# Installing Linux on ZIP disk using ppa ZIP Drive Mini−Howto

# Table of Contents

**Installing Linux on ZIP disk using ppa ZIP Drive Mini−Howto....1**
John Wiggins, jwiggins@comp.uark.edu....1
1.Disclaimer....1
2.Introduction....1
3.Setting up the ZIP disk....1
4.Red Hat 4.2 installation....1
5.Slackware 2.2 installation....2
6.Creating /etc/fstab....2
7.Debian 1.2 Installation....2
8.Afterthoughts....2
1.Disclaimer....2
2.Introduction....3
2.1 What's new....3
2.2 Conventions....3
2.3 Updates....4
2.4 References....4
Acknowledgments/Contributors....4
3.Setting up the ZIP disk....4
3.1 Partitioning ZIP disk....5
3.2 Formatting and mounting the ZIP disk....5
3.3 Creating the boot disk....6
Configuring and making the kernel....6
Getting the kernel to a floppy....6
LILO installation....7
Creating the ext2 filesystem....7
Copying the essential files....7
Kernel only installation....8
Setting the root and swap on the floppy....8
4.Red Hat 4.2 installation....9
4.1 Personal setup....9
4.2 Package installation....9
What packages to get....9
Updates; errata....11
How to install packages with rpm; without glint....11
Which came first, pamconfig or pam?....12
4.3 Problems after installation of packages....12
The case of the missing /etc/ld.so.cache and libc.so.5....12
/etc/ld.so.cache....12
libc.so.5....12
Setting root password....13
What the install program created....13
Networking setup....13
5.Slackware 2.2 installation....15
5.1 Requirements....15
5.2 Installation....15
5.3 What to install....16
6.Creating /etc/fstab....16

# Table of Contents

7.Debian 1.2 Installation........16
7.1 Requirements........16
7.2 Overview........17
7.3 Creating the modified Rescue disk........17
    Use dd (or RAWRITE under DOS) to create a new Rescue disk........17
    Build a new kernel with ZIP ppa support........17
    Mount the new Rescue disk........18
    Copy the kernel image........18
    Editing the 'rdev.sh' script........18
    Run this modified 'rdev.sh' script........19
7.4 Install the base system on the ZIP drive........19
7.5 Creating the boot disk........19
7.6 Reboot the system........19
7.7 Configure the base system and complete the install process........19
7.8 Installing the modules you built in step 7.3.2........20
8.Afterthoughts........20

# Installing Linux on ZIP disk using ppa ZIP Drive Mini−Howto


## John Wiggins, jwiggins@comp.uark.edu



v0.7, 26 January 1998


---


*This document is only useful for those with the printer port version of a ZIP drive who wish to have either a portable or backup Linux system on a ZIP disk.*


---


## 1.Disclaimer

## 2.Introduction

- 2.1 What's new
- 2.2 Conventions
- 2.3 Updates
- 2.4 References

## 3.Setting up the ZIP disk

- 3.1 Partitioning ZIP disk
- 3.2 Formatting and mounting the ZIP disk
- 3.3 Creating the boot disk

## 4.Red Hat 4.2 installation

- 4.1 Personal setup
- 4.2 Package installation
- 4.3 Problems after installation of packages

## 5.Slackware 2.2 installation


- 5.1 Requirements
- 5.2 Installation
- 5.3 What to install


## 6.Creating /etc/fstab

## 7.Debian 1.2 Installation


- 7.1 Requirements
- 7.2 Overview
- 7.3 Creating the modified Rescue disk
- 7.4 Install the base system on the ZIP drive.
- 7.5 Creating the boot disk
- 7.6 Reboot the system.
- 7.7 Configure the base system and complete the install process.
- 7.8 Installing the modules you built in step 7.3.2.


## 8.Afterthoughts

---

## 1.Disclaimer

NOTE: I have no idea if the IDE drive works the same way as the printer port version does since I don't have one, so please don't ask me.

