Diskless Nodes HOW−TO document for Linux

Table of Contents

Table of Contents

Diskless Nodes HOW−TO document for Linux

Robert Nemkin buci@math.klte.hu , Al Dev (Alavoor Vasudevan) − Maintainer of this HOWTO alavoor@yahoo.com , Markus Gutschke markus+etherboot@gutschke.com , Ken Yap ken.yap@acm.org , Gero Kuhlmann gero@gkminix.han.de

v19.0, 07 Mar 2001

This document describes how to set up a diskless Linux box. As technology is advancing rapidly, network−cards are becoming cheaper and much faster − 100 MBits ethernet is standard now and in about 1 to 2 years 1000 MBits i.e. 1GigBits ethernet cards will become an industry standard. With high−speed network cards, remote access will become as fast as the local disk access which will make diskless nodes a viable alternative to workstations in local LAN. Also diskless nodes eliminates the cost of software upgrades and system administration costs like backup, recovery which will be centralized on the server side. Diskless nodes also enable "sharing/optimization" of centralised server CPU, memory, hard−disk, tape and cdrom resources. Diskless nodes provides mobility for the users i.e., users can log on from any one of diskless nodes and are not tied to one workstation. Diskless Linux box completely eliminates the need for local floppy disk, cdrom drive, tape drive and hard−disk. Diskless nodes JUST has a network card, 8MB RAM, a low−end cpu and a very simple mother−board which does not have any interface sockets/slots for harddisks, modem, cdrom, floppy etc.. With Diskless linux nodes you can run programs on remote Linux 64 CPU SMP box or even on Linux super−computer! Diskless nodes lowers the "Total Cost of Ownership" of the computer system. This document is copy-righted by Robert Nemkin and other authors as listed above. Copyright policy is GPL. Thanks to Bela Kis bkis@cartan.math.klte.hu for translating this initial document v0.0.3 (which was a mini−howto) to English.

---

5. Building EEPROM Burner

6. EPROM Burners and Memory chips

7. Introduction to Network Booting and Etherboot

8. Redhat Linux configuration

9. LanWorks BootWare PROMs

10. Etherboot

11. Netboot

12. Related URLs

---

1. Buying is cheaper than building!

Sometimes, buying a diskless linux computer will be cheaper than building!! Checkout the following commercial sites, which are selling diskless linux network−cards and diskless computers. These companies do **mass production** of Linux Diskless computers selling millions of units and thereby reducing the cost per unit. Each and every fortune 1000 companies in USA will be replacing the MS Windows PCs with diskless computers in near future as diskless linux computers can run both Linux and MS Windows 95 programs (via VMWare BIOS software). VMWare is NOT a emulator but has BIOS which allows you to install Windows 98/NT as guest OS to linux. You can use the 'xhost' command and DISPLAY environment from diskless node to run Windows95/Linux programs. See 'man xhost' on linux. You can also use Virtual Network Computing (VNC) to run Windows95/NT programs on linux diskless nodes. Get VNC from http://www.uk.research.att.com/vnc

- Linux Systems Labs Inc., USA http://www.lsl.com Click on "Shop On−line" and than click on "HardWare" where all the Diskless computers will be listed. Phone 1−888−LINUX−88.
- Diskless Workstations Corporation, USA http://www.disklessworkstations.com
- Unique Systems of Holland Inc., Ohio, USA http://www.uniqsys.com

Even if you buy diskless linux computer, you may be very much interested in reading this entire document.

---

2. Diskless Computer for Microsoft Windows 95/NT !!

Since Microsoft Windows 95/NT **DOES NOT** support diskless nodes, there is an intelligent work−around to overcome this short coming. Microsoft corporation will be **surprised** !!

2.1 VMWare package

Use the VMWare BIOS software with Linux which can host the Windows 95/98/NT. Linux will be the "host" OS and Windows 95/NT will be the "guest" OS. VMWare is NOT a emulator but has BIOS which allows you to install Windows 95/98/NT as the guest OS to linux. Install the VMWare on Linux server and than install Windows 95/NT on VMWare.

You can use the 'xhost' command and DISPLAY environment from **any** diskless node. See 'man xhost' on linux. At diskless node give −

```
        export DISPLAY=server_hostname:0.0
```

```
where server_hostname is the name of the server machine. And start X−terminal with
        xterm
```

Using VMWare, Diskless linux computers can run both Linux and MS Windows 95 programs. VMWare is at http://www.vmware.com.

2.2 VNC package from AT and T

You can also use the VNC (Virtual Network Computing) Technology from the telecom giant AT & T. VNC is GPLed and is a free software. Using VNC you can run Windows 95/NT programs on diskless linux computer but actually running on remote Windows95/NT server. VNC is at

---

3. Advantages of Diskless Computer

Diskless linux computer will become **immensely** popular and will be the product of this century and in the next century. The diskless linux computers will be very successful because of the availability of very high−speed network cards at very low prices. Today 100 Megabit per second (12.5 MB per sec transfer rate) network cards are common and in about 1 to 2 years 1000 MBit (125 MB per sec transfer rate) network cards will become very cheap and will be the standard.

In near future, Monitor manufacturers will place the CPU, NIC, RAM **right inside** the monitor to form a diskless computer!! This eliminates the diskless computer box and saves space. The monitor will have outlet for mouse, keyboard, network RJ45 and power supply.

The following are benefits of using diskless computers −

- Diskless Linux computers can run BOTH MS Windows 95/NT and linux programs.
- Total cost of ownership is very low in case of Diskless computers. Total cost of ownership is cost of initial purchasing + cost of maintainence. The cost of maintainence is usually **3 to 5 times** the cost of initial computer purchase and this cost is recurring year after year. In case of Diskless computers, the cost of maintainence is **completely eliminated**!!
- All the backups are centralized at one single main server.
- More security of data as it is located at server.
- No need of UPS battery, air−conditioning, dust proof environment for diskless clients, only server needs UPS battery, A/C and dust proof environment.
- Noise is completely eliminated since diskless computer does not have Fan motor, and local harddisk. Only server makes lots of noise but it is enclosed in a server room.
- Protection from Virus attack − Computer virus cannot attack diskless computers as they do not have any hard disk. Virus cannot do any damage to diskless computers. Only one single server box need to be protected against virus attack. This saves millions of dollars for the company by avoiding installtion of vaccines and cleaning the hard disks!!

