从源代码打造一个最小化的 Linux 系统实作指南

从源代码打造一个最小化的 linux 系统实作指南
作者：greg o"keefe, gcokeefe@postoffice.utas.edu.au
译者：梁昌泰 linuxrat@gnuchina.org
v0.8, sep 2000 翻译日期：2001 年 01 月第 0.8 版

以下就是从源代码中打造一个最小化的 linux 系统的操作说明. 它曾经是《从加电启动到 bash 提示符(from powerup to bash prompt)》的一部分. 但是我将它们分离开来，以便使得它们更简短而更为集中化. 我们在此所要打造的系统是非常小的,而且并不准备作为工作产品来使用.如果您想从头开始打造一个有实际用途的系统,请参阅 gerard beekmans 所撰写的 linux 空手道实作指南篇 (linux from scratch howto)。

1. 您所需要具备的条件

我们首先要安装一个 linux 发行套件比如小红帽(redhat)到一个分区上,然后使用它来在另一个分区上打造一个新的 linux 系统.我将我们所要打造的系统称为目标系统(target)而把我们所使用来打造新系统的系统称为源头系统(source). 可别把这个源头(source)系统同我们同时使用的源码(source code)相混淆了哦. :)

因此，您得需要一台具有两个独立分区的机器. 如果可能，请尽量使用一台没有重要资料在里头的机器,以免数据受损. 您可以使用一个已经存在的 linux 系统作为源头系统，但是我并不推荐这种方式. 如果您不慎遗漏了我们打造的指令的某些参数，您有可能会意外地在这个系统上安装了一些没有必要的东西，有可能会导致不兼容和冲突。

旧型的 pc 机硬件，大部分的 486 机器或者更早的机型，其 bios 都有一些极其烦人的限制. 它们没有办法读取硬盘超过前 512 兆之后的空间. 当然，这个对于 linux 来说并不是什么大问题，因为只要 linux 能够引导启动了，将使用 linux 自己的磁盘 io，略过 bios 的调用. 但是为了能够让这些旧型机器能够引导 linux，那么 linux 内核必须存放在硬盘的前 512 兆之前的某个位置. 如果您正好有这么个旧型机器，您得准备好一个独立的且完全在前 512 兆范围内的硬盘分区，并将其挂载为/boot. 其它的分区就可以在任何位置，可以任意处理而不必担心是在硬盘的什么位置了。

上一次我打造这个系统时，所使用的源头系统是小红帽 6.1(redhat 6.1)，我安装了基本系统，附加有以下软件包：

* cpp (c++编译器)
* egcs (增强型 c 编译器)
* egcs-c++ (增强型 c++语言编译器)
* patch (打补丁程序)
* make (编译批处理解释器)
* dev86 (设备文件包)
* ncurses-devel (ncurses 库开发包)
* glibc-devel (glibc 库开发包)
* kernel-headers(内核源码头文件包)

我还安装了 x window 视窗系统和 mozilla 网络浏览器以便更轻松地阅读文档，而实际上这两个东东并不是

必要的. 在我竣工之时，这个源头系统大概使用了 350 兆的磁盘空间(看起来是多了一些，可是我还在纳闷为什么呢)。

竣工之时的目标系统占用了 650 兆磁盘空间，但是这个数值包含了所有的源码以及中途打造出来的文件.如果空间比较紧凑，您应该在每个软件包都打造完毕之后执行一下 make clean 来清除临时文件. 当然了，我对这个也是有点吃惊的。

最后，您的准备好我们所要用来打造系统的源码包. 这些就是我在本文所讨论的软件包. 这些软件包都可以从源码盘里面找到，或者从国际互联网上找到. 我会给出美国的站点和位于澳大利亚的镜像站点的地址。