The Debian install portion was basically just copied with little or no editing by this author. As such, there may be some duplication of instructions.

This document assumes the following:

- You have a printer port ZIP drive (since the ZIP Plus has both, I assume that it will work as well.)
- You already have Linux installed and running; this document is not for a first time install of Linux.
- You have ppa support in your current kernel or if module, the ppa module has been loaded.
- The mount point for the ZIP disk is the /iomega directory.

---

# 2.Introduction

This document is divided into four basic sections each describing how to install a bare−bones Linux system on a 100MB ZIP disk using a printer port ZIP drive. The first section describes how to set up the ZIP disk and is common to both Red Hat and Slackware distribution installations. The second, third, and fourth sections describe how to install Red Hat 4.2, Slackware 2.2, and Debian 1.2 distributions, respectively, onto the ZIP disk.

NOTE: I realize that Red Hat 5.0 has been released now, but between classes and work, well let's just say that it may be late May before I can get around to work on it. I also, hopefully, will be testing out the other distributions.

## 2.1 What's new

I've finally found enough time (although I really should be studying for a sociology test...) to update this document. Thanks for all who emailed me with comments.

New to this document:

- The Debian distribution
- LILO on the floppy
- New version of Red Hat (4.2 Biltmore)
- Network configuration (For Red Hat; Untested)

## 2.2 Conventions

Indicates the following text are commands:

**==>** Indicates something noteworthy:

```
NOTE:
```

Indicates a screen shot/capture:

```
    Text here.
```

## 2.3 Updates

For any updates, however rare they may be, please check: http://comp.uark.edu/~jwiggins/linuxZIP/

## 2.4 References

- Installation−HOWTO
- SCSI−HOWTO
- NET−3−HOWTO (for section 4.3.3)
- ZIP−Drive (mini−HOWTO)
- ParPort kernel patch (gives access to pass−through printer port) http://www.cyberelk.demon.co.uk/parport/

### Acknowledgments/Contributors

Slackware 2.2 section courtesy of Michael Littlejohn mike@mesa7.mesa.colorado.edu

Debian 1.2 section courtesy of John D. Blair jdblair@uab.edu

LILO information and many other helpful insight courtesy of Darcy Boese possum@niagara.com and Javier Rodriguez jrodrigu@nextgeninter.net.mx

---

# 3.Setting up the ZIP disk

(Common for both Red Hat and Slackware distributions.) Before starting, make sure that you have access to the ZIP drive; either by having ppa in the kernel or by having the ppa module loaded. One easy way to find this out is by checking dmesg:

**==> dmesg**

You may have to pipe this to more as dmesg tends to be rather long. Here's a snip from mine:

```
scsi0 : PPA driver version 0.26 using 4-bit mode on port 0x3bc.
scsi : 1 host.
  Vendor: IOMEGA    Model: ZIP 100           Rev: D.08
  Type:   Direct-Access                      ANSI SCSI revision: 02
Detected scsi removable disk sda at scsi0, channel 0, id 6, lun 0
SCSI device sda: hdwr sector= 512 bytes. Sectors= 196608 [96 MB] [0.1 GB]
 sda: Write Protect is off
 sda: sda1 sda2
```

If you only see something like:

```
scsi : 0 hosts.
scsi : detected total.
```

then you have SCSI support but the ZIP wasn't found.

## 3.1 Partitioning ZIP disk

To partition the ZIP disk, run fdisk:

**==> fdisk /dev/sda/**

Here is a snap of the partition table I have setup:

```
Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/sda1            1        1       81    82928   83  Linux native
/dev/sda2           82       82       96    15360   82  Linux swap
```

I decided to use a swap partition since I wanted to be able to use this with any machine.