- Server can have large powerful/high performance hard disks, can optimize the usage of disk space via sharing by many diskless computer users. Fault tolerance of hard disk failure is possible by using RAID on main server.
- Server can have 64 bit CPU SMP box having many CPUs or even linux super−computers. CPU power can be shared by many diskless computer users
- Sharing of central server RAM memory by many diskless computer users. For example, if many users are using web browser than at server RAM there will be only one copy of web browser in the RAM. In case Windows 95 PCs, many users need to have individual copy of web browser in local RAM and hence there is wastage of RAM space.
- Diskless computers are extremely fast because program loading time is completely eliminated. For example, if the server loads the StarOffice suite into memory due to request from one diskless user then if another diskless user wants to use the StarOffice suite then loading time is avoided since StarOffice is already loaded.
- Diskless linux computers can run programs on multiple servers using the "xhost" and DISPLAY environment.
- Very few system administrators required to maitain central server unlike Windows 95 PC clients which need many administrators.
- Zero administration at diskless client side. Diskless computers are absolutely maintainence free and troublefree.
- Long life of diskless clients − more than **300 years** without any hardware or software upgrades.
- Eliminates install/upgrade of hardware, software on diskless client side.
- Eliminates cost of cdrom, floppy, tape drive, modem, UPS battery, Printer parallel ports, serial ports etc..
- Prevents pilferage of hardware components as diskless node has very little RAM and low−cost CPU. The server has lots of memory and many powerful CPUs.
- Can operate in places like factory floor where a hard disk might be too fragile.

4. Quick Steps to implement Diskless Nodes

An overview to build diskless nodes is as follows:

- Download/Install redhat RPM packages from LTSP org
- Test with floppy disk (1.44MB) having the PROM program.
- Next you have to make the Network card which has the bootable prom
    - Either purchase NIC ready with prom or
    - Purchase the eproms
    - (or) Purchase Eprom burner to burn your own eproms. Transfer the tested program from floppy to prom via eprom burner

- Visit http://www.disklessworkstations.com to buy eprom burners and see also List of EPROM Burner manufacturers, Build EEPROM burner

4.1 Linux Terminal Server Project − LTSP

LTSP is an open source code project to build diskless linux computers.

At LTSP site you will find RPM packages for Redhat Linux and packages for Debian Linux which will save you lots of time. The subsequent chapters given in this document are for academic purposes only, which you can read them if you have more time.

Visit the LTSP and related sites at :−

- http://www.ltsp.org
- http://www.disklessworkstations.com
- http://www.slug.org.au/etherboot and at mirror−site and at google−site
- http://metalab.unc.edu/Linux/HOWTO/XFree86−Video−Timings−HOWTO.html

Related topics worth seeing −

- NCD X−terminal http://www.linuxdoc.org/HOWTO/mini/NCD−X−Terminal.html

---

5. Building EEPROM Burner

5.1 What is this ?

(**Note**: This chapter is written by Abhijit Dasgupta. Abhijit's email: takdoom@yahoo.com

The name of this project is EEP and it can be obtained from:

- Primary site (download tarball only): http://metalab.unc.edu/pub/Linux/apps/circuits/
- Browse and/or download: http://homepages.go.com/~abhijit_dasgupta/eep/index.html
- Browse and/or download: http://members.nbci.com/abhijit_dasgupta/eep/index.html

Please do not use the old URL for EEP anymore.)

EEP is an open hardware design (you are free to copy, use, and modify the hardware design) EEPROM burner for 24−pin and 28−pin 5−volt EEPROMs. There are various designs available, but my main goal was to have something which

- is easy to build and uses only the most commonly available parts,
- is cheap, and
- is controlled by Linux.

The latest version is EEP−0.2.

The ICs in EEP are all common 74HCT series logic chips, and it uses the PC parallel port interface. I wrote the driver code for Linux only, but it is GPL code, and it should be easy to modify it for other PC operating systems.

I use EEP to burn netboot PROMs for ethernet cards, which are used to make diskless linux boxes. See the netboot/etherboot packages for details of how to do that. You can also use it for microcontroller systems with external ROM (e.g. 8031).

A photo of the burner is at http://homepages.go.com/~abhijit_dasgupta/eep/eeprom−burner.jpg and a PostScript schematic diagram is at http://homepages.go.com/~abhijit_dasgupta/eep/schematic.ps.

5.2 Supported EEPROMs

Most 5−volt−programmable 24−pin and 28−pin EEPROMs should work with EEP−0.2. Here is a partial list of common EEPROMS that are known to work:

- 24−pin 2816/28C16, 2048 bytes (16 kilobits)
- 28−pin 2817/28C17, 2048 bytes (16 kilobits)
- 28−pin 2864/28C64, 8192 bytes (64 kilobits)
- 28−pin 28256/28C256, 32768 bytes (256 kilobits)

Various vendors manufacture these EEPROMs. Some are: Microchip, Atmel, Xicor, Catalyst, and STM.

5.3 Schematics and pinouts

The schematic is in PostScript (schematic.ps), but a GIF image (schematic.gif) is also included. The ascii version is older. In the schematic diagram, pin numbers are shown outside each IC diagram. Pin numbers for the big box on the right side are for the 28−pin ZIF socket.

The file pinouts.txt has pinout information for the ICs used.

For the 74HCT ICs used in the circuit, Vcc and Ground connections are not shown in the schematic. Of course, these pins must be properly connected. Please refer to the pinouts.txt file for full pinouts (in particular Vcc/Ground connections).

5.4 Construction

WARNING: It is easy to destroy the parallel port of your PC by connecting things to it. It is also possible to damage or destroy the whole PC, its attachments, peripherals, and people near it by improper connections and electrical accidents. **USE EXTREME CAUTION**.

Disclaimer: Use at your own risk. There is absolutely no warranty of any kind here, see COPYING/LICENSE below.

The programmer can be built on a breadboard, but use a protoboard for a more permanent version. Use 0.1uF power−bus bypass capacitors generously. The 5V power source can be obtained from the PC itself, but be careful here. The 28−pin ZIF socket is perhaps the most expensive component. If you are building on a breadboard, you may be able to get by without it (not recommended).

The 180 ohms resistor connecting pin 10 (Y6) of the upper 74HCT259 to pin 1 of the ZIF socket is a current limiting resistor to protect the 74HCT259 IC in cases where a 28−pin EEPROM with RDY/BSY pin is used. When using 32 kilobytes (256 kilobits) EEPROMs like the 28256, it is recommended that this resistor be shorted for more reliable operation.

5.5 Jumper setup

J1 and J2 are single−row 3−pin headers for jumpers. When using 28−pin EEPROMs, jumper the right two pins on both J1 and J2. For 24−pin EEPROMs, jumper the left two pins on both J1 and J2.

