一个。

如果一条命令执行失败，你可以指定一条fallback指令。这条fallback指令必须完全符合配置的结构。

你可以连在一起使用这些指令，下面是一些真实应用的例子。

代码 1.1: 配置示例

```
# 加上三个IPv4的地址
config_eth0=(
  "192.168.0.2/24"
  "192.168.0.3/24"
  "192.168.0.4/24"
)

# 加上一个IPv4的地址和两个IPv6的地址
config_eth0=(
  "192.168.0.2/24"
  "4321:0:1:2:3:4:567:89ab"
  "4321:0:1:2:3:4:567:89ac"
)

# 保持使用内核分配的地址，除非网络接口没有启动。如果是这样则改用DHCP获
#取。如果DHCP失败的话则由APIPA指定一个网络地址。
config_eth0=(
  "noop"
  "dhcp"
```

```
)
fallback_eth0=(
  "null"
  "apipa"
)
```

注意： 当你使用的是ifconfig模块并且绑定了一个以上的网络地址，系统会为每一个网络地址建立一个别名。在上面的两个例子中你将会得到这几个网络接口：eth0、eth0:1和eth0:2。由于内核和其他程序会把eth0:1和eth0:2看作eth0，所以你将无法对这些接口做什么特殊的事情。

重要： fallback中的顺序相当重要！如果我们没有指定null选项，那么apipa只会在noop命令失败后才会执行。

注意： APIPA和DHCP稍后会讨论。

2.b. 网络依赖性

/etc/init.d中的初始化脚本可能依赖于某个特定的网络接口或只是net服务。可以通过定义在/etc/conf.d/rc中的RC_NET_STRICT_CHKING这个变量，来使net代表不同的意思。

值	描述
none	**系统始终认为net服务已经启动了**
no	**这是说除了net.lo之外，至少还需要再启动一个net.*服务。同时拥有WIFI网络和静态NIC网络的笔记本用户可使用这个值。对他们来说某一时间只要有一个网络接口启动，那么net服务就应被视为启动了。**
lo	**这和no选项相同，不过就是把net.lo也考虑在内了。它对那些不怎么在意在系统启动时哪个网络接口会被启动的用户很有用。**
yes	**所有的网络接口都启动后，net服务才算启动。**

可是需要依赖net.eth0和net.eth1的net.br0应该怎么处理呢？net.eth1可能是一个无线网络或是一个PPP设备，且需要在桥接前先设定好。这不能在/etc/init.d/net.br0中定义，因为它只是一个到net.lo的符号连接。

答案是在/etc/conf.d/net里定义一个自己的depend()函数。

代码 2.1: /etc/conf.d/net中net.br0的依赖关系

```
# 你可以使用目前脚本里能用的任何依赖关系（use、after、before）
depend_br0() {
  need net.eth0 net.eth1
}
```

要获得更多关于依赖性的讨论。请参考Gentoo手册中的这一节撰写初始化脚本

2.c. 变量名称和值

变量的名称是动态的，它们通常的结构是variable_${interface|mac|essid|apmac}。例如dhcpcd_eth0变量代表eth0的dhcpcd命令参数。而dhcpcd_essid变量代表连接到ESSID为"essid"的AP的网络接口的dhcpcd命令参数。

不过，没有什么强制性的和快速的标准规定了网络接口名称必须是ethx。事实上，许多无线网络接口除了使用ethx之外，也会使用类似于wlanx、rax之类的名称。而且一些用户自定义的网络接口（比如网桥接口）可以被命名为任何名字（比如foo）。更有趣的是，无线网络接入点可以在ESSID中使用非英文字符或数字字符——这一点很重要，因为你可以为每一个ESSID设定网络参数。

不幸的是Gentoo在网络设定方面使用的是bash的变量名——而bash不能使用数字、英文字母之外的任何字符。为了突破这样的一个限制，我们把所有除数字、英文字母之外的字符用_字符表示。

bash的另一个问题是变量的内容——一些字符需要转义。这可以通过在需要转义的字符前面加\来解决。下面列出的是需要转义的字符列表："、'和\。

在这个例子中，我们使用无线网络ESSID，因为它可以包含的字符范围最广。我们使用My "\ NET这个ESSID：

代码 3.1: 变量名示例

```
（这样确实可以传递域名，但是域名的值是非法的）
dns_domain_My____NET="My \"\\ NET"

（上面的例子意思是当一块无线网卡连接到ESSID为My "\ NET的无线接入点时
把dns域名设置为My "\NET）
```