## 3.2 Formatting and mounting the ZIP disk

After running fdisk, format the new partition:

**==> mke2fs −c /dev/sda1**

Then, create the swap partition: (15360 blocks as taken from fdisk)

**==> mkswap −c /dev/sda2 15360**

Last, you'll need to mount the ZIP disk:

**==> mount /dev/sda1 /iomega −t ext2**

## 3.3 Creating the boot disk

Since the ppa version of the ZIP drive isn't a true SCSI device, it isn't a bootable device and, therefore, requires a boot disk which has ppa included in the kernel and not as a module.

### Configuring and making the kernel

First, you'll need to configure and make a kernel that has ppa support enabled and not as a loadable module. In order to get to the ppa option, select SCSI support:

*SCSI support (CONFIG_SCSI) [Y/m/n/?]*

Plus, SCSI disk support:

*SCSI disk support (CONFIG_BLK_DEV_SD) [Y/m/n/?]*

And finally, under the SCSI low−level drivers, is the ppa support:

*IOMEGA Parallel Port ZIP drive SCSI support (CONFIG_SCSI_PPA) [Y/m/n/?]*

Again, be sure not to include ppa as a module, but rather in the kernel. Thus far, without the use of the parport kernel patch (see 1.4), the ppa driver will not allow the passive port of the ZIP drive to be used for a printer, so you may want to say no to parallel printer support:

*Parallel printer support (CONFIG_PRINTER) [N/y/m/?]*

```
NOTE: For more information concerning the ppa driver, please
      refer to the ZIP−Drive mini−HOWTO.
```

Once the kernel is configured, make the kernel:

**==> make dep; make clean; make zImage**

The new kernel should be found in arch/i386/boot/zImage.

### Getting the kernel to a floppy

After having to have 4 separate floppies due to different kernels and needing different parameters, (plus the great email I got telling me how to do this) I have included a section on LILO as one of the means of creating a bootable floppy.

## LILO installation

For those who have to, or for that matter just want to, have several kernels on one floppy (now adays, mine are too large) or just want to be able to pass arguments (such as single user mode) I received email on how to install LILO on a floppy.

## Creating the ext2 filesystem

To create an ext2 filesystem on a floppy, just do the same command for the ZIP disk:

**==> mke2fs −t /dev/fd0**

## Copying the essential files

Next, make sure there's a directory for a mount point, and mount the floppy (I used /mnt/floppy):

**==> mount /dev/fd0 /mnt/floppy −t ext2**

is to boot properly, you'll need the same files that LILO uses on your current Linux installation.

```
    NOTE: The file locations here are from my machine
          and may not be the same for everyone.
```

**==> cp /boot/boot.b /mnt/floppy**

**==> cp /boot/map /mnt/floppy**

**==> cp /usr/src/linux/arch/i386/boot/zImage /mnt/floppy/vmlinuzDESK**

Now to create the config file for LILO, now I miss the liloconfig program... (Thanks to Javier Rodriguez for this info) First, create the LILO config file, /mnt/floppy/lilo.conf, for the kernel(s) for the ZIP disk. Here's what I used so that I could have different kernels to test with:

```
boot=/dev/fd0
map=/mnt/floppy/map
install=/mnt/floppy/boot.b
prompt
compact
timeout=50
image=/mnt/floppy/vmlinuzLAP
   label=Laptop
   root=/dev/sda1
   read-only
```

```
image=/mnt/floppy/vmlinuzDESK
   label=Desktop
   root=/dev/sda1
   read-only
image=/mnt/floppy/vmlinuzDESK
   label=rescue
   root=/dev/hdc1
   read-only
```

I have two kernels, one for my 486 laptop which required the math−co emulation in the kernel and the other for my desktop. The rescue allows me to make an emergency boot to the hard drive.