* makedev (设备生成器包)
  美国站点:ftp://tsx-11.mit.edu/pub/linux/sources/sbin
  另外一个美国站点:ftp://sunsite.unc.edu/pub/linux/system/admin
* lilo (linux 引导器包)
  美国站点：ftp://lrcftp.epfl.ch/pub/linux/local/lilo/
  澳大利亚: ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/boot/lilo
* linux 内核包(kernel) 使用 主页上所列举的镜像站点而最好不要使用美国站点站点下载，因为这些站点通常是超负荷运转的。
  美国站点：ftp://ftp.kernel.org/pub/linux/kernel
  澳大利亚站点: ftp://kernel.mirror.aarnet.edu.au/pub/linux/kernel/
* gnu libc 库包 其本身，以及 liuxthreads 线程附加库可在以下地址下载到：
  美国站点:ftp://ftp.gnu.org/pub/gnu/glibc
  澳大利亚站点:ftp://mirror.aarnet.edu.au/pub/gnu/glibc
* gnu libc 附加库包 您可能还会需要 linuxthreads 线程附加库和 libcrypt 加密附加库. 如果 libcrypt 没在那个站点找到，那就是因为美国出口法律限制的原因， 那么您就可以从这里弄到 libcrypt 加密附加库. 通常 linuxthreads 线程附加库跟 libc 库是放在同一个地方的.
  libcrypt 加密附加库：ftp://ftp.gwdg.de/pub/linux/glibc
* gnu ncurses
  美国站点:ftp://ftp.gnu.org/gnu/ncurses
  澳大利亚站点:ftp://mirror.aarnet.edu.au/pub/gnu/ncurses
* sysvinit (初始化脚本包)
  美国站点:ftp://sunsite.unc.edu/pub/linux/system/daemons/init
  澳大利亚: ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/daemons/init
* gnu bash (命令解释器包)
  美国站点:ftp://ftp.gnu.org/gnu/bash
  澳大利亚站点:ftp://mirror.aarnet.edu.au/pub/gnu/bash
* gnu sh-utils (命令解释器工具包)
  美国站点:ftp://ftp.gnu.org/gnu/sh-utils
  澳大利亚站点:ftp://mirror.aarnet.edu.au/pub/gnu/sh-utils
* util-linux (linux 常用工具包) 本软件包包含有 agetty 和 login。
  另外一个站点:ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/
  澳大利亚站点:ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/misc

总结一下，您所需要的就是：
* 一台具有两个分别是 400 兆和 700 兆独立分区的机器，或许您可能会需要少一些.。
* 一个 linux 发行套件(譬如一个 redhat 光盘)和安装方式(譬如一个光驱)

* 以上所列举的源码的 tar 包。

我假定您可以自己安装源头系统，而用不着我来帮忙．从这里开始，我假定源头系统已经安装好了。

本小项目的第一个里程碑就是使得内核启动起来然后死翘翘，因为它没找到 init 初始化程序．也就是说我们得安装一个内核和安装 lilo．为了顺利安装 lilo，我们要用上在目标系统上/dev 目录下的设备文件．lilo 需要它们来实现底层必需的磁盘存取来写入引导扇区．makedev 正是用来创建这些设备文件的脚本程序(您当然可以只需要从源头系统当中复制出来，不过这可是作弊不劳而获哦)．但是最重
要的事情就是，我们需要一个文件系统来放置所有的这些东西。

2. 文件系统

我们的新系统是要安装在文件系统上的．因此首先我们得使用命令 mke2fs 来创建文件系统，然后将其挂载到某个地方．我建议是挂载到/mnt/target 这个目录上．接下来的操作中，我假定就用这个目录了．为了节省您的宝贵时间，您可以在/etc/fstab 文件里面添加上这一项，以便每次源头系统启动的时候就能够自动将这个目录挂载上。

当我们启动了目标系统，放置在/mnt/target 上的所有东西就会被当成了放置在/根目录上。

我们需要在目标系统上建立固定的目录结构．请参阅"文件层次结构标准(简称 fhs，file heirarchy standard)"，见于 文件系统一节来了解详情，或者只需要 cd 切换目录到目标系统所挂载的地方然后尽管执行以下命令：

```
mkdir bin boot dev etc home lib mnt root sbin tmp usr var
cd var; mkdir lock log run spool
cd ../usr; mkdir bin include lib local sbin share src
cd share/; mkdir man; cd man
mkdir man1 man2 man3 ... man9
```