3. 模块化网络

3.a. 网络模块

我们现在支持模块化的网络脚本，这意味着我们可以很方便地增添新的网络接口和配置模块，同时和原来已存在的模块或是接口保持良好的兼容性。

只要模块需要的软件包已安装，它就会被默认载入。如果你指定一个模块而它需要的软件包未被安装，则你就会收到一个错误信息通知你需要安装哪一个软件包。一般来说，只有当你安装了提供相同服务的两个及以上的软件包

时，你才需要使用模块设定来选择你想使用其中的哪一个。

注意： 除非有特别说明，这里所讨论的全部设定都保存在/etc/conf.d/net中。

代码 1.1: 模块选择

```
# 选择iproute2而不用ifconfig
modules=( "iproute2" )

# 你也可以为一个网络接口指定其他的模块
# 这个例子中我们希望使用pump来代替dhcpcd
modules_eth0=( "pump" )

# 你也可以指定不使用哪一个模块——例如你一方面想使用supplicant或是
# linux-wlan-ng来控制无线网络配置，一方面又想配置每一个ESSID相关联的网络设定。
modules=( "!iwconfig" )
```

3.b. 网络接口处理程序

目前我们提供了两个网络接口处理程序：ifconfig和iproute2。你可以使用i这两个程序中的一个进行任意种类的网络配置。

ifconfig是目前Gentoo默认使用的网络接口处理程序，而且它也包含在系统的profile之中。iproute2是另一个灵活且强大的软件包，但是默认的系统配置中并不包含它。

代码 2.1: 安装iproute2

```
# emerge sys-apps/iproute2

# 在iproute2和ifconfig都已安装的情况下，选用iproute2
modules=( "iproute2" )
```

由于ifconfig和iproute2的功能很类似，我们允许使用它们的基本配置功能以使二者可以通用。例如说下面的代码片段对于它们之中的任何一个来说都是起效的。

代码 2.2: ifconfig和iproute2示例

```
config_eth0=( "192.168.0.2/24" )
config_eth0=( "192.168.0.2 netmask 255.255.255.0" )

# 我们也可以指定广播地址
config_eth0=( "192.168.0.2/24 brd 192.168.0.255" )
config_eth0=( "192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255" )
```

3.c. DHCP

DHCP是一种通过DHCP服务器来获得相关的网络信息（比如IP地址、DNS服务器、网关等）的方式。这意味着如果在网络上存在着一个DHCP服务器的话，你只要让每一个客户端使用DHCP，这些客户端就会自动地配置好相关的网络设置。当然，如果在使用DHCP之前还需要配置无线网络或是PPP等其他设定的话，你就必须先配置好这些才行。

dhclient、dhcpcd或者pump都可以提供DHCP功能。每一种DHCP模块都有各自的优缺点——下面是一个快速的检阅。

DHCP模块	软件包	优点	缺点
dhclient	net-misc/dhcp	由ISC（也是BIND DNS软件的开发人员）制作；高度可配置性。	配置起来过于复杂；软件十分臃肿；无法通过DHCP获得NTP服务器；默认不发送主机名。
dhcpcd	net-misc/dhcpcd	Gentoo一直以来的预设软件；不依赖其他外部软件；由Gentoo开发的。	有时很慢；当租约为无穷大时不会变成守护进程。
pump	net-misc/pump	轻量级的；不依赖其他外部软件	不再被上游维护，不可靠，尤其在使用modem的时候；不能通过DHCP获得NIS服务器。

当你在系统上安装有不止一种DHCP客户端，你需要指定使用哪一个——否则如果系统装有dhcpcd，我们将默认使用它。

你可以使用module_eth0="..."（*把module改成你所使用DHCP模块的名称——例如：dhcpcd_eth0*）为DHCP模块指定选项。

我们尝试使DHCP的配置更加灵活——所以我们通过dhcp_eth0变量来支持下面这些命令，默认不使用它们中的任何一个：

- release - 释放IP地址以便它可以被重新使用
- nodns - 不覆盖/etc/resolv.conf
- nontp - 不覆盖/etc/ntp.conf
- nonis - 不覆盖/etc/yp.conf