5.6 Low−justification of 24−pin EEPROM devices

When plugging in a 24−pin EEPROM device (like 2816) into the 28−pin ZIF socket, make sure the 24−pin device is low−justified in the ZIF socket. This means that pins 1, 2, 27, and 28 of the ZIF socket will remain unused, and the ground pin of the devices match up (i.e. pin 12 of the 24−pin device should sit in to pin 14 of the ZIF socket).

5.7 Parts List

- **ICs:** 74HCT123, 74HCT132, 74HCT138, 74HCT157, 74HCT574 (1 ea), and 2 74HCT259s.
- **Resistors:** 100K, 10K, 1K, 180 ohms, and 390 ohms (1 ea).
- **Capacitors:** 100pF, 1uF, (1 ea) and 3 0.1uF power−bus bypass capacitors.
- **Misc:** 1 LED, 1 SPST switch, 25−conductor ribbon cable with DB25 male connector, 28 pin ZIF socket (small breadboard can be used instead), header pins for jumpering.

5.8 If you have already built EEP−0.1

If you have already built the EEP−0.1 burner, you can make the following modificatons to make the EEP−0.2 burner:

1. remove the connection from 74HCT157 pin 1 (SEL) to the upper 74HCT259 pin 11 (Y6)
2. remove the 1K resistor that is connected from pin 1 of ZIF−socket to Vcc
3. add a new connection from pin 1 (SEL) of the 74HCT157 to pin 9 of of the DB−25 parallel port
4. add a new connection from pin 10 of the upper 74HCT259 to the unused pin of J1
5. add a 180 ohms resistor from pin 11 of the upper 74HCT259 to pin 1 of the ZIF−socket

5.9 How to build the software

Download the software http://metalab.unc.edu/pub/Linux/apps/circuits/EEP−0.2.tar.gz and unpack it. Then cd to the src directory and type `make'.

5.10 Usage

The progran eep is used for burning and reading an eeprom. It reads data from stdin and writes it to the eeprom. The data needs to be in binary (raw) format. None of the usual hex and/or ascii formats (Intel, Motorola srecord, etc) are supported, so if your assembler ouputs in only a hex/ascii format, you will need to convert it to binary (see, e.g., the Hex2bin and srecord, available from the metalab.unc.edu/pub/Linux archive). When reading, the output is also raw binary to stdout (unless the −t option is given).

```
Usage:

      eep  -0|-1|-2  -r|-w  -b|-t  offset  size
```

```
where:

    -0|-1|-2  -0 chooses port lp0, -1 port lp1, and -2 port lp2,
    -r|-w     -r reads the eeprom to stdout, and -w burns it from stdin,
    -b|-t     -b is normal (binary) mode, and -t is debugging (ascii hex),
    offset    is the start address within the eeprom, 0..32767, and,
    size      is the number of bytes to read/write, 0..32768.

The offset and size can be specified as a string of digits in decimal
notation, but will be taken as hexadecimal when there is a ``0x'' prefix,
and octal when preceded by ``0''.

Examples
--------

# Read the contents of a 2864 in binary (raw) form and save it in a file
eep -1 -r -b 0 8192 > contents.bin
# Same as:
eep -1 -r -b 0 0x2000 > contents.bin


# List 16 bytes starting at offset 128
eep -1 -r -t 128 16
# Same as:
eep -1 -r -t 0x80 0x10


# Write 16384 bytes from the file nepci.lzrom into the first-half of
# a  28C256 eeprom, through lp0:
cat nepci.lzrom | eep -0 -w -b 0 16384
```

5.11 Schematic Diagram in ASCII

```
                       +-------+                      J1
              +5-------|RST    |               +5---o o o----+   +-----------+
              +5--o----|/CLR1  |         10K          |      |   |           |
                  |    |       |-----o--/VVV\-- +5    +------|---|26 A13(+5V)|
      +------+    |    |1/2 123|     |              +--------|-->|27 /WE(NC) |
 16 o-|/CS2  |    |    |       |--||-+              | +------|-->|23 A11(/WE)|
      |   CS1|----o----|B1     | 100pF              | | J2   |   |           |
      |      |         |    /Q1|---------->---------o o o    |   |  ZIF28    |
      |    Y1|---------|/A1    |                        |    |   |  socket   |
      | 138  |         +-------+         _ 1/2 74HCT132 |    |   |   for     |
      |      |                     +5 --| \   __        |    |   |  EEPROM   |
      |    Y2|--------------------------|  O--| \       |    |   |           |
  8 o-|A2    |                +-------+ |_/   |  O-----------|-->|22 /OE     |
  7 o-|A1  Y4|--------------->|EN   Y7|-----o-|_/       |    |   |           |
  6 o-|A0  Y3|----+    +5-----|RST    |    |   180 ohm |    |   |           |
      |    Y0|-+  |           |     Y6|-----|---/VVV\---|----|---|1 A14(NC)  |
      |  /CS3| |  |           | 259 Y5|-----|-----------|----+   |           |
      +------+ |  |           |     Y4|-----|-----------|------->|2 A12(NC)  |
            |  |  |           |     Y3|-----|-----------+        |           |
  5 o--->---|--|--|--------o--|D    Y2|-----|------------------->|21 A10     |
  4 o--->---|--|--|------o-|--|A2   Y1|-----|------------------->|24 A9      |
  3 o--->---|--|--|----o-|-|--|A1   Y0|-----|------------------->|25 A8      |
  2 o--->---|--|--|--o-|-|-|--|A0     |     |                    |           |
```

```
             |  |  |  | | | |  +-------+     |      +5------------|28 +5V(NC) |
             |  |  |  | | | |                |                    |           |
             |  |  |  | | | |  +-------+     |                    |           |
             |  |  |  | | | |  |     Y7|-----|------------o------>|3  A7       |
             |  |  +---------->|EN     |-----|-----------o|------>|4  A6       |
             |  |     | | | |  |       |-----|----------o||------>|5  A5       |
             |  |     | | | |  | 259   |-----|---------o|||------>|6  A4       |
             |  |     | | | |  |       |-----|--------o||||------>|7  A3       |
             |  |     | | | |  |       |-----|-------o|||||------>|8  A2       |
             |  |     | | | +--|D      |-----|------o||||||------>|9  A1       |
             |  |     | | +----|A2   Y0|-----|-----o|||||||------>|10 A0       |
             |  |     | +------|A1     |     |     ||||||||       |           |
             |  |     +--------|A0  RST|     |     ||||||||       | ZIF28     |
             |  |              +-------+     |   +------------+   | socket    |
             |  |                    |       |   |   data in  |   |  for      |
             |  |                   +5       +-->|/OE         |   | EEPROM    |
             |  |                                |     574    |   |           |
             |  +------------------------------->|CLK         |   |           |
             +----+                              |   data out |   |           |
                  |                              +------------+   |           |
                  |              +------------+     ||||||||      |           |
   9 o--------------------------->| SEL        |     ||||||||      |           |
                  |              |          B3|<----|||||||o------|19 D7      |
  11 o---<-----------------------|Y3        B2|<----||||||o-------|18 D6      |
  12 o---<-----------------------|Y2        B1|<----|||||o--------|17 D5      |
  13 o---<-----------------------|Y1  157   B0|<----||||o---------|16 D4      |
  15 o---<-----------------------|Y0        A3|<----|||o----------|15 D3      |
                  |              |          A2|<----||o--- data---|13 D2      |
                  |              |          A1|<----|o---- bus ---|12 D1      |
                  |       GND----|/OE       A0|<----o-------------|11 D0      |
   +5--o--+       |              +------------+                   |           |
       |  |  __   o----------------------------------------------->|20 /CE   14|
     100K +-|  \   |  __                                           +---------+-+
   sw1 |    |   O-o-|  \ 1/2 74HCT132                                       |
   o-->o----|__/    |   O---390ohm--+                                       |
   |   |          +-|__/          |                                  GND -+
   |  --- 1uF     |                LED
   |  ---    +5--+                 |
   |   |                            |
   +---o----------------------------o- GND
```