因为 fhs 标准和大部分的软件包在手册页(man page)放置位置上的处理并不一致，因此我们需要做一个符号连接：

```
cd ..; ln -s share/man man
```

3. makedev(设备生成器)

我们要把源代码放置到目标系统的/usr/src 目录下面．因此，举个例子吧，如果您的目标系统是挂载在/mnt/target 这个地方，且您的 tar 包是放在/root 里面，那么您要做的就是：

```
cd /mnt/target/usr/src
tar -xzvf /root/makedev-2.5.tar.gz
```

然后就把这些 tar 包复制到您要解开它们的地方就行了．千万别迷糊了哦．;->

当您安装软件的时候，通常情况下您会把它们安装在正在使用的系统上．但是我们并不想这么做，因为我们是要把/mnt/target 当做根文件系统(root filesystem)，就是要把这些软件安装到这个地方．不同的软件包有不同的处理方式．比如说 makedev 设备生成器包，您要做的是：

```
root=/mnt/target make install
```

您得先在这个包当中的 readme 说明文件和 install 安装说明文件当中查出这些选项，或者执行命令./configure --help 查看帮助说明。

查看一下 makedev 包当中的 makefile 文件，看看它是怎样处理我们在命令行当中设置的 root 变量的. 接着通过执行 man ./makedev.man 来查看一下它的手册页，看看它是怎么起到作用的. 您会发现生成我们自己的设备的方式就是执行 cd /mnt/target/dev 然后./makedev generic. 请使用 ls 命令来看看它都为我们生成了哪些设备文件吧。

4. 内核

下一步就是生成内核了. 我假设您以前是做过编译内核这种事的，所以我就长话短说了. 如果要启动的内核已经准备好的话，那么要安装 lilo 就会更容易. 请返回到目标系统的 usr/src 目录，然后在那儿解开 linux 内核源码. 进入 linux 源码树(cd linux)然后使用您最喜欢的方式配置内核，比如 make menuconfig. 如果您想让自己的轻松一些，那么您可以为自己配置一个没有模块的内核. 如果您已经
配置了模块，那么您就得编辑 makefile 文件，找出 install_mod_path 并将其设置为/mnt/target.

现在您可以执行 make dep，make bzimage 了. 如果您设置了模块项，可以再执行 make modules，make modules_install. 把内核映象文件 arch/i386/boot/bzimage 和系统函数映象文件 system.map 复制到目标系统的 boot 启动目录/mnt/target/boot 下面，然后准备安装系统引导器 lilo 了。

5. lilo 系统引导器

lilo 包里面带有一个很小巧的脚本，名叫 quickinst. 请把 lilo 源码包解压到目标系统的源代码目录/mnt/target/usr/src 下面，然后执行该脚本，方法是：root=/mnt/target ./quickinst. 它会询问您一些关于您想怎样安装 lilo 的问题。

切记：因为我们已经设置 root 根系统为目标系统分区了，所以您回答提问时所给出的文件名同它是密切相关的. 比如当它询问您默认想启动哪个内核的时候，您的回答应该是 /boot/bzimage，而并不是 /mnt/target/boot/bzimage 哦. 我发现这个脚本里面有个小错误，它会提示说：

./quickinst: /boot/bzimage: no such file

但是您甭理这个提示就是了，不会有事的。

我们该让 quickinst 把引导扇区(boot sector)放在何处为妥呢？当我们重启时，我们希望可以选择引导进入源头系统或者目标系统或者其它共存于同一台机器的其它系统. 而且我们还希望我们要使用所编译的 lilo 来引导我们新系统的内核. 我们怎么把这两件事情合而为一呢？让我们先跑一小会儿题，看看 lilo 在一个双重启动的 linux 系统上是怎样引导 dos 的. 在这样的一个系统上的 lilo.conf 文件的
内容看起来可能会跟下面的差不多：

```
prompt
   timeout = 50
   default = linux

   image = /boot/bzimage
          label = linux
```

```
            root = /dev/hda1
            read-only

    other = /dev/hda2
    label = dos
```