代码 3.1: 在/etc/conf.d/net中的DHCP配置范例

```
# 只在你安装了一个以上的DHCP模块后才需要
modules=( "dhcpcd" )

config_eth0=( "dhcp" )
dhcpcd_eth0="-t 10" # 10秒后超时
dhcp_eth0="release nodns nontp nonis" # 只获取IP地址
```

注意： 由于dhcpcd和pump默认会把主机名发送给DHCP服务器，所以你就可以不用去指定它了。

3.d. 基于PPPoE/PPPoA的ADSL

首先我们安装ADSL软件。

代码 4.1: 安装ppp软件包

```
# emerge net-dialup/ppp
```

注意： 如果你需要PPPoA，确保你使用的是>=baselayout-1.12.x。

其次，创建PPP网络脚本和PPP所使用的以太网卡的网络脚本：

代码 4.2: 创建PPP和以太网卡网络脚本

```
# ln -s /etc/init.d/net.lo /etc/init.d/net.ppp0
# ln -s /etc/init.d/net.lo /etc/init.d/net.eth0
```

确保在/etc/conf.d/rc设定RC_NET_STRICT_CHECKING="yes"。

现在我们需要配置/etc/conf.d/net。

代码 4.3: 一个基本的PPPoE设定

```
config_eth0=( null ) （指定你的以太网卡）
config_ppp0=( "ppp" )
link_ppp0="eth0" （指定你的以太网卡）
plugins_ppp0=( "pppoe" )
username_ppp0='user'
password_ppp0='password'
pppd_ppp0=(
        "noauth"
        "defaultroute"
        "usepeerdns"
        "holdoff 3"
        "child-timeout 60"
        "lcp-echo-interval 15"
        "lcp-echo-failure 3"
        noaccomp noccp nobsdcomp nodeflate nopcomp novj novjccomp
)

depend_ppp0() {
    need net.eth0
}
```

你也可以在/etc/ppp/pap-secrets中定义你的密码。

代码 4.4: /etc/ppp/pap-secrets范例

```
# *号非常重要
"username"  *  "password"
```

如果你通过一个USB调制解调器来使用PPPoE，你需要emerge br2684ctl。请阅读/usr/portage/net-dialup/speedtouch-usb/files/README来了解如果正确的配置它。

重要： 请仔细阅读/etc/conf.d/net.example里有关ADSL和PPP的部分。它包含了你的PPP设定可能需要的设置的详尽解释。

3.e. APIPA（自动获得私有的IP地址）

APIPA通过在某网络接口arping在169.254.0.0-169.254.255.255范围内的某个随机地址的方式来在该范围内尝试寻找空闲地址。如果没有收到任何ARP应答，该IP地址就会被绑定到这个网络接口上。

只有在网络上不存在DHCP服务器，同时本机不直接连接到internet并且其他所有的计算机都使用APIPA时才适用。

为了支持APIPA，请emerge net-misc/iputils或net-analyzer/arping。

代码 5.1: /etc/conf.d/net中的APIPA配置

```
# 首先尝试DHCP——如果DHCP失败则使用APIPA
config_eth0=( "dhcp" )
fallback_eth0=( "apipa" )
```

```
# 只使用APIPA
config_eth0=( "apipa" )
```

3.f. 绑定

为了使用连接绑定，请emerge net-misc/ifenslave。

绑定可以用来增加网络带宽。如果你有两块网卡连接到同一个网络，你可以把它们绑定在一起这样你的应用程序只看到一个网卡，但实际上却是在同时使用两个物理网卡传送数据。

代码 6.1: 在/etc/conf.d/net中配置绑定

```
# 把网络接口绑定在一起
slaves_bond0="eth0 eth1 eth2"

# 你可以不为绑定而得的网络接口配置IP地址
config_bond0=( "null" )

# 依赖eth0、eth1和eth2， 因为它们需要额外的配置
depend_bond0() {
  need net.eth0 net.eth1 net.eth2
}
```

3.g. 桥接（802.1d的支持）

为了获得对桥接支持，请emerge net-misc/bridge-utils。

桥接是用来把两个网络连接在一起。举例来说，你有一台通过ADSL modem接入internet的服务器，它同时通过无线网卡为其他计算机提供internet接入服务。你就可以创建桥接把这两个网络接口连接在一起。