Notes:

1. Pin numbers on the left margin are for DB25 parallel port.
3. A 24-pin chip (e.g. 2816) must be low-justified in the 28-pin ZIF socket.
2. Pin numbers in the right box are for the ZIF-28 socket, not the IC.
7. The signal labels inside the ZIF-28 socket box are for 28-pin EEPROMs
   (they are given in parentheses for 24-pin EEPROMs).
4. J1 and J2 are single-row 3-pin headers for jumpers (or use a DPDT switch).
5. For 28-pin EEPROMs, jumper the right two pins of both J1 and J2.
6. For 24-pin EEPROMs, jumper the left two pins of both J1 and J2.
8. The SPST switch sw1 needs to be open to enable operation of the programmer.
9. Please refer to the file pinouts.txt for full pinouts of the ICs used.

Abhijit Dasgupta
takdoom@yahoo.com

6. EPROM Burners and Memory chips

Below is the information about EPROM and various types of memory chips.

6.1 Non−Volatile Memory chips

Here is the brief descriptions of memory chips and their types.

- **PROM**: Pronounced prom, an acronym for programmable read−only memory. A PROM is a memory chip on which data can be written only once. Once a program has been written onto a PROM, it remains there forever. Unlike RAM, PROMs retain their contents when the computer is turned off. The difference between a PROM and a ROM (read−only memory) is that a PROM is manufactured as blank memory, whereas a ROM is programmed during the manufacturing process. To write data onto a PROM chip, you need a special device called a PROM programmer or PROM burner. The process of programming a PROM is sometimes called burning the PROM. An EPROM (erasable programmable read−only memory) is a special type of PROM that can be erased by exposing it to ultraviolet light. Once it is erased, it can be reprogrammed. An EEPROM is similar to a PROM, but requires only electricity to be erased.
- **EPROM**: Acronym for erasable programmable read−only memory, and pronounced e−prom, EPROM is a special type of memory that retains its contents until it is exposed to ultraviolet light. The ultraviolet light clears its contents, making it possible to reprogram the memory. To write to and erase an EPROM, you need a special device called a PROM programmer or PROM burner. An EPROM differs from a PROM in that a PROM can be written to only once and cannot be erased. EPROMs are used widely in personal computers because they enable the manufacturer to change the contents of the PROM before the computer is actually shipped. This means that bugs can be removed and new versions installed shortly before delivery. A note on EPROM technology: The bits of an EPROM are programmed by injecting electrons with an elevated voltage into the floating gate of a field−effect transistor where a 0 bit is desired. The electrons trapped there cause that transistor to conduct, reading as 0. To erase the EPROM, the trapped electrons are given enough energy to escape the floating gate by bombarding the chip with ultraviolet radiation through the quartz window. To prevent slow erasure over a period of years from sunlight and fluorescent lights, this quartz window is covered with an opaque label in normal use.
- **EEPROM**: Acronym for electrically erasable programmable read−only memory. Pronounced double−e−prom or e−e−prom, an EEPROM is a special type of PROM that can be erased by exposing it to an electrical charge. Like other types of PROM, EEPROM retains its contents even when the power is turned off. Also like other types of ROM, EEPROM is not as fast as RAM. EEPROM is similar to flash memory (sometimes called flash EEPROM). The principal difference is that EEPROM requires data to be written or erased one byte at a time whereas flash memory allows data to be written or erased in blocks. This makes flash memory faster.
- **FRAM**: Short for Ferroelectric Random Access Memory, a type of non−volatile memory developed by Ramtron International Corporation. FRAM combines the access speed of DRAM and SRAM with the non−volatility of ROM. Because of its high speed, it is replacing EEPROM in many devices. The term FRAM itself is a trademark of Ramtron.
- **NVRAM**: Abbreviation of Non−Volatile Random Access Memory, a type of memory that retains its contents when power is turned off. One type of NVRAM is SRAM that is made non−volatile by connecting it to a constant power source such as a battery. Another type of NVRAM uses EEPROM chips to save its contents when power is turned off. In this case, NVRAM is composed of a combination of SRAM and EEPROM chips.
- **Bubble Memory**: A type of non−volatile memory composed of a thin layer of material that can be easily magnetized in only one direction. When a magnetic field is applied to circular area of this

substance that is not magnetized in the same direction, the area is reduced to a smaller circle, or bubble. It was once widely believed that bubble memory would become one of the leading memory technologies, but these promises have not been fulfilled. Other non−volatile memory types, such as EEPROM, are both faster and less expensive than bubble memory.
- **Flash Memory**: A special type of EEPROM that can be erased and reprogrammed in blocks instead of one byte at a time. Many modern PCs have their BIOS stored on a flash memory chip so that it can easily be updated if necessary. Such a BIOS is sometimes called a flash BIOS. Flash memory is also popular in modems because it enables the modem manufacturer to support new protocols as they become standardized.

6.2 List of EEPROM Burner manufacturers

For a list of **EPROM burner manufacturers** visit the Yahoo site and go to economy−>company−>Hardware−>Peripherals−>Device programmers.