如果机器是这么安装起来的，那么主引导记录(mbr，master boot record)就可以被 bios 读取并加载，然后 mbr 加载 lilo 启动引导器，而后者则给出一个提示．如果您在提示后面输入 dos，lilo 就会从 hda2 加载引导记录，就加载了 dos。

我们所要做的事情跟上头是一样的，除了在 hda2 的引导记录应该是另外一个 lilo 引导记录之外，也就是在 quickinst 所询问要安装的那个．因此来自 linux 发行套件的 lilo 会加载我们所编译安装的 lilo，然后我们所编译安装的 lilo 就会加载我们所编译安装的内核．当您重启后，您会看到两次 lilo 的提示。

长话短说，当 quickinst 询问您该把引导扇区(boot sector)放到什么地方时，您就回答目标系统所在的分区，比如说是：/dev/hda2.

现在来修改您的源头系统上的 lilo.conf 配置文件，那么看起来会有点像这个样子：

```
other = /dev/hda2
            label = target
```

修改完毕，接着执行 lilo 安装 lilo．我们应该可以第一个引导进入目标系统了。

本小项目的第一个里程碑就是使得内核启动起来然后死翘翘，因为它没找到 init 初始化程序．也就是说我们得安装一个内核和安装 lilo．为了顺利安装 lilo，我们要用上在目标系统上/dev 目录下的设备文件．lilo 需要它们来实现底层必需的磁盘存取来写入引导扇区．makedev 正是用来创建这些设备文件的脚本程序(您当然可以只需要从源头系统当中复制出来，不过这可是作弊不劳而获哦)．但是最重
要的事情就是，我们需要一个文件系统来放置所有的这些东西。

2. 文件系统

我们的新系统是要安装在文件系统上的．因此首先我们得使用命令 mke2fs 来创建文件系统，然后将其挂载到某个地方．我建议是挂载到/mnt/target 这个目录上．接下来的操作中，我假定就用这个目录了．为了节省您的宝贵时间，您可以在/etc/fstab 文件里面添加上这一项，以便每次源头系统启动的时候就能够自动将这个目录挂载上。

当我们启动了目标系统，放置在/mnt/target 上的所有东西就会被当成了放置在/根目录上。

我们需要在目标系统上建立固定的目录结构．请参阅"文件层次结构标准(简称 fhs，file heirarchy standard)"，见于 文件系统一节来了解详情，或者只需要 cd 切换目录到目标系统所挂载的地方然后尽管执行以下命令：

```
mkdir bin boot dev etc home lib mnt root sbin tmp usr var
cd var; mkdir lock log run spool
cd ../usr; mkdir bin include lib local sbin share src
cd share/; mkdir man; cd man
```

```
mkdir man1 man2 man3 ... man9
```

因为 fhs 标准和大部分的软件包在手册页(man page)放置位置上的处理并不一致，因此我们需要做一个符号连接：

```
cd ..; ln -s share/man man
```

3. makedev(设备生成器)

我们要把源代码放置到目标系统的/usr/src 目录下面. 因此，举个例子吧，如果您的目标系统是挂载在/mnt/target 这个地方，且您的 tar 包是放在/root 里面，那么您要做的就是：

```
cd /mnt/target/usr/src
tar -xzvf /root/makedev-2.5.tar.gz
```

然后就把这些 tar 包复制到您要解开它们的地方就行了. 千万别迷糊了哦. ;->

当您安装软件的时候，通常情况下您会把它们安装在正在使用的系统上. 但是我们并不想这么做，因为我们是要把/mnt/target 当做根文件系统(root filesystem)，就是要把这些软件安装到这个地方. 不同的软件包有不同的处理方式. 比如说 makedev 设备生成器包，您要做的是：

```
root=/mnt/target make install
```