代码 7.1: /etc/conf.d/net中桥接的配置

```
# 配置桥接——可以使用"man brctl"来获得更多信息
brctl_br0=( "setfd 0" "sethello 0" "stp off" )

# 增加端口到网桥br0中
bridge_br0="eth0 eth1"

# 你需要把端口配置成null，这样dhcp就不会启动了
config_eth0=( "null" )
config_eth1=( "null" )

# 最后为网桥配置一个IP地址——你也可以使用DHCP
config_br0=( "192.168.0.1/24" )

# 依赖eth0和eth1，因为它们需要额外的配置
depend_br0() {
  need net.eth0 net.eth1
}
```

重要：在使用一些桥接的设定时，你可能需要参考文档变量名称

3.h. MAC地址

如果你使用的是sys-apps/baselayout-1.11.14或者更新版本的话，你无需安装任何软件就可以更改网络接口对应的MAC地址或者为某个网络接口指定一个MAC地址。不过，如果你需要为网络接口指定一个随机的MAC地址或者你的baselayout的版本比上述版本要旧的话，你需要安装好net-analyzer/macchanger后才能使用这些功能。

代码 8.1: MAC地址变更示例

```
# 设置网络接口的MAC地址
mac_eth0="00:11:22:33:44:55"

# 只随机选择MAC地址的最后3个字节
mac_eth0="random-ending"

# 在同种物理连接类型(比如：光纤、铜缆、无线)中随机选择，范围至所有制造商
mac_eth0="random-samekind"

# 在任意物理连接类型(比如：光纤、铜缆、无线)中随机选择，范围至所有制造商
mac_eth0="random-anykind"

# 完全随机——警告：一些通过这个方式获得的MAC地址有可能无法正常工作
mac_eth0="random-full"
```

3.i. 隧道

你不需要安装任何软件就可以使用隧道，因为网络接口处理程序会帮你做好这些事。

代码 9.1: 在/etc/conf.d/net中配置隧道

```
# GRE隧道
iptunnel_vpn0="mode gre remote 207.170.82.1 key 0xffffffff ttl 255"

# IPIP隧道
iptunnel_vpn0="mode ipip remote 207.170.82.2 ttl 255"

# 配置网络接口
config_vpn0=( "192.168.0.2 peer 192.168.1.1" )
```

3.j. VLAN（802.1q的支持）

为了获得对VLAN的支持，请emerge net-misc/vconfig.

虚拟局域网（Virtual LAN）是一组有如运作在同个网段的一组网络设备的集合——尽管实际上可能不是。VLAN成员只能看到同一VLAN的成员，哪怕和其他VLAN成员处于同一物理网段中。

代码 10.1: 在/etc/conf.d/net中配置VLAN

```
# 像这样为网络接口指定VLAN号
# 请确认每个VLAN号前面没有以0填充
vlans_eth0="1 2"

# 你也可以配置VLAN
# 查看vconfig的手册页获取更多的信息
vconfig_eth0=( "set_name_type VLAN_PLUS_VID_NO_PAD" )
vconfig_vlan1=( "set_flag 1" "set_egress_map 2 6" )

# 按通常方法配置网络接口
config_vlan1=( "172.16.3.1 netmask 255.255.254.0" )
config_vlan2=( "172.16.2.1 netmask 255.255.254.0" )
```

重要： 在使用一些VLAN的设定时，你可能需要查看文档变量名称。

4. 无线网络

4.a. 介绍

目前您可以使用我们提供的wireless-tools 或wpa_supplicant工具来配置无线网络。请记住重要的一点是，您对无线网络的配置是全局性的，而非针对具体的接口。

wpa_supplicant是一个最好的选择，但缺点是它不支持所有的驱动。请浏览wpa_supplicant网站获得它所支持的驱动列表。另外，wpa_supplicant目前只能连接到那些你已经配置好ESSID的无线网络。

wireless-tools支持几乎所有的无线网卡和驱动，但它不能连接到那些只支持WPA的AP。

警告： 由于linux-wlan-ng驱动有其自己特有的安装和配置方式，而且与其他软件的方式完全不同，因此目前它并不被baselayout所支持。有传闻说linux-wlan-ng的开发者要将它的设置方式改成wireless-tools的方式——到时候您就可以在baselayout中使用linux-wlan-ng了。