Last but not least, with the floppy still mounted, run LILO to install it on the floppy with the command:

**==> lilo −C /mnt/floppy/lilo.conf**

Once LILO has been installed on the floppy, skip the next two steps, unless you enjoy doing this over again :)

## Kernel only installation

```
NOTE: This does not pertain for the LILO install.
```

Copy the newly made kernel to a floppy disk:

**==> cp arch/i386/boot/zImage /dev/fd0**

or

**==> cat arch/i386/boot/zImage > /dev/fd0**

Yes, there are many ways to copy the kernel to a floppy, but the last way, my favorite, is a little more encrypted. Try not to forget the > unless you like viewing binary files :)

## Setting the root and swap on the floppy

```
NOTE: This does not pertain for the LILO install.
```

Once the kernel is on the floppy, you need to set the root device to the ZIP disk: **==> rdev /dev/fd0 /dev/sda1** I'm not sure if the next option is needed, but I did it none the less. To set the swap:

**==> rdev −s /dev/fd0 /dev/sda2**

---

# 4.Red Hat 4.2 installation

With anything computer related, something 3 months old is considered obsolete and is in need of upgrading. Since I'm not always going to have the time to update this document with every update, I'll try my best to at least update it every other version. As for the other distributions, unless the authors wish to send updates, they well remain as they are.

## 4.1 Personal setup

For my installation, I have and used:

- Kernel 2.0.30
- Iomega ppa disk drive
- Red Hat 4.2

## 4.2 Package installation

When I first decided to attempt to install Red Hat on a ZIP disk, I figured it would be much easier to just use a Red Hat boot disk. Then I woke up. I came very close to actually getting a boot disk created, even got help from various folks at Red Hat via e−mail but ultimately I began to give up on the whole project when I discovered the −−root option with rpm.

### What packages to get

I found what packages to install by browsing a file I had found on one of Red Hat's mirrors. This file can be found on any mirror at:

```
redhat/redhat-4.2/i386/RedHat/base/comps
```

For this installation, I wanted to include network support but due to Red Hat's X network configuration, I'm going to have to manually configure, or rather manually attempt to configure, the network setup scripts found in /etc/sysconfig/ (see section 4.3.3.)

I decided against installing any development packages as the ZIP drive, at least an unpatched kernel version, is rather slow to get anything to compile. I also choose not to install X mainly for disk space issues. Later on, I may attempt to mount my hard drive and create a symlink with /usr to see if I can get X to work.

Following is a list of what packages I installed, listed in order of installation. Those marked with a * have updates from Red Hat's errata. In parentheses is the updated package number;

```
e.g. NetKit−B−0.09−6 was updated to NetKit−B−0.09−8 so the entry
    would be: *91) NetKit−B−0.09−6 (−8)
```