- Yahoo URL for EPROMs is at http://dir.yahoo.com/Business_and_Economy/Companies/Computers/Hardware/Peripherals/Device_Programmers/

- Advanced Research Technology B.V − development, production and sales of electronic programmer equipment; development of hardware and software.
- Advin Systems Inc. − PC−based device programmers that support the latest in package types and device technologies.
- Andromeda Research Labs − manufactures a portable eprom and device programming system.
- B and C Microsystems, Inc − offers test and duplication/programming equipment for PCMCIA (PC) Cards, ISA/PCI Cards, SIMMs, Memory Devices (including FLASH), PLDs.
- BP Microsystems − Device Programmers.
- Bytek − designs, develops, manufactures and markets micro−processor−based, modular electronic systems used to program and test semiconductor devices. Product line includes the ChipBurner.
- Concentrated Programming Ltd − offers a full range of device programming solutions.
- Dataman Programmmers Ltd. − manufacture of hand−help EPROM programmer/emulator. Also sell PC−based programmers, and Gang−Pro programmers.
- General Device Instruments − IC Device programmers. Universal and Gang programmers for Pld, Flash, microcontrollers, Proms, EEproms, Memory, Epld, Mach and many other ic devices.
- HI−LO System Research Co., Ltd. − manufacturer of universal and gang device programmers.
- ICE Technology − EPROM and universal device programmers which support memories, microcontrollers, and programmable logic devices.
- Iceprom − in−circuit erasable programmable read−only memory.
- Incept Ltd.
- International Microsystems Inc − High speed reliable gang programmer. (PROM, FLASH, Microcontroller, PCMCIA memory card).
- JED Microprocessors Pty. Ltd. − plugs into a PC printer port D25 connector, and programs any 28−pin or 32−pin EPROM and FLASH device.
- Logical Devices, Inc − device programming for PLDs, FPGAs, PROMs, microcontrollers. Producers of CUPL compiler for programmable logic and the ALLPRO and Chipmaster device programmer.
- MCL Systems − new method not only for programming but also for developing your new hardware with Integrated Controller Unit. And you don't need to be an expert.
- MQP Electronics − manufacturer of universal device programmers, gang programmers, production software, and package converters. High thoughput and reliability.
- Needham's Electronics − manufacturer of device programmers.
- NP Programming Services − provides programming for memory and logic parts.

- Program Automation, Inc. − independent service company specializing in high volume PROM programming, including flash I/Cs.
- Stag Programmers Inc − manufacturer of prom and logic programmers, production handling equipment and UV erasers.
- Sunrise Electronics − universal device programmers, gang and in−circuit programmers with life time support.
- System General Co. − Device Programmer, EPROM Writer and IC Tester
- Tribal Microsystems − universal and gang device programmers, 8051 and EPROM emulators, test and burn−in sockets and production sockets.
- Universal Device Programmers

---

7. Introduction to Network Booting and Etherboot

This chapter is written by Ken Yap ken.yap@acm.org and explains how to bootstrap your computer from a program stored in non−volatile memory without accessing your hard disk. It is an ideal technique for maintaining and configuring a farm of linux boxes.

7.1 What is Network booting?

Network booting is an old idea. The central idea is that the computer has some bootstrap code in non−volatile memory, e.g. a ROM chip, that will allow it to contact a server and obtain system files over a network link.

7.2 How does it work

In order to boot over the network, the computer must get

1. an identity
2. an operating system image and
3. usually, a working filesystem.

Consider a diskless computer (DC) that has a network boot ROM. It may be one of several identical DCs. How can we distinguish this computer from others? There is one piece of information that is unique to that computer (actually its network adapter) and that is its Ethernet address. Every Ethernet adapter in the world has an unique 48 bit Ethernet address because every Ethernet hardware manufacturer has been assigned blocks of addresses. By convention these addresses are written as hex digits with colons separating each group of two digits, for example − **00:60:08:C7:A3:D8** .

The protocols used for obtaining an IP address, given an Ethernet address, are called **Boot Protocol (BOOTP)** and **Dynamic Host Configuration Protocol (DHCP)**. DHCP is an evolution of BOOTP. In our discussion, unless otherwise stated, anything that applies to BOOTP also applies to DHCP. (Actually it's a small lie that BOOTP and DHCP only translate Ethernet addresses. In their foresight, the designers made provision for BOOTP and DHCP to work with any kind of hardware address. But Ethernet is what most people will be using.)

An example of a BOOTP exchange goes like this:

DC: Hello, my hardware address is **00:60:08:C7:A3:D8**, please give me my IP address.

BOOTP server: (Looks up address in database.) Your name is aldebaran, your IP address is 192.168.1.100, your server is 192.168.1.1, the file you are supposed to boot from is /tftpboot/vmlinux.nb (and a few other pieces of information).

You may wonder how the DC found the address of the BOOTP server in the first place. The answer is that it didn't. The BOOTP request was broadcast on the local network and any BOOTP server that can answer the request will.

After obtaining an IP address, the DC must download an operating system image and execute it. Another Internet protocol is used here, called **Trivial File Transfer Protocol (TFTP)**. TFTP is like a cut−down version of FTP−−−there is no authentication, and it runs over User Datagram Protocol (UDP) instead of Transmission Control Protocol (TCP). UDP was chosen instead of TCP for simplicity. The implementation of UDP on the DC can be small so the code is easy to fit on a ROM. Because UDP is a block oriented, as opposed to a stream oriented, protocol, the transfer goes block by block, like this:

```
DC: Give me block 1 of /tftpboot/vmlinux.nb.
TFTP server: Here it is.
DC: Give me block 2.
```

and so on, until the whole file is transferred. Handshaking is a simply acknowledge each block scheme, and packet loss is handled by retransmit on timeout. When all blocks have been received, the network boot ROM hands control to the operating system image at the entry point.

Finally, in order to run an operating system, a root filesystem must be provided. The protocol used by Linux and other Unixes is normally **Network File System (NFS)**, although other choices are possible. In this case the code does not have to reside in the ROM but can be part of the operating system we just downloaded. However the operating system must be capable of running with a root filesystem that is a NFS, instead of a real disk. Linux has the required configuration variables to build a version that can do so.

7.3 Netbooting in Practice

Net Loader is a small program that runs as a BIOS extension, usually on an EPROM on the NIC. It handles the BOOTP query and TFTP loading and then transfers control to the loaded image. It uses TCP/IP protocols but the loaded image doesn't have to be Linux. The loaded image can be anything, even DOS. They can also be loaded from a floppy for testing and for temporary setups.

Besides commercial boot ROMs, there are **TWO** sources for free packages for network booting. Free implementations of TCP/IP net loaders are −

1. **ETHERBOOT** http://www.slug.org.au/etherboot/ and and at mirror−site and at google−site
2. **NETBOOT** http://www.han.de/~gero/netboot.html

Etherboot uses built−in drivers while Netboot uses Packet drivers. First you have to ascertain that your network card is supported by Etherboot or Netboot. Eventually you have to find a person who is willing to put the code on an EPROM (Erasable Programmable Read Only Memory) for you but in the beginning you can do **network booting from a floppy**.