4.b. WPA Supplicant

WPA Supplicant工具包可以让您连接到那些使用WPA的AP。因为还只是beta版，所以它的配置方法仍会常常变化——尽管如此，在大部分情况下它已经能很好的工作。

代码 2.1: 安装wpa_supplicant

```
# emerge net-wireless/wpa_supplicant
```

重要： 要让wpa_supplicant正常工作，您必须在内核中打开CONFIG_PACKET支持。

现在我们需要配置一下/etc/conf.d/net文件以便我们可以选择使用wpa_supplicant而不用wireless-tools（两者都安装在系统中时，默认使用的是wireless-tools）。

代码 2.2: 在/etc/conf.d/net中配置wpa_supplicant

```
# 使用wpa_supplicant代替wireless-tools
modules=( "wpa_supplicant" )

# 因为wpa_supplicant还不能很好的自动侦测驱动，所以需要我们为其指定正在使用的驱动。
wpa_supplicant_eth0="-Dmadwifi"
```

注意： 如果您用host-ap驱动，您首先要将无线设备卡设置成*Managed模式*以便能正确地配合wpa_supplicant工作。你可以在/etc/conf.d/net中设置iwconfig_eth0="mode managed"来实现这一点。

看起来这很简单，不是么？不过我们还需要配置wpa_supplicant本身，这将会比较麻烦一些，具体取决于你要连接的AP的安全程度。下面的例子是从/usr/share/doc/wpa_supplicant-<version>/wpa_supplicant.conf.gz中抽取并简化而来的，此文件出自wpa_supplicant软件包。

代码 2.3: 一个/etc/wpa_supplicant/wpa_supplicant.conf的例子

```
# 请不要修改下面这一行内容，否则将不能正常工作
ctrl_interface=/var/run/wpa_supplicant

# 确保只有root用户能读取WPA的配置
ctrl_interface_group=0

# 使用wpa_supplicant来扫描和选择AP
ap_scan=1

# 简单的情形：WPA-PSk密码验证方式，PSK是ASCII密码短语，所有合法的加密方式都允许连接
network={
  ssid="simple"
  psk="very secret passphrase"
  # 优先级越高，就能越早匹配到。
  priority=5
}

# 与前面的设置相同，但要求对特定的SSID进行扫描（针对那些拒绝广播SSID的AP）
network={
  ssid="second ssid"
  scan_ssid=1
  psk="very secret passphrase"
  priority=2
}

# 仅使用WPA-PSK方式。允许使用任何合法的加密方式的组合
network={
  ssid="example"
  proto=WPA
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP WEP104 WEP40
  psk=06b4be19da289f475aa46a33cb793029d4ab3db7a23ee92382eb0106c72ac7bb
  priority=2
}

# 明文连接方式（不使用WPA和IEEE802.1X）
network={
  ssid="plaintext-test"
  key_mgmt=NONE
}

# 共享WEP密钥连接方式（不使用WPA和IEEE802.1X）
network={
  ssid="static-wep-test"
  key_mgmt=NONE
  # 引号包含的密钥是ASCII密钥
  wep_key0="abcde"
  # 没有引号包含的密钥是十六进制密钥
  wep_key1=0102030405
  wep_key2="1234567890123"
  wep_tx_keyidx=0
  priority=5
}

# 共享WEP密钥连接方式（无WPA和IEEE802.1X），使用共享密钥IEEE802.11验证方式
network={
  ssid="static-wep-test2"
  key_mgmt=NONE
  wep_key0="abcde"
  wep_key1=0102030405
  wep_key2="1234567890123"
  wep_tx_keyidx=0
  priority=5
  auth_alg=SHARED
}

# 在IBSS/ad-hoc网络中使用WPA-None/TKIP
network={
  ssid="test adhoc"
  mode=1
  proto=WPA
  key_mgmt=WPA-NONE
  pairwise=NONE
  group=TKIP
  psk="secret passphrase"
}
```

