Linux Remote-Boot mini-HOWTO: Configuring Remote-Boot Workstations with Red-Hat Linux, DOS, Windows 3.1, Windows 95 and Windows NT

Marc Vuilleumier Stückelberg, David Clerc and Sandro Viale

v3.10, June 1998

This document describes how to set up a very robust and secure server-based configuration for a cluster of PCs, allowing each client to choose at boot-time which operating system to run. The key of this configuration is a bootprom based program, which let the user choose at boot time one of several boot images. This configuration is applicable using InCom TCP/IP Bootprom or any PXE-compliant bootprom (as in NetPC). The most up-to-date version of this document, with hypertext links to downloadable software and other related materials, can be found at the address `http://cuiwww.unige.ch/info/pc/remote-boot/howto.html`. Linuxdoc-SGML, DVI and postscript versions are available in the same directory.

Contents

1 Disclaimer and Copyrights 2

2 What has changed... 3

2.1 ...since version 2.x ? 3

2.2 ...since version 3.0 ? 3

3 Introduction 3

3.1 The Network 4

3.2 How it Works 4

3.3 Related non-commercial documentations 5

4 The Configuration How-To 5

4.1 Setting Up the Boot Process 6

4.1.1 Discovering BpBatch 7

4.2 Setting Up Linux 7

4.2.1 Configuring the Client 7

4.2.2 Testing the Configuration 8

4.2.3 Building the Disk Image 8

4.2.4 System Maintenance and Upgrades 9

4.3 Setting up DOS 6 and Windows 3.1 9

4.3.1 Building the Disk Image 10

4.3.2 Adapting the configuration for other machines 10

4.3.3 System Maintenance and Upgrades 11

4.4 Setting up Windows 95 11

4.4.1 Setting up a Stand-Alone Client 11

4.4.2 Building the Disk Image 12

4.4.3 Adapting the configuration for other Machines 12

4.4.4 System Maintenance and Upgrades 13

4.5 Setting up Windows NT 14

4.5.1 Building the Disk Image 14

4.5.2 System Maintenance and Upgrades 14

4.6 Troubleshooting 15

5 Remote-Boot Tools Reference Manual **16**

5.1 BpBatch, MrBatch and MrZip 16

5.1.1 Command Line Arguments 17

5.1.2 Syntax rules 17

5.1.3 The Cache Filesystem 19

5.1.4 Special variables 20

5.1.5 Monitoring commands 22

5.1.6 Control commands 23

5.1.7 Keyboard-related commands 24

5.1.8 Text output commands 24

5.1.9 Graphics output commands 25

5.1.10 Security-related commands 25

5.1.11 Disk-related commands 26

5.1.12 Boot commands 27

5.1.13 National language support 28

5.1.14 Commands specific to MrZip 28

5.2 NoBreak.sys 30

6 Special TFTP Servers **30**

6.1 InCom's Enhanced TFTP Server 30

6.2 The Security Gateway 30

6.3 The Broadcast TFTP Server 31

7 Discussion **32**

7.1 Bootproms and Hard Disks 32

7.2 Which Bootprom ? 32

1 Disclaimer and Copyrights

This document and the related software are provided as is to the Linux and Internet community, with no form of warranty. Please note that **some operations related in this document may destroy the content of your hard-disk**. We assume no liability for any use, correct or not, of this document and of the related software.

You are free to do anything you want with the remote-boot tools as long as you do not make money by selling them or by distributing them with a commercial product. If you want to commercialize a product derived from these tools, please contact the authors first to make a commercial agreement. These remote-boot tools will remain available for free forever, but we may authorize derived commercial tools.

These provisions shall be interpreted under and in accordance with the laws of Switzerland, canton of Geneva. All disputes, defenses, controversies or claims arrising in conncetion with this document and the related software, shall be subject to the exclusive juridiction of the courts of the canton of Geneva, Switzerland.

If you like this program, you can send us a Postcard and/or make a gift to the International Committee of the Red Cross (ICRC) or to the UNICEF.

2 What has changed...

2.1 ...since version 2.x ?