To create a boot floppy, a special boot block is provided in the distribution. This small 512 byte program loads the disk blocks following it on the floppy into memory and starts execution. Thus to make a boot floppy, one has only to concatenate the boot block with the Etherboot binary containing the driver for one's network card like this:

```
# cat floppyload.bin 3c509.lzrom > /dev/fd0
```

Get the nfsboot package (the package is available from your favourite linux mirror site in the /pub/Linux/system/Linux−boot directory). It contains a booteprom image for the network cards (like wd8013) which can be directly burned in. See also the LTSP site at http://www.ltsp.org

Before you put in the network boot floppy, you have to set up three services on Linux −

1. BOOTP (or DHCP)
2. TFTP and
3. NFS.

You don't have to set up all three at once, you can do them step by step, making sure each step works before going on to the next.

Bootp

Install Bootp. See bootp*.rpm on Redhat linux cdrom. See also LTSP site for RPM packages at http://www.ltsp.org. See also unix manual pages 'man 5 bootptab', 'man 8 bootpd', 'man 8 bootpef', 'man 8 bootptest'. You then have to ensure that this server is waiting for bootp requests. The daemon can be run either directly by issuing command

```
bootpd −s
```

Or by using inetd edit the file /etc/inetd.conf and put a line like this:

```
bootps dgram   udp     wait    root    /usr/sbin/in.bootpd    bootpd
```

Insert or uncomment the following two lines in /etc/services:

```
bootps          67/tcp          # BOOTP server
tftp            69/udp          # TFTP server
```

If you had to modify /etc/inetd.conf, then you need to restart inetd by sending the process a HUP signal.

```
kill −HUP <process id of inetd>.
```

Next, you need to give bootp a database to map Ethernet addresses to IP addresses. This database is in /etc/bootptab. You must modify it by inserting the IP addresses of your gateway, dns server, and the ethernet address(es) of your diskless machine(s). It contains lines of the following form:

```
aldebaran.foo.com:ha=006008C7A3D8:ip=192.168.1.100:bf=/tftpboot/vmlinuz.nb
```

Other information can be specified but we will start simple.

Another example of /etc/bootptab is :

```
global.prof:\
        :sm=255.255.255.0:\
        :ds=192.168.1.5:\
        :gw=192.168.1.19:\
        :ht=ethernet:\
        :bf=linux:
machine1:hd=/export/root/machine1:tc=global.prof:ha=0000c0863d7a:ip=192.168.1.140:
machine2:hd=/export/root/machine2:tc=global.prof:ha=0800110244e1:ip=192.168.1.141:
machine3:hd=/export/root/machine3:tc=global.prof:ha=0800110244de:ip=192.168.1.142:
```

global.prof is a general template for host entries, where

- sm field contains the subnet mask
- ds field contains the address of the Domain Name Server
- gw field contains the default gateway address
- ht field contains the lan media hardware type
- bf field contains the name of the boot file

After this, every machine must have a line:

- the first field contains the host name,
- hd field contains the directory of the bootfile,
- the global template can be included with the tc field,
- ha field contains the hardvare address of the ethernet card,
- ip field contains the assigned ip address.

Now boot the DC with the floppy and it should detect your Ethernet card and broadcast a BOOTP request. If all goes well, the server should respond to the DC with the information required. Since /tftpboot/vmlinux.nb doesn't exist yet, it will fail when it tries to load the file. Now you need to compile a special kernel, one that has the option for mounting the root filesystem from NFS turned on. You also need to enable the option to get the IP address of the kernel from the original BOOTP reply. You also need to compile the Linux driver for your network adapter into the kernel instead of loading it as a module. It is possible to download an initial ramdisk so that module loading works but this is something you can do later.

You cannot install the zImage resulting from the kernel compilation directly. It has to be turned into a tagged image. A tagged image is a normal kernel image with a special header that tells the network bootloader where the bytes go in memory and at what address to start the program. You use a program called mknbi−linux to create this tagged image. This utility can be found in the Etherboot distribution. After you have generated the image, put it in the /tftpboot directory under the name specified in /etc/bootptab. Make sure to make this file world readable because the tftp server does not have special privileges.

Tftp

For TFTP, see tftp*.rpm on Redhat Linux cdrom. TFTP (Trivial File Transfer Protocol) is a file transfer protocol, such as ftp, but it's much simpler to help coding it in EPROMs. TFTP can be used in two ways:

- **Simple tftp:** means that the client can acces to your whole file system. It's simpler but it's a big security hole (anyone can get your password file via tftp).
- **Secure tftp:** the tftp server uses a chroot.2 system call to change it's own root directory. Anything outside the new root directory will be completely inaccessible. Because of the chroot dir becomes the new root dir, the hd filed in the bootptab must reflect the new situation. For example: when using insecure tftp, the hd field contains the full path to the boot directory: /export/root/machine1. When using secure tftp whith /export as root dir, then /export becomes / and the hd field must be /root/machine1.

Tftpd is normally started up from inetd with a line like this in /etc/inetd.conf.

```
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd -s /tftpboot
#tftp   dgram   udp     wait    root    /usr/sbin/in.tftpd     tftpd /export
```

Again, restart inetd with a HUP signal and you can retry the boot and this time it should download the kernel image and start it. You will find that the boot will continue until the point where it tries to mount a root filesystem. At this point you must configure and export NFS partitions to proceed.

NFS root filesystem

For various reasons, it's not a good idea to use the root filesystem of the server as the root filesystem of the DCs. One is simply that there are various configuration files there and the DC will get the wrong information that way. Another is security. It's dangerous to allow write access (and write access is needed for the root filesystem, for various reasons) to your server's root. However the good news is that a root filesystem for the DC is not very large, only about 30 MB and a lot of this can be shared between multiple DCs.

Ideally, to construct a root filesystem, you have to know what files your operating system distribution is expecting to see there. Critical to booting are device files, files in /sbin and /etc. You can bypass a lot of the hard work by making a copy of an existing root filesystem and modifying some files for the DC. In the Etherboot distribution, there is a tutorial and links to a couple of shell scripts that will create such a DC root filesystem from an existing server root filesystem. There are also troubleshooting tips in the Etherboot documentation as this is often the trickiest part of the setup.

The customised Linux kernel for the DC expects to see the root filesystem at /tftpboot/(IP address of the DC), for example: /tftpboot/192.168.1.100 in the case above. This can be changed when configuring the kernel, if desired.