4.c. Wireless Tools

初始设置和管理模式

Wireless Tools提供了一个通用的方法设置无线网络接口，最高可达WEP安全等级。虽然WEP是一种较弱的安全方式，但它也是最普遍使用的加密方式。

Wireless Tools的配置由几个主要变量来控制，以下配置文件的例子描述了您所需要了解的所有内容。要牢记于心的是：可确保“连接到没有加密的并且信号最强的AP”的配置并不存在——但我们会一直尝试并帮您连接到某个AP。

代码 3.1: 安装wireless-tools

```
# emerge net-wireless/wireless-tools
```

注意： 尽管您可以将无线设置保存在/etc/conf.d/wireless中，但是本指南还是推荐您将它们保存在/etc/conf.d/net之中。

重要： 您*将*需要参阅文档变量名称。

代码 3.2: 在/etc/conf.d/net中配置iwconfig的例子

```
# 使用iwconfig而不用wpa_supplicant
modules=( "iwconfig" )

# 为名为ESSID1和ESSID2的AP配置WEP密钥
# 您最多可以配置4个WEP密钥，但任何时候只有其中1个起作用。
# 所以我们提供一个默认的下标[1]来设置密钥[1]，之后紧接着把活动密钥设置为[1]。
# 我们这么做以备你让其他ESSID使用[1]以外的WEP密钥。
#
# key加上前缀s：意味着它是一个ASCII密钥，否则它就是一个16进制密钥。
#
# enc open 指定开放安全性（最安全）
# enc restricted 指定限制安全性（较不安全）
key_ESSID1="[1] s:yourkeyhere key [1] enc open"
key_ESSID2="[1] aaaa-bbbb-cccc-dd key [1] enc restricted"

# 以下仅在我们扫描可用的AP时起作用

# 有时有多个AP可见，所以我们需要规定一个首选的连接次序。
preferred_aps=( "ESSID1" "ESSID2" )
```

细致调整AP选择

您可以添加一些额外的选项来细致的调整AP的选择，不过正常情况下并不需要这么做。

您可以决定是否只连接首选的AP。默认情况下，当配置中列出的所有AP的连接都失败后，这时如果环境中有一个非加密的AP，系统将会与其连接。这个行为可以用associate_order变量来进行控制。下面给出一个相关的值的列表以及它们如何控制AP的选择。

值	描述
any	**默认行为**
preferredonly	**只连接首选列表里的可见AP**
forcepreferred	**按首选列表里的顺序强制连接AP，如果扫描不到的话**
forcepreferredonly	**不进行扫描——按首选列表里的顺序直接尝试连接AP**
forceany	**和forcepreferred一样，外加连接到任何其他可用的AP**

最后我们还提供了一些blacklist_aps和unique_ap的选择。blacklist_aps的工作方式和preferred_aps类似。unique_ap是一个yes或no的值，它决定了是否允许两个无线接口同时接入一个AP。

代码 3.3: blacklist_aps和unique_ap的例子

```
# 有时您根本不想连接到某些AP
blacklist_aps=( "ESSID3" "ESSID4" )

#如果您有多个无线网卡，您可以决定是否允许每个卡都能连接到同一个无线AP
# 可以取的值是"yes"或者"no"
# 默认设置为"yes"
unique_ap="yes"
```

Ad-Hoc和Master模式

当您在管理模式中无法连接到任何AP时，您也可以将自己的设备设置成Ad-Hoc节点。

代码 3.4: 失败后转用ad-hoc模式

```
adhoc_essid_eth0="This Adhoc Node"
```

那么，要如何连接到Ad-Hoc网络，或者干脆运行于Master模式使自身成为一个无线接入点呢？这里有这样一个设置！您可能需要参照本章前面的内容来指定WEP密钥。

代码 3.5: ad-hoc/master配置的例子

```
# 设定模式为managed（默认）、ad-hoc或者master。并不是所有的设备都支持所有的模式。

mode_eth0="ad-hoc"

# 设定接口的ESSID
# 在managed模式中，这将强制此接口只尝试连接特定的ESSID。

essid_eth0="This Adhoc Node"

# 指定使用的频道，否则将默认使用频道3。

channel_eth0="9"
```