To say it frankly, almost everything. The underlying concepts are the same, but the software part has been completly redesigned to overcome the limitations of previous versions and to make it easier to use. An highlight of the new features :

- All functions (bpmenu, bpclean, bpunzip) are encompassed in a single program.
- The program can run not only from the bootprom, but also under DOS, Windows 95 and Linux.
- The program can now restore images of FAT16, FAT32 and EXT2FS partitions. If someone want to write NTFS support, let me know... For now, NT users still have to stick to FAT16.
- The program can not only restore disk images but also add and patch individual files in order to customize the client behaviour.
- Disk images are not any more bound to 87 MB. They are now file-system independant archives.
- We provide a mean for automatically downloading a disk image to an arbitrary big number of clients at the same time (broadcast).
- You can now write your own secure boot script, that will determine the behaviour of the machine before the real boot.
- You can now boot any Linux kernel, without applying any patch. Its is also possible to provide a command line and a ramdisk image.
- You can authenticate users at boot time using a Unix, NT or Radius server and deny them any access to the machine.
- Full national language support is included.
- And many, many other new features...

Is there a program for converting old archives to the new format ?

No, because the internal format is radically different. But you can easily do the conversion by yourself:

1. Boot an old image (unzip it to your disk)
2. Remove calls to the old `unzipreg` utility and replace them by the adequate `patch` commands (it is very easy, see the detailed instructions below)
3. Run the new `mrzip` program to create a new-style disk image

2.2 ...since version 3.0 ?

Version 3.0 was the beta-release. A dozen of sites around the world have tested it during a month and given much of their time to help us finding bugs and to suggest enhancements. Thanks to all of them for their patience, and in particular to Maciek Uhlig, Dick Velders and Jeff Teeters.

A few minor features have been added since 3.01, such as support for diskless Linux boot (by disabling the cache).

Version 3.10 introduced compatibility with Intel's *Wired for Management* NetPC standard. The tools now work with any PXE-compliant bootprom (as are most on-board bootproms). Thanks to InCom GmbH for giving us the PXE bootprom that permitted this developpment.

3 Introduction

The configuration described here was developped since Summer 1996 at the CUI, University of Geneva. The Computer Science Department uses several servers and a number of PCs, which fall into two classes:

- computers devoted to students
- computers devoted to research and teaching assistants

We developped the current configuration with the following aims:

- Every computer should be able to run under Linux, DOS, Windows 3.1, Windows 95 or Windows NT. One should be able to choose the desired operating system for each session.

- All softwares, including operating systems, should be take from the server, in order to facilitate the installations and upgrades.
- Clients computers should be able to run without any write-access on the server (for security reasons), except for their home directory.
- Client-side configuration should be reduced to its very minimum. Clients should automatically get their IP configuration parameters from the server, and this information should be located in a single file, used for all operating systems.
- Since almost every computer now has a hard-disk, clients should be able to take profit of it for reducing network load and as temporary storage space for the user.
- Users *must* have a login to be able to use any of the computers.
- The login should be the same for all operating system and should let the user access its unique home directory, common to all operating systems.
- Student (and secretary :-) computers should be fully cleaned up at each start. That is, the PC should always look like if it were just installed.
- Every computer has to be protected from virus attacks.

These constraints lead us to base our configuration on bootprom tools. We first developped new tools for the excellent TCP/IP Bootprom from InCom GmbH. Now that a standard for preboot execution environments as finally emerged, we ported the tools so that it now also works for any PXE-compliant bootprom (for more info on Intel *Wired for Management* standard, please read from `http://www.intel.com/managedpc`).

3.1 The Network

The University of Geneva owns a class B domain, subdivided into several subnets. The CUI uses four subnets, among them one is dedicated to students.

Originally, our PCs were concerned about two network protocols: IPX and IP. On the IPX side, we used a single Novell Netware 3 server for sharing software and users files for DOS and Windows. On the IP side, we used a SUN server for sharing software and users partitions for Linux, with NFS.

In our latest configuration, we do not any more use IPX. There is a single Unix server (which could be Linux as well as a SUN), sharing software and user files using NFS for Linux clients and using SMB (NetBIOS) over TCP/IP for Dos and Windows clients. In this way, we have a single home directory used by all operating systems.