Now create or edit /etc/exports (see 'man 5 exports' and 'man 8 exportfs') on the server and put in a line of the following form:

```
/tftpboot/192.168.1.100 aldebaran.foo.com(rw,no_root_squash)
```

The rw access is needed for various system services. The no_root_squash attribute prevents the NFS system from mapping root's ID to another one. If this is not specified, then various daemons and loggers will be unhappy.

Start or restart the NFS services (rpc.portmap and rpc.mountd) and retry the diskless boot. If you are successful, the kernel should be able to mount a root filesystem and boot all the way to a login prompt. Most likely, you will find several things misconfigured. Most Linux distributions are oriented towards disked operation and require a little modification to suit diskless booting. The most common failing is reliance on files under /usr during the boot process, which is normally imported from a server late in the boot process. Two possible solutions are −

1. Provide the few required files under a small /usr directory on the root filesystem, which will then be overlaid when /usr is imported, and

2. Modify the paths to look for the files in the root filesystem. The files to edit are under /tftpboot/192.168.1.100 (remember, this is the root directory of the DC).

You may wish to mount other directories from the server, such as /usr (which can be exported read−only).

Burn EPROM

When you are satisfied that you can boot over the network without any problems, you may wish to put the code on an EPROM.

7.4 Uses of Network booting

X−terminals are one natural use of network booting. The lack of a disk in the terminal makes it quieter and contributes to a pleasant working environment. The machine should ideally have 16MB of memory or more and the best video card you can find for it. This is an ideal use for a high−end 486 or low−end Pentium that has been obsoleted by hardware advances. Other people have used network booting for clusters of machines where the usage is light on the DC and does not warrant a disk, e.g. a cluster of classroom machines.

7.5 For more information

Your first stop should be the Etherboot home page: http://www.slug.org.au/etherboot/ and at mirror−site and at google−site

There you will find links to other resources, including a mailing list you can subscribe to, where problems and solutions are discussed.

Related documents

- NFS−root Mini Howto at /usr/doc/HOWTO/mini or on Linux cdrom.
- Linux Networking−HOWTO by Terry Dawson, at /usr/doc/HOWTO or on linux cdrom 94004531@postoffice.csu.edu.au
- NET−3−Howto at /usr/doc/HOWTO or on Linux cdrom.
- /usr/src/linux/README about configuring and compiling new kernels

8. Redhat Linux configuration

The DC requests to mount /tftpboot/< *IP address of DC* > (in Linux Kernel 2.1 and above it is − /tftpboot/< *name of DC in bootptab* >) as its root directory '/' by NFS from server. You must export this from the server (rw, no_root_squash) because the DC wants to write on it (log files, etc).

The root directory / must contain /sbin, /bin, /lib, /etc, /var, /tmp, /root, /dev and /proc.

/sbin, /bin, /lib can be a copy of an existing Redhat Linux system. They can be shared between all DCs. But hard links only. By the way, don't link to server originals.

/etc, /var and /dev should be non−sharable copies. Customise /etc/sysconfig/network, /etc/sysconfig/network−scripts/ifcfg−eth0, /etc/fstab, /etc/conf.modules, and others. Turn off all network services you don't need. Remove all stuff you don't need from /var, e.g. RPM db, lpd files.

/root and /proc should just exist. /tmp should exist and be mode 1777.

You probably want to create /usr and /home mount points. /usr can be mounted ro (read−only).

About 10 MB per DC plus about 15 MB of shared files should be sufficient. By the way, if your DCs are quite similar, the kernel image can also be shared.

Here is an illustrative script to create the first root filesystem.

```
#!/bin/sh
if [ $# != 1 ]
then
        echo Usage: $0 client-IP-addr
        exit 1
fi

cd /

umask 022

mkdir -p /tftpboot/$1

# just make these ones
for d in home mnt proc tmp usr
do
        mkdir /tftpboot/$1/$d
        done

        chmod 1777 /tftpboot/$1/tmp

        touch /tftpboot/$1/fastboot
        chattr +i /tftpboot/$1/fastboot

        # copy these ones
        cp -a bin lib sbin dev etc root var /tftpboot/$1

cat <<EOF
Now, in /tftpboot/$1/etc, edit

                sysconfig/network
```

```
                sysconfig/network-scripts/ifcfg-eth0
                fstab
                conf.modules

and configure

                rc.d/rc3.d
EOF
```

Here is an illustrative script to duplicate the root filesystem

---

```
#!/bin/sh
if [ $# != 2 ]
then
        echo Usage: $0 olddir newdir
        exit 1
fi

cd /tftpboot

if [ ! -d $1 ]
then
        echo $1 is not a directory
        exit 1
fi

umask 022

mkdir -p $2

# just make these ones
for d in home mnt proc tmp usr
do
        mkdir $2/$d
done

chmod 1777 $2/tmp

touch $2/fastboot
chattr +i $2/fastboot

# link these ones
for d in bin lib sbin
do
        (cd $1; find $d -print | cpio -pl ../$2)
done

# copy these ones
for d in dev etc root var
do
        cp -a $1/$d $2
done

cat <<EOF
Now, in /tftpboot/$2/etc, edit

        sysconfig/network
        sysconfig/network-scripts/ifcfg-eth0
        fstab (maybe)
```

```
        conf.modules (maybe)

and configure

        rc.d/rc3.d
EOF
```

8.1 X−terminal

On the server, make sure the DC is matched by a clause in /etc/X11/xdm/Xaccess and comment out the :0 in /etc/X11/xdm/Xservers. Then make sure that xdm is run from the init scripts.

On the client, run X −query server

You will get the xdm login box and then all your X clients will run on the server.

For other applications use − you could use diskless technique for netboot routers, print servers (but should not be spooling print server), standalone apps, etc.

9. LanWorks BootWare PROMs

This information may save you time. In order to make LanWorks BootWare(tm) PROMs to correctly start up a Linux kernel image, the "bootsector" part of the image must be modified so as to enable the boot prom to jump right into the image start address. The net−bootable image format created by netboot/etherboot's `mknbi−linux' tool differs and will not run if used with BootWare PROMs.

A modified bootsector together with a Makefile to create a BootWare−bootable image after kernel compilation can be found at −

- Bwimage package ftp://ftp.ipp.mpg.de/pub/ipp/wls/linux/bwimage−0.1.tgz
- See also http://www.patoche.org/LTT/net/00000096.html
- LanWorks BootWare Boot ROMs http://www.3com.com/lanworks

Refer to the README file for installation details. Currently, only "zImage"−type kernels are supported. Unfortunately, kernel parameters are ignored.

This section courtesy of Jochen Kmietsch email to − jochen.kmietsch@tu−clausthal.de for any questions.