您得先在这个包当中的 readme 说明文件和 install 安装说明文件当中查出这些选项，或者执行命令./configure --help 查看帮助说明。

查看一下 makedev 包当中的 makefile 文件，看看它是怎样处理我们在命令行当中设置的 root 变量的. 接着通过执行 man ./makedev.man 来查看一下它的手册页，看看它是怎么起到作用的. 您会发现生成我们自己的设备的方式就是执行 cd /mnt/target/dev 然后./makedev generic. 请使用 ls 命令来看看它都为我们生成了哪些设备文件吧。

4. 内核

下一步就是生成内核了. 我假设您以前是做过编译内核这种事的，所以我就长话短说了. 如果要启动的内核已经准备好的话，那么要安装 lilo 就会更容易. 请返回到目标系统的 usr/src 目录，然后在那儿解开 linux 内核源码. 进入 linux 源码树(cd linux)然后使用您最喜欢的方式配置内核，比如 make menuconfig. 如果您想让自己的轻松一些，那么您可以为自己配置一个没有模块的内核. 如果您已经
配置了模块，那么您就得编辑 makefile 文件，找出 install_mod_path 并将其设置为/mnt/target.

现在您可以执行 make dep，make bzimage 了. 如果您设置了模块项，可以再执行 make modules，make modules_install. 把内核映象文件 arch/i386/boot/bzimage 和系统函数映象文件 system.map 复制到目标系统的 boot 启动目录/mnt/target/boot 下面，然后准备安装系统引导器 lilo 了。

5. lilo 系统引导器

lilo 包里面带有一个很小巧的脚本，名叫 quickinst. 请把 lilo 源码包解压到目标系统的源代码目录/mnt/target/usr/src 下面，然后执行该脚本，方法是：root=/mnt/target ./quickinst. 它会询问您一些关于

您想怎样安装 lilo 的问题。

切记：因为我们已经设置 root 根系统为目标系统分区了，所以您回答提问时所给出的文件名同它是密切相关的．比如当它询问您默认想启动哪个内核的时候，您的回答应该是 /boot/bzimage，而并不是 /mnt/target/boot/bzimage 哦．我发现这个脚本里面有个小错误，它会提示说：

./quickinst: /boot/bzimage: no such file

但是您甭理这个提示就是了，不会有事的。

我们该让 quickinst 把引导扇区(boot sector)放在何处为妥呢？当我们重启时，我们希望可以选择引导进入源头系统或者目标系统或者其它共存于同一台机器的其它系统．而且我们还希望我们要使用所编译的 lilo 来引导我们新系统的内核．我们怎么把这两件事情合而为一呢？让我们先跑一小会儿题，看看 lilo 在一个双重启动的 linux 系统上是怎样引导 dos 的．在这样的一个系统上的 lilo.conf 文件的
内容看起来可能会跟下面的差不多：

```
prompt
   timeout = 50
   default = linux

   image = /boot/bzimage
          label = linux
          root = /dev/hda1
          read-only

   other = /dev/hda2
   label = dos
```

如果机器是这么安装起来的，那么主引导记录(mbr，master boot record)就可以被 bios 读取并加载，然后 mbr 加载 lilo 启动引导器，而后者则给出一个提示．如果您在提示后面输入 dos，lilo 就会从 hda2 加载引导记录，就加载了 dos。

我们所要做的事情跟上头是一样的，除了在 hda2 的引导记录应该是另外一个 lilo 引导记录之外，也就是在 quickinst 所询问要安装的那个．因此来自 linux 发行套件的 lilo 会加载我们所编译安装的 lilo，然后我们所编译安装的 lilo 就会加载我们所编译安装的内核．当您重启后，您会看到两次 lilo 的提示。

长话短说，当 quickinst 询问您该把引导扇区(boot sector)放到什么地方时，您就回答目标系统所在的分区，比如说是：/dev/hda2.

现在来修改您的源头系统上的 lilo.conf 配置文件，那么看起来会有点像这个样子：

```
other = /dev/hda2
            label = target
```

修改完毕，接着执行 lilo 安装 lilo．我们应该可以第一个引导进入目标系统了。

看起来好像我们打造的是一个毫无用处的系统．说真的，要让它能够有实用价值也并不是什么难事．首先要做的事情之一就是您应该使得根文件系统(root filesystem)以可读写方式挂载起来．sysvinit 软件包里面有干这活儿的脚本，就在/etc/init.d/mountall.sh 里面．还执行了一次 mount -a 把所有在/etc/fstab 当中的条目以您所指定的方式挂载起来．请在目标系统的 etc/rc2.d 目录下生成一个类似 s05mountall 的符号连接。

您可能会看到这个脚本会用到您尚未安装的命令．如果真是这样，找到包含该命令的软件包并安装之．请参看 随机小技巧(random tips)这一小节，了解如何查找软件包．

看看在/etc/init.d 里面的其它脚本．它们大部分都应该包含在任何正经的系统里面．一次添加一个，别忘了要确定添加下一个之前个个都运行无误．请对照文件层次结构标准(file heirarchy standard)，请参看 文件系统(filesystem)一节．那里有一个命令列表，都是该在/bin 和/sbin 的命令．请确定您已经把那里列举的所有命令都安装在系统上了．最好就是再找找相关这类问题的 posix 文档来看看．

从此，在这个系统里面添加更多必要的软件包就真是个事儿了．越是早些把编译工具，比如说 gcc 和 make 这些添加进去就越好．一旦这些都完工了，您就可以利用目标系统来自我生息，就会越来越简单了．

13. 更多信息

13.1 随机小技巧

如果您的 linux 系统上曾经使用 rpm 安装过有一个叫做 thingy 命令，而您想获知这个命令的源码来源，那么您就使用如下命令：

```
rpm -qif `which thingy`
```

如果您有小红帽 redhat 的源码光盘，那么您就可以使用下列命令安装源码包了：

```
rpm -i /mnt/cdrom/srpms/what.it.just.said-1.2.srpm
```

这个命令会把 tar 包以及任何 redhat 补丁包放到/usr/src/redhat/sources 目录下面。

13.2 资源链接

* 有一个关于从源代码编译软件的小型实作指南(mini-howto)，就是《软件打造小小实作指南(software building mini-howto)》：
http://www.linuxdoc.org/howto/software-building.html.

* 另外还有一个关于从一穷二白空手起家打造一个 linux 系统的实作指南.该文更为集中于打造一个有实际应用价值的系统,而不仅仅是一个实习.请看：《inux 系统空手道实作指南篇(the linux from scratch howto)》：
http://www.linuxfromscratch.org/.

* unix 文件系统标准(unix file system standard) 还有一个关于 unix 文件系统标准的 链接．这个标准描述了在一个 unix 系统中什么东东该呆在什么位置以及原因．它还描述了在/bin，/sbin 等等目录中最小化的要求．如果您的目标是要打造一个小而全的系统，那么这个标准正是一个好的参考．
ftp://tsx-11.mit.edu/pub/linux/docs/linux-standards/fsstnd/

14. administrivia

14.1 版权声明(copyright)

14.2 主页

本文最新的英文版本"from powerup to bash prompt"可在此找到：
http://learning.taslug.org.au/power2bash

14.3 您的反馈意见

我很乐意从读者您那儿得知任何评论、改进意见和建议. 请写信给我： greg o"keefe

14.4 鸣谢录

本文所提及的产品名称是相应持有者的商标，在此我一并致谢. 我想对以下人员致谢，因为他们的帮助才有了这篇实作指南.

michael emery
因其提醒我注意到unios.

tim little
因其提供了关于/etc/passwd的一些线索.

spakr on #linux in efnet
因其发现 syslogd需要/etc/services的支持以及介绍给我 使用短语``rolling your own""来表述从源码打造系统.

alex aitkin
因其引起了我对vico以及他的``verum ipsum factum""(对编译进一步的理解) 的注意.

dennis scott
因其纠正了我的十六进制计算错误.

jdd
因其指出一些拼写错误.

14.5 修订历史记录

0.8

* 最初版本. 自"from powerup to bash prompt(从加电启动到bash提示符)实作篇"分离独立出来.

14.6 未来计划(todo)

* 转换为docbook格式.