重要： 以下是从NetBSD 文档的BSD wavelan文档中逐字逐句复制过来的内容。目前共计有14个可用的频道；1-11频道在北美是合法的，而在欧洲大部分地区则是频道1-13，在法国是频道10-13，在日本只允许使用频道14。如果有疑问，请参考随您所购买的无线网卡或AP附带的说明书。确保调整您的无线网卡与AP（或工作于ad-hoc模式的另一块无线网卡）使用同一个频道。默认情况下，在北美和欧洲大部分地区销售的无线网卡使用频道3；在法国使用频道11，在日本使用频道14。

Wireless Tools故障修除

一些环境或驱动的问题可能会使无线网络不能正常工作，下表多给出一些变量，可能有助于你解决问题。

参数	默认值	描述
iwconfig_eth0		请参看iwconfig的man page了解iwconfig各项参数
iwpriv_eth0		请参看iwpriv的man pages了解iwpriv各项参数
sleep_scan_eth0	0	在尝试扫描前的休眠时间（以秒为单位）。当驱动 / 固件需要时间激活时设置这个值。
sleep_associate_eth0	5	无线网络尝试连接到AP等待的秒数，超时则转向下一个AP。
associate_test_eth0	MAC	一些驱动程序在失去连接或尝试连接时不会重置无效AP的MAC地址，而有一些驱动程序在碰到这些情况时不会重设quality level。这里有效的设定是MAC，quality和all。
scan_mode_eth0		某些驱动必须在ad-hoc模式下扫描，因此若扫描失败的话请尝试把此变量设置成ad-hoc
iwpriv_scan_pre_eth0		扫描前先向接口发送一些iwpriv命令。更多细节请参看iwpriv的man page。
iwpriv_scan_post_eth0		扫描后向接口发送一些iwpriv命令。更多细节请参看iwpriv的man page。

4.d. 针对每个ESSID的网络配置

有时，您连接*ESSID1*需要使用一个固定IP，而连接*ESSID2*要使用DHCP。实际上，大多数模块变量可以针对每个ESSID来定义。下面我们给出具体的做法。

注意： 只有使用WPA Supplicant或者Wireless Tools时，这些设定才起作用。

重要： 您*将*需要参阅文档变量名称。

代码 4.1: 覆盖每一个ESSID的配置

```
config_ESSID1=( "192.168.0.3/24 brd 192.168.0.255" )
routes_ESSID1=( "default via 192.168.0.1" )

config_ESSID2=( "dhcp" )
fallback_ESSID2=( "192.168.3.4/24" )
fallback_route_ESSID2=( "default via 192.168.3.1" )

# 我们可以定义DNS服务器和其他的一些东西
# 注意:DHCP将覆盖这些设定，除非我们要求它不要覆盖
dns_servers_ESSID1=( "192.168.0.1" "192.168.0.2" )
dns_domain_ESSID1="some.domain"
dns_search_domains_ESSID1="search.this.domain search.that.domain"

# 根据无线AP的MAC地址来覆盖相关设定
# 这在不同地点有相同ESSID的情况下非常有用
config_001122334455=( "dhcp" )
dhcpcd_001122334455="-t 10"
dns_servers_001122334455=( "192.168.0.1" "192.168.0.2" )
```

5. 附加功能

5.a. 标准函数钩子

你可以定义4个函数，它们将会在start/stop操作前后被调用。这些函数将会以接口名称作为参数被调用，以便于

一个函数可以控制多个适配器。

preup()和predown()函数的返回值必须为0（成功），这意味着可以使一个网络接口进入配置状态或退出配置状态。如果preup()返回一个非0值，则网络接口配置过程将被中止。如果predown()返回一个非0值，则停止该网络接口运行的操作将被终止。