---

```
(List created by the command rpm --root /iomega -qa)

  1) setup−1.7−2                      2) pamconfig−0.51−2
  3) filesystem−1.3−1                 4) MAKEDEV−2.2−9
  5) adduser−1.7−1                    6) libc−5.3.12−18
  7) SysVinit−2.64−8                  8) ash−0.2−8
  9) at−2.9b−2                       10) libtermcap−2.0.8−4
 11) bash−1.14.7−1                   12) bc−1.03−6
 13) bdflush−1.5−5                   14) cpio−2.4.2−4
 15) cracklib−dicts−2.5−1            16) tmpwatch−1.2−1
 17) crontabs−1.5−1                 *18) db−1.85−10 (−11)
 19) dev−2.5.1−1                     20) diffutils−2.7−5
 21) etcskel−1.3−1                   22) file−3.22−5
 23) fileutils−3.16−1                24) findutils−4.1−11
 25) grep−2.0−5                      26) groff−1.10−8
*27) ld.so−1.7.14−4 (−5)             28) getty_ps−2.0.7h−4
 29) gzip−1.2.4−7                    30) mingetty−0.9.4−3
*31) initscripts−2.92−1 (93−1)       32) ed−0.2−5
 33) info−3.9−1                      34) ncurses−1.9.9e−4
 35) libg++−2.7.1.4−5               *36) pwdb−0.54−3 (−4)
 37) rootfiles−1.5−1                *38) pam−0.57−2 (−4)
 39) redhat−release−4.2−1            40) less−321−3
 41) mount−2.5l−2                    42) zlib−1.0.4−1
 43) rpm−2.3.11−1                    44) e2fsprogs−1.10−0
 45) sysklogd−1.3−15                 46) tar−1.11.8−11
 47) passwd−0.50−7                   48) gawk−3.0.2−1
 49) gdbm−1.7.3−8                    50) gpm−1.10−8
 51) hdparm−3.1−2                    52) kbd−0.91−9
 53) slang−0.99.37−2                 54) newt−0.8−1
 55) kbdconfig−1.4−1                 56) ncompress−4.2.4−7
*57) sh−utils−1.16−4 (−5)            58) procinfo−0.9−1
*59) logrotate−2.3−3 (4−1)           60) lilo−0.19−1
 61) losetup−2.5l−2                  62) linuxthreads−0.5−1
*63) mkinitrd−1.6−1 (7−1)            64) mailcap−1.0−3
*65) man−1.4h−5 (j−1)                66) mt−st−0.4−2
 67) modules−2.0.0−5                 68) mailx−5.5.kw−6
 69) net−tools−1.32.alpha−2          70) procmail−3.10−10
 71) procps−1.01−11                  72) psmisc−11−4
 73) quota−1.55−4                    74) readline−2.0−10
 75) sed−2.05−6                      76) setconsole−1.0−1
 77) sendmail−8.8.5−4                78) shadow−utils−960530−6
 79) stat−1.5−5                      80) tcsh−6.06−10
 81) termcap−9.12.6−5                82) textutils−1.22−1
 83) time−1.7−1                      84) timeconfig−1.8−1
 85) util−linux−2.5−38               86) vim−4.5−2
 87) vixie−cron−3.0.1−14             88) which−1.0−5
 89) zoneinfo−96i−4                  90) tcp_wrappers−7.5−1
*91) NetKit−B−0.09−6 (−8)           *92) lpr−0.18−1 (19−1)
*93) bind−4.9.5p1−2 (9.6−1)         *94) bind−utils−4.9.5p1−2 (9.6−1)
*95) wu−ftpd−2.4.2b12−6 (b15−1)      96) anonftp−2.3−3
 97) zip−2.1−1                       98) unzip−5.12−5
```

```
  99) statserial-1.1-7            100) minicom-1.75-2
 101) lrzsz-0.12.14-1             102) dip-3.3.7o-9
 103) ppp-2.2.0f-3                104) portmap-4.0-3
 105) perl-5.003-8               *106) traceroute-1.0.4.4bsd-2 (1.4a5-1)
*107) elm-2.4.25-7 (-8)           108) lynx-2.6-2
 109) ncftp-2.3.0-5               110) pine-3.95-2
 111) rdate-0.960923-1            112) apache-1.1.3-3
*113) nfs-server-2.2beta16-7     *114) nfs-server-clients-2.2beta16-7
      (2.2beta16-8)                     (2.2beta16-8)
```

And with all the above installed, I still have 32MB left!

## Updates; errata

As many, I hope, Red Hat users know, some packages may be found to have some security flaws or anything else which may cause issues to arise. For this reason, Red Hat releases updates for such packages. I have updated what packages I had which had updates and are marked in the above list. Please refer to Red Hat's web page concerning the updated packages at:

http://www.redhat.com/support/docs/rhl/rh42−errata−general.html

```
NOTE: Before you can update the packages, you'll have to
      run the ldconfig as described in section 3.3.1.1.
```

## How to install packages with rpm; without glint

With rpm, use the −−root option to specify the mounted directory as the root for installation. I had discovered that many packages were failing to install becuase of preinstall or postinstall scripts that weren't executing correctly due to the different root directory, thus, use the −−noscripts option:

**==> rpm −−root /iomega −i −−noscripts PACKAGE.i386.rpm**

As I'm sure many will notice, you should get an error message like:

```
failed to open /iomega/var/lib/rpm/packages.rpm

error: cannot open /iomega/var/lib/rpm/packages.rpm
```