3.2 How it Works

1. When a client PC is turned on, it first performs the traditional system checks before the TCP/IP bootprom takes the control.
2. The bootprom issues a BOOTP/DHCP request in order to get its IP configuration parameters.
3. If the server knows the PC issuing the request, it will send back a BOOTP/DHCP reply with informations such as the client's IP address, the default gateway, and which bootdisk image to use.
4. In case of a PXE bootprom, there is one more exchange between the client and the server to get installation parameters.
5. The bootprom then downloads the boot image from the server using the TFTP protocol. The boot image happens to be a small program called `bpbatch`, our boot-time batch file interpreter.
6. The batch interpreter is started. At this time, it is almost alone in the computer memory. There is no operating system loaded, except the preboot execution environment (offered by the BootPROM).
7. The batch interpreter look in the BOOTP/DHCP reply for command-line options, and in particular for the name of the batch to execute.
8. According to the instructions in the batch file, it will for instance:
   (a) Load a national keyboard mapping
   (b) Authenticate the user according to a remote server (Unix, Radius or Windows NT)
   (c) Let the user choose between the available operating systems
   (d) According to the operating system choosen, repartition the hard-disk and quick-format some partitions
   (e) Check if an up-to-date compressed image of the selected OS is present at the end of the disk. If not, it download it using TFTP
   (f) Uncompress the selected OS to the main partition
   (g) If the selected OS is Linux, load a kernel and start it

(h) If the selected OS is DOS or Windows, simply let the computer boot on its fresh new hard-disk

For **DOS and Windows 3.1**, we use the freely available Microsoft LanManager for DOS (search the network for the mirror nearest to you; the distribution consists of three files named `disk1` to `disk4`) as SMB client. Microsoft LanManager supports dynamic configuration using DHCP. After logging in, the user is faced to DOS, and can start Windows 3.1 by typing the traditional `win` command. Note that at this point, DOS and Windows 3.1 appear to be installed locally.

For **Windows 95 and Windows NT**, we also use Microsoft SMB client (called *Client for the Microsoft Network*), that supports dynamic configuration using DHCP. We reduce network load using Shared LAN Cache, a nice and powerful network-to-disk cache program.

Students computers can be turned off *the hard way* at any time without risks, since the hard disk is reinitialized at each start.

For "safe" computers (ie. for assistants computers), once the computer has been booted once using the above described system, the boot script simply redirect the boot to the local hard-disk, without cleaning it again. This allow users to leave data on their local hard disk. But whenever the configuration gets corrupted, the user can simply choose from the boot menu in order to have a fresh installation.

3.3 Related non-commercial documentations

This configuration has been successfully reproduced at several places around the world. A few people have written some hints and tricks that complement this How-To. If you did so and that your page is not already referenced in this documentation, please send an e-mail to `Marc.VuilleumierStuckelberg@cui.unige.ch`. And if you experience problems while reproducing this configuration, have a look at these pages !

- `http://www.katedral.se/system/elevsyst`, by Johan Carlstedt of The Cathedral School of Uppsala, Sweden.
- `http://vitoria.upf.tche.br/~fred/`, in portuguese, by Frederico Goldschmidt of the Passo Fundo University, Brasil.

4 The Configuration How-To

First, arrange to have the following two machines within arm's reach:

- the **server**, for us a Unix machine
- the **client**, a PC with a TCP/IP bootprom enabled, and nothing valuable on the hard disk.

If you want to test the configuration but you do not yet have a bootprom, you can download the TCP/IP BootProm demo diskette from InCom GmbH at `http://www.incom.de`. This diskette will make your computer behave like if it had a TCP/IP Bootprom plugged in.

For student computers, we configured our Bootprom for boot on network first, and disabled hard-disk and floppy-disk boot. For assistant computers, we also configured our Bootprom for network-boot, but we allow hard-disk and floppy-disk boot.

On the server, setup a DHCP daemon. Solaris 2.6 DHCP server is very good, Windows NT DHCP server is quite bad (see the web page at InCom GmbH for a few tricks if you still want to use Windows NT DHCP server). You also need to enable a TFTP daemon. This document assume that you use the extended TFTP daemon provided on your TCP/IP Bootprom Utility disk. If you prefer to use a standard TFTP daemon, remove the `P` in all boot image name extensions, in order to tell the Bootprom to use only the standard TFTP port (This trick was introduced by InCom GmbH for the TCP/IP Bootprom. We still use it as an easy way to select the default TFTP port with PXE bootproms).

If you are using a PXE bootprom, you need in addition a BINL server. There is one available for NT in Intel Wired for Management PDK, but it is not very easy to use. We are currently writing a minimal BINL server for UNIX. Don't worry, expect many Wired for Management tools to appear on the market during the coming months.