10. Etherboot

Etherboot is a package for creating ROM images that can download code over the network to be executed on an x86 computer. Typically the computer is diskless and the code is Linux, but these are not the only possibilities.

This document is at the Etherboot Home Page and at mirror−site and at google−site This document explains how to install, configure and use the Etherboot package.

11. Netboot

Netboot was written by Zurück zu Gero. The main site is at http://www.han.de/~gero/netboot.html.

11.1 Introduction

The following list shows just a few examples of what Netboot can be used for:

- Printer spooler
- Terminal server
- X11 terminal
- Data logging system
- Network−Computer (NC)
- Some more ....

For the bootrom to find the kernel image it uses the BOOTP protocol as defined in RFCs and RFCs to get the necessary boot information, and then loads the actual image using the TFTP protocol as defined in RFCs .

The exact specifications for this netboot process can be found http://www.han.de/~gero/netboot/english/spec.html.

11.2 Mailing list

There exists a mailing list devoted to network booting. To subscribe simply send a mail with the line

subscribe netboot

in it's body to majordomo@baghira.han.de

The subject in the mail header doesn't matter. After subscribing to it, you can send messages into the list by writing a mail to netboot@baghira.han.de.

11.3 Netboot useful links

Netboot mailing list archive is at http://www.han.de/~gero/netboot/archive/maillist.html

- 3com drivers at http://support.3com.com/infodeli/tools/nic
- Accton drivers at here
- Artisoft
- CNET
- Compaq
- D−Link
- Microdyne
- Many NE2000 PCI cards are based on Realtek chipsets. Get drivers here
- Standard Microsystems Corp
- Surecom
- Thomas Conrad corp
- Winbond
- Xircom

- Webopaedia page on network cards
- Jargon's driver page with many drivers for older network cards.
- Etherboot and at mirror−site and at google−site This is a project similar to Netbot but based on the BSD bootrom code.
- How to make an X Window Terminal out of your old or outdated PC.
- List of jumper settings for various network cards. This page also contains many other good links.
- Freefire is the home page of the Freefire project, which lists many resources for network security issues.

---

12. Related URLs

- See 'Diskless−root−NFS−HOWTO' at http://metalab.unc.edu/LDP/HOWTO/Diskless−root−NFS−HOWTO.html
- Linux goodies http://www.aldev.8m.com and mirrors at webjump, angelfire, geocities, virtualave, bizland, theglobe, spree, infoseek, bcity, 50megs

---

13. Copyright Notice

---

14. Other Formats of this Document

This document is published in 11 different formats namely − DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML, RTF(Rich Text Format), Plain−text, Unix man pages and SGML.

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/other−formats/ and http://www.linuxdoc.org/docs.html#howto
- Plain text format is in: ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto
- Single HTML file format is in: http://www.linuxdoc.org/docs.html#howto
- Translations to other languages like French, German, Spanish, Chinese, Japanese are in ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML−Tools" which can be got from − http://www.sgmltools.org Compiling the source you will get the following commands like

- sgml2html CVS−HOWTO.sgml (to generate html file)
- sgml2rtf CVS−HOWTO.sgml (to generate RTF file)
- sgml2latex CVS−HOWTO.sgml (to generate latex file)

LaTeX documents may be converted into PDF files simply by producing a Postscript output using **sgml2latex** (and dvips) and running the output through the Acrobat **distill** (http://www.adobe.com) command as follows:

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
bash$ man dvips
bash$ dvips −o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

Or you can use Ghostscript command **ps2pdf**. ps2pdf is a work−alike for nearly all the functionality of Adobe's Acrobat Distiller product: it converts PostScript files to Portable Document Format (PDF) files. **ps2pdf** is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called **pdfwrite**. In order to use ps2pdf, the pdfwrite device must be included in the makefile when Ghostscript was compiled; see the documentation on building Ghostscript for details.

This howto document is located at −

- http://sunsite.unc.edu/LDP/HOWTO/CVS−HOWTO.html

Also you can find this document at the following mirrors sites −

- http://www.caldera.com/LDP/HOWTO/CVS−HOWTO.html
- http://www.WGS.com/LDP/HOWTO/CVS−HOWTO.html
- http://www.cc.gatech.edu/linux/LDP/HOWTO/CVS−HOWTO.html
- http://www.redhat.com/linux−info/ldp/HOWTO/CVS−HOWTO.html
- Other mirror sites near you (network−address−wise) can be found at http://sunsite.unc.edu/LDP/hmirrors.html select a site and go to directory /LDP/HOWTO/CVS−HOWTO.html

In order to view the document in dvi format, use the xdvi program. The xdvi program is located in tetex−xdvi*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Publishing | TeX menu buttons. To read dvi document give the command −

```
xdvi −geometry 80x90 howto.dvi
man xdvi
```

And resize the window with mouse. To navigate use Arrow keys, Page Up, Page Down keys, also you can use 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' letter keys to move up, down, center, next page, previous page etc. To turn off expert menu press 'x'.

You can read postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in ghostscript*.rpm package and gv program is in gv*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Graphics menu buttons. The gv program is much more user friendly than ghostscript. Also ghostscript and gv are available on other platforms like OS/2, Windows 95 and NT, you view this document even on those platforms.

- Get ghostscript for Windows 95, OS/2, and for all OSes from http://www.cs.wisc.edu/~ghost

To read postscript document give the command −

```
gv howto.ps
ghostscript howto.ps
```

You can read HTML format document using Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser or any of the 10 other web browsers.

You can read the latex, LyX output using LyX a X−Windows front end to latex.

15. Topics for Academics and Universities

This section is for academic interest only − for universities or research institutes. If you have plenty of time then you can read it. These links are to RFCs and to the history of diskless nodes. Students will find these links interesting to read the history of development of diskless workstations.

Word of Caution: The information and data given by these URLs may be old.

- Install Instructions at http://www.aldev.8m.com/disklesshowto−install.html Mirror : http://aldev.webjump.com/disklesshowto−install.html
- Troubleshoot Problems http://www.aldev.8m.com/disklesshowto−RFC−951.html Mirror : http://aldev.webjump.com/disklesshowto−RFC−951.html
- RFC 951 http://www.aldev.8m.com/disklesshowto−RFC−1350.html Mirror : http://aldev.webjump.com/disklesshowto−RFC−1350.html
- RFC 1533 http://www.aldev.8m.com/disklesshowto−RFC−1533.html Mirror : http://aldev.webjump.com/disklesshowto−RFC−1533.html
- RFC 1350 http://www.aldev.8m.com/disklesshowto−Troubleshoot.html Mirror : http://aldev.webjump.com/disklesshowto−Troubleshoot.html