postup()和postdown()函数的返回值会被忽略，因为就算它们返回失败，系统也没有什么可以做的。

${IFACE}表示要打开/关闭的接口。${IFVAR}是${IFACE}转化而来的bash允许的变量名。

代码 1.1: pre/post up/down函数范例

```
preup() {
  # 在打开接口之前先测试它的网络物理连接是否已连接上。这只在某些网络
  # 适配器上有效，而且需要安装ethtool包。
  if ethtool ${IFACE} | grep -q 'Link detected: no'; then
    ewarn "No link on ${IFACE}, aborting configuration"
    return 1
  fi

  # 记得返回0以表示操作成功
  return 0
}

predown() {
  # 脚本默认内容是测试根文件系统是否是NFS提供的。在此情况下系统是不会允
  # 许你关闭接口的。注意，如果你定义了一个predown()函数，你就覆盖了这个逻辑。
  # 如果你仍然需要它的话，请往下看……

  if is_net_fs /; then
    eerror "root filesystem is network mounted -- can't stop ${IFACE}"
    return 1
  fi

  # 记得返回0以表示操作成功
  return 0
}

postup() {
  # 这个函数可以被用来实现一些功能，比如注册一个动态DNS服务。
  # 而另一个可能是在网络接口一启动时就开始传送/接收邮件。
  return 0
}

postdown() {
  # 这个函数出现在这主要是为了完整性……我还没有想到任何可以用它来做的好玩的事情 ;-)
  return 0
}
```

5.b. 无线工具函数钩子

注意： 此功能不适用于WPA Supplicant——但是${ESSID}和${ESSIDVAR}变量在postup()函数中是可用的。

你可以定义2个函数，它们将会在执行相关的操作前后被调用。这些函数将会以接口名称作为参数被调用，以便于一个函数可以控制多个适配器。

preassociate()函数的返回值应该为0（成功），这意味着一个网络接口的配置或取消配置过程可以继续。如果preassociate()返回一个非0值，则网络接口的配置过程将被中止。

postassociate()函数的返回值是被忽略的，因为就算它们返回失败，系统也没有什么可以做的。

${ESSID}被设置为你所连接到的AP的ESSID。${ESSIDVAR}是${ESSID}转化而来的bash允许的变量名。

代码 2.1: pre/post association functions

```
preassociate() {
  # 下面添加了两个配置变量leap_user_ESSID和leap_pass_ESSID。
  # 当你已经连接的ESSID的这两个变量都被配置时，我们就执行CISCO LEAP脚本。


  local user pass
  eval user=\"\$\{leap_user_${ESSIDVAR}\}\"
  eval pass=\"\$\{leap_pass_${ESSIDVAR}\}\"

  if [[ -n ${user} && -n ${pass} ]]; then
    if [[ ! -x /opt/cisco/bin/leapscript ]]; then
      eend "For LEAP support, please emerge net-misc/cisco-aironet-client-utils"
      return 1
    fi
    einfo "Waiting for LEAP Authentication on \"${ESSID//\\\\//}\""
    if /opt/cisco/bin/leapscript ${user} ${pass} | grep -q 'Login incorrect'; then
      ewarn "Login Failed for ${user}"
```

```
        return 1
      fi
    fi

    return 0
}

postassociate() {
    # 这个函数出现在这主要是为了完整性……我还没有想到任何可以用它来做的好玩的事情 ;-)

    return 0
}
```

注意： ${ESSID}和${ESSIDVAR}在predown()和postdown()函数中不可用。

6. 网络管理

6.a. 网络管理

如果你经常移动你的电脑，你可能会遇到没有网线和没有插入网线或者没有无线网络可用的情况。你也许希望在插入网线或无线网络可用时网络能够自动连上。

在这里你可以找到帮助你实现这个功能的一些工具。

注意： 本章只介绍ifplugd，但还有一些工具如netplug可供选择。netplug是ifplugd的轻量级替代者，但是它依赖于你的内核网络驱动的正常工作，而很多驱动不能正常工作。

6.b. ifplugd

ifplugd是一个在插入或拔出网线时启动或停止网络连接的程序。它也可以检测你的无线网卡和AP的关联，或者是在AP进入范围时检测。

代码 2.1: 安装ifplugd

```
# emerge sys-apps/ifplugd
```

ifplugd的配置相当简单。配置文件放置在/etc/conf.d/net中。运行man ifplugd以获得详细的变量信息。同时，请查看/etc/conf.d/net.example以获得更多例子。

代码 2.2: ifplug设置范例

```
（把eth0替换成要监控的网卡）
ifplugd_eth0="..."

（监控一个无限网卡）
ifplugd_eth0="--api-mode=wlan"
```

除了管理多个网络连接之外，你可能需要增加一个工具来使得管理多个DNS服务器和配置更加容易。当你通过DHCP来获取IP地址时，这将非常有用。只需要emerge openresolv.

代码 2.3: 安装openresolv

```
# emerge openresolv
```

查看man resolvconf来学习他更多的特性。