So, just create the var/lib/rpm directory :)

**==> mkdir /iomega/var; mkdir /iomega/var/lib; mkdir /iomega/var/lib/rpm**

### Which came first, pamconfig or pam?

If anyone's tried to install pamconfig, it'll complain about a failed dependency of pam; and when you go try to install pam, pam complains about a failed dependency of pamconfig! This, being the chicken or the egg issue, puzzled me for a while, but thanks to the −−nodeps flag, we can force pamconfig to install; besides pam has more failed dependencies than just pamconfig.

**==> rpm −−root /iomega −i −−nodeps −−noscripts pamconfig−0.51−2**

## 4.3 Problems after installation of packages

Once everything is all nicely installed, unfortunately, the disk is not fully functionable, if that's a word. What I mean to say is, if you try to boot now with the floppy, you won't get very far. As soon as init tries to start up, you'll get two lovely errors; both of which complain about some files not being found that would have been made had the scripts been run.

### The case of the missing /etc/ld.so.cache and libc.so.5

If you tried booting, you'd get two error messages, the first will be the abscence of /etc/ld.so.cache file. The second complains about a missing libc.so.5.

### /etc/ld.so.cache

As mentioned by many readers, my previous instructions didn't quite work as stated. In order to get this file created, you'll have to run ldconfig while the ZIP disk is still mounted:

**==> chroot /iomega /sbin/ldconfig**

Thanks to Javier Rodriguez for this solution.

### libc.so.5

To solve the missing lib case, you'll have to create a symlink which would have been created by the installation scripts.

**==> cd /iomega/lib; ln −s libc.so.5.3.12 libc.so.5**

Thanks to Darcy Boese for this solution.

## Setting root password

Just as ldconfig was ran in 4.3.1.1, you might as well change/create a root password for this new system:

**==> chroot /iomega passwd root**

## What the install program created

```
NOTE: This is just a very brief setup, one which I haven't
      been able to test to see if it works.  In theory it
      should, but please do not send me complaints saying
      that this didn't work.
```

While exploring my Red Had 4.2 CD−ROM, I came across something rather interesting; the source code for the install program. I found it under /misc/src/install and one thing which I found of some use was the net.c file. In this, I found what other files that would be created had the install program been run. Most of these just give network support (hince the name net.c) but even if you don't have a network card, you can still use localhost for networking (plus apache will complain about not being able to determine a hostname.) These files include:

```
/etc/hosts
/etc/HOSTNAME
/etc/resolv.conf
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-eth0
(or any other network device you may have.)
```

## Networking setup

For this document, I wanted network support for my 3Com 3c595 fast−ethernet card (which for the last several months has been used in another machine.)

First I needed a name, and since I'm running my own name server (which is another long story) I gave myself the name: *dash−dot.wig.org* (I just liked the sound of it.) Having a name without an IP is kind of pointless, so I used a reserved non−internet usable network of 192.168.10.0 which my name server also uses. Even though the hostname is typically stored in /etc/HOSTNAME, Red Hat checks for /etc/sysconfig/network for this name; so let's start there. Sample of my /etc/sysconfig/network:

```
NETWORKING=yes
HOSTNAME=dash-dot
DOMAINNAME=wig.org
GATEWAY=
GATEWAYDEV=eth0
```

```
  NS1=192.168.10.7
```

Next, bascially duplicate the same info here for /etc/HOSTNAME, /etc/resolv.conf, and /etc/hosts:

**/etc/HOSTNAME:**

```
dash−dot.wig.org
```

**/etc/resolv.conf:**

```
search wig.org
nameserver ns.wig.org
```

**/etc/hosts:**

```
127.0.0.1 localhost
192.168.10.99 dash−dot.wig.org dash−dot
192.168.10.7  ns.wig.org ns
```

Red Hat configures all network devices from scripts found in /etc/sysconfig/network−scripts. The configuration of any network device is usually first created via the install program so I had to create these config files manually. They all begin with ifcfg−XXX where XXX is the network interface which ifconfig brings up; e.g. ppp0, eth0, etc. For this example, I had to create a file called ifcfg−eth0 which contains the following:

```
  DEVICE=eth0
  ONBOOT=yes
  BOOTPROTO=none
  BROADCAST=192.168.10.255
  NETWORK=192.168.10.0
  NETMASK=255.255.255.0
  IPADDR=192.168.10.99
```

And last but not least, in order to get this going right now, while in the /etc/sysconfig/network−scripts directory, just run:

**==> ./ifup ifcfg−eth0 boot**

This will start the script which configures the network interface if that interface was set to start at 'boot' time.

For further information, please refer to the NET−3−HOWTO document.

---

# 5.Slackware 2.2 installation

---

```
NOTE: This portion has not been updated.
```

---

## 5.1 Requirements

- ZIP Disk and Drive (obviously)
- Kernel with ZIP support
- ZIP Howto (recommended)
- 1.44" HD formated floppy
- 1 to 2 hours of time

## 5.2 Installation

Okay, now comes the fun part: Figuring out what files need to be on the system disk, and what packages that you want (and can fit) on your ZIP drive.

I decided that the easiest way to get started was install Slackware directly to the ZIP drive. I decided on this approach mostly because Slackware is a smaller distribution then Red Hat, and it would be easier to trim out what I didn't want. That and the fact that I am using the Slackware distribution anyway made it an obvious choice.

Installing Slackware onto the ZIP disk is easy, as root run the setup program, and choose /iomega as the install to partition, set the install from partition to where the Slackware sources are (cdrom, hardrive, etc), select install and follow the prompts.

## 5.3 What to install

The hardest part is deciding what to add, and what not to add. Obviously, you'll need the 'A' series (Which is about 8 megs), the rest is up to you.

I managed to trim down the Slackware release to a respectable installation of 70 megs, which included gcc/g++, perl, X11R6 (NOT ALL OF IT!), sendmail, online docs (Minus all the development man pages, but including all the howto's), and an assortment of other goodies, while leaving about 10 megs free for user files. YMMV

---

# 6.Creating /etc/fstab

(Common to both Red Hat and Slackware distributions)

The last thing that needs to be done before rebooting is to create the fstab file on the soon to be root partion. The following is what you should have as a minimum for /iomega/etc/fstab:

```
/dev/sda1      /        ext2       defaults   1   1
/dev/sda2      none     swap       sw
none           /proc    proc       defaults   1   1
```

Save the file, and reboot with the freshly made boot floppy and enjoy!

(Special thanks to Mike for reminding me about this very important and crucial step. − John)

---

# 7.Debian 1.2 Installation

```
NOTE: The author of this section sent this to me June, 11th 1997.
```

## 7.1 Requirements

- Ppa ZIP drive and disk.
- 2 blank 1.44 floppy disks
- A complete set of Debian install disks (review the Debian install docs if you don't remember how to make these)

- A couple hours of time

## 7.2 Overview

After spending a few hours wrestling with dpkg I decided it would be simpler to modify the Debian "Rescue" disk so that it would recognize the ppa ZIP drive. This proved to be very easy. You can then use this modified disk to procede through the normal Debian base system install. Once you've completed installing the base system you can use a boot disk to start the new base system and complete the installation using dselect. To use this technique you need to build two kernels – one with ppa and initial RAM disk support, and another without the RAM disk support.

If you want, you can skip all the steps in section 2 and let the Debian install procedure handle formatting the ZIP disk for you.

## 7.3 Creating the modified Rescue disk

The Debian rescue disk is a SYSLINUX style boot disk, which uses a DOS formatted floppy disk and a special boot loader to avoid loading MS−DOS. These disks are very easy to modify to start your own custom boot configuration. The Debian 'boot−floppies' package contains a set of scripts to automate the process of building boot disks. However, its so simple I found it easier to do the process by hand. This deviates a bit from the Debian philosophy, but I'm over it :). There are brief instructions in the 'readme.txt' file of the Rescue floppy.