Don't forget that BOOTP/DHCP requests are bounded by subnets. If the client and the server do not reside on the same subnet, you should install a gateway on any computer between the two. For now, just assume that both machines are on the same subnet.

First, we will do set up the part common to all operating systems, ie. the batch-file interpreter. Then, for each operating system, we will go through the following steps:

1. Setup a stand-alone client
2. Save its configuration on the server
3. Test it as a remote-boot client
4. Adapt it so that it works for any similar client machine

Once this is done, you will be able to setup any supplemental client just by plugging a BootPROM in it and adding one line in the DHCP configuration file.

Our examples assume that you have a hard disk of 1.4 Gb or more. If you have less, reduce the sizes of the partitions, but remember the you need to leave a few hundreds megabytes unallocated (that is, the last partition must not take up to the last cylinder) to leave free room for the special cache partition. Moreover, as the cache always starts at the cylinder following the last allocated cylinder, if you do not use the same total size for all your tests, you will have to download several times the same files (the cache will be automatically cleared).

Never despair. If you can't get it to work, first look in the *Troubleshooting* section if your problem is not already solved (get the latest version from the Web, it will be updated daily during the Beta stage) and otherwise, send me an Email. If your problem is strictly related with the remote-boot configuration and if I am not overflowed, I will try to solve your problem.

4.1 Setting Up the Boot Process

Get the `BpBatch` software, either as `.zip` or as `.tar.gz`. The executables are available at

- `http://cuiwww.unige.ch/info/pc/remote-boot/soft/bpb-exe.zip`
- `http://cuiwww.unige.ch/info/pc/remote-boot/soft/bpb-exe.tar.gz`

The source code (Assembler and C) is also available on request.

In the server `/tftpboot` directory, put the following three special boot images, which together make our pre-boot batch file interpreter:

- `bpbatch.P`, the dynamic loader (respect the uppercase !)
- `bpbatch.ovl`, the relocated interpreter
- `bpbatch.hlp`, the on-line help file

Then add an entry in the DHCP configuration file for your client, with the boot file set to `"bpbatch.P"`. Define a vendor option tag 155 (decimal) with the value `"-i"`. It is interpreted by `bpbatch` as the command line, and `-i` stands for "interactive".

Boot the client computer. You might shortly see

- The BootPROM copyright
- The string `DHCP` while the client waits for a DHCP reply
- The string `TFTP` while the client waits for the first TFTP packet
- The string `Loading BpBatch` while the loader download the interpreter
- And finaly our banner, followed by a nice *greather-than* prompt

Congratulations ! You have started the batch interpreter... If you are curious about what you can do with it, continue reading the next section. If you are on a hurry, skip it and go directly install the operating system of your choice. If you have any doubt about a command within the interpreter, type `help`.

Note that you can run the same interpreter within DOS and Linux by running the `MrBatch` program. There are a only very few differences (the Linux version do not have graphics support and the DOS version can only send BOOTP and TFTP requests if the BootProm is not hidden by the operating system).

It may be a good idea to read now the section about the *Syntax Rules* of `BpBatch`, and in particular the paragraphs on *File References* and on *The Cache Filesystem*. This will help you understand the examples.

Once all operating systems will be set up, you will have to make a menu to let the user choose the one he wants. You should be able to discover by yourself how to make such a menu. All necessary commands are documented at the end of this document.

4.1.1 Discovering BpBatch

Try to type `LogVars`. You should get about thirty variables listed. Roughly, the first are BpBatch settings, then come all parameters extracted from the BOOTP/DHCP reply, and the last variable is a list of disks sizes, in Megabytes.

Type `GetPartitions part`, then `LogVars` again. There should be one more variable containing the list of defined partitions on your first hard-drive. Assuming that the first partition is either BIGDOS, FAT32 or LINUX-EXT2, try `LogDir "{:1}"` to get the content of the root directory, then `LogDir "{:1}/usr"` if there is an `usr` directory. You can even try `LogTree "{:1}/etc"` to get a directory tree.

Put a GIF file (format GIF-87a, interlaced or not, but NOT GIF-89a) on your TFTP server. We will suppose that the file is named `image.gif`. You can copy it wherever you want with the following command: `Copy "image.gif" "{:1}/temp/image.gif"`. Or you can use it directly from the server. Now type `Logvars "V*"` and look at the value of the `VESA` variable. If it is `On`, which is most probable, that means you have a VESA-compliant video adapter and you will be able to display the image with the following command: `DrawGif "image.gif"`. The image should be on the upper left corner of the screen. You can draw it on another place by specifying X and Y coordinates after the image name. You can also draw text with `DrawText 200 200 "Hello world" yellow`. Or draw an empty window with `DrawWindow 200 200 300 150`. When you are tired of graphic mode, simply type `CloseGraph`.

Now take a text editor, and create a file named `test.bpb` in the `tftpboot` directory with the following content:

```
:again
DrawWindow 150 200 400 160
DrawText 155 204 "Identity check" white
TextAttr Black LightGray
At 15,20 Print "Username : "
Input username 8
At 17,20 Print "Password : "
Getpasswd userpass 8
if "$username" != "smith" goto again
if not "$userpass" match-passwd "BpR8oiIlRR9bo" goto again
#
clear
DrawWindow 200 200 150 100 green
DrawText 205 204 "Congratulations" white
DrawText 220 250 "You got it !" yellow
WaitForKey 3
CloseGraph
interact
```

In your BOOTP/DHCP configuration, change the option-155 from `"-i"` to `"test"`, and reboot the client computer. The small script should run automatically, and ask you for a username and password. If you do not type `smith` and `justdoit`, you wont be able to boot the computer. Later you will learn how to use a Unix, NT or Radius server to check valid user names.

4.2 Setting Up Linux

In order to set up Linux, you will need to boot the floppy disk provided with the RedHat Linux distribution. BpBatch includes a command that can redirect the boot to the floppy: `FloppyBoot`. Unfortunately, it does not seem to work correctly with this special boot floppy (any suggestion is welcome to fix this). Consequently, you will probably need to boot a computer without bootprom for this first part of Linux setup.

Set up RedHat Linux on your client, with network support, and any packages you may want. You may want to recompile the kernel to better fit your hardware, but it is not necessary.

4.2.1 Configuring the Client

It is probably a good idea to include BOOTP support to the kernel, so that you do not have to customize the client IP address manually.

In order to reduce network load, you might also want to setup the `filecache` for caching on the hard disk files that are loaded by NFS. Roughly, the principle of the filecache is that whenever a symbolic link from the `cache` subdirectory is followed, it is replaced by its target. If the target is itself a subdirectory, each entry of the subdirectory becomes a symbolic link to the original entry of the foreign filesystem. The filecache has been written by Unifix GmbH, and is part of Unifix Linux 2.0. It is freely distributable, and you can get the necessary files from `http://cuiwww.unige.ch/info/pc/remote-boot/soft/filecache.tar.gz`. In order to use the filecache, you have to

- apply a patch to the kernel (file `patch-filecache`), enable filecache support through `make config` or whatever you prefer, and recompile the kernel
- copy the filecache binary file to `/sbin`
- create a mount point called `/mnt/nfs` (using `mkdir`)
- copy `filecache.conf` to `/etc`. This file contains the following lines:

  ```
  Max 100 MB 50 % #
  Cache /mnt/nfs/usr /usr
  Cache /mnt/nfs/opt /opt
  ```

- copy the content of `/usr` and `/opt` to the server, export them read-only with `anon=0` (for allowing root access) and mount them under `/mnt/nfs` (add a line for that in `/etc/fstab`)
- rename `/usr` as `/usr.orig`
- link `/usr` to `/mnt/nfs/usr`
- rename `/opt` as `/opt.orig`
- link `/opt` to `/mnt/nfs/opt`
- ensure that `/usr` and `/opt` are not empty and contains the correct directories
- recursively remove `/usr.orig` and `/opt.orig`
- copy `filecache.init` to `/etc/rc.d/init.d`
- And finally link `/etc/rc.d/rc3.d/S35filecache` to `/etc/rc.d/init.d/filecache.init`

If you successfully followed each of these steps, you should have the filecache working next time you boot, as long as you do not forget to use your patched kernel.

4.2.2 Testing the Configuration

Copy your compressed kernel image (`zImage`, `bzImage`, `vmlinuz` or whatever you call it) to the server `/tftpboot` directory as `linux.krn`. If you had to unplug the bootprom from the PC, you can now plug it again. When `BpBatch` starts, type `LinuxBoot "linux.krn" "root=/dev/hda