### Use dd (or RAWRITE under DOS) to create a new Rescue disk.

Review the Debian install docs if you don't remember how to do this.

### Build a new kernel with ZIP ppa support

Build a new kernel with ZIP ppa support (as in step 3.3.1), but also configure RAM disk and initial RAM disk support. You also need to configure the msdos, fat, minix, ext2fs, and procfs filesystems.

Also configure any modules that you would like in your final installation on the ZIP disk.

Once the kernel is configured, build with:

**==> make dep; make clean**

**==>** ***make bzImage***

Build the modules with:

**==> make modules**

You will install these later.

```
NOTE: Make sure that you are using 'make bzImage', and
      not 'make zImage'.
```

## Mount the new Rescue disk.

**==> fdmount fd0**

or

**==> mount /dev/fd0 /mnt**

or

**==> whatever :)**

## Copy the kernel image

Copy the kernel image (on the i386 platform it will be located at arch/i386/boot/bzImage) to 'linux' on the floppy disk.

## Editing the 'rdev.sh' script

Open the 'rdev.sh' script located on the Rescue floppy with your favorite editor. Change the last line: from−−:

```
'rdev /mnt/linux /dev/ram0'
```

to−−:

```
'rdev /mnt/linux /dev/sda1'
```

You will also have to change all occurences of /mnt/linux to the appropriate path. Since I mount my floppies under /fd0, I had to change /mnt/linux to /fd0/linux.

### Run this modified 'rdev.sh' script.

**==> ./rdev.sh**

## 7.4 Install the base system on the ZIP drive.

Boot your modified Rescue disk. If all goes correctly you will be presented with the familiar Debian menu based install process, except that now it is aware of your ppa ZIP drive. Procede through this process as if you were installing the system on a normal hard drive, but mount /dev/sda1 as root and initialize /dev/sda2 as swap.

There is one deviation from the standard install process−− don't install and/or configure any loadable modules. You will install the modules you built in step 7.3.2 later.

## 7.5 Creating the boot disk

You can create the boot disk just as described in steps 3.3.1 to 3.3.2, or, if you want, just use the "Create Boot Disk" option during the Debian install. I like this second option because I get another SYSLINUX boot disk, allowing me to edit the greeting message to describe the nature of my custom boot floppy and allow me to enter additional kernel arguments. You can even include help files, accessible via the functions keys. You may still wish to rebuild the kernel and modify this boot disk by hand later to remove the RAM disk support. Your call.

## 7.6 Reboot the system.

Insert your boot disk and choose the 'Reboot' option from the install menu.

## 7.7 Configure the base system and complete the install process.

When the system reboots you will have a slow but completely workable Debian base installation running off of your ppa ZIP drive. Proceed normally with the installation at this point. I installed all the normal UNIX utilities, along with documentation sets, make, gcc, libraries, and various useful file manipulation utilities. The result is a very powerful emergency boot system that I can use to rescue any of the systems in our department in an emergency.

```
        NOTE: You'll at least need to install 'make' to
              complete the next step.
```

## 7.8 Installing the modules you built in step 7.3.2.

If you installed make in the last step, you should be able to mount the hard drive partition containing your kernel build, cd to the proper directory and run 'make modules_install'. Here's how I did it:

**==> mount /dev/hda2 /mnt**

**==> cd /mnt/usr/src/linux**

**==> make modules_install**

---

```
NOTE: You won't need to modify the /etc/fstab file, as explained
      in step 5.  The Debian installation process has already
      taken care of that.
```

---

## 8.Afterthoughts

Whew, and sigh. After amassing 31 ZIP disks and a ZIP unleashed battery pack, my next experiment is to try to get pcmcia working for my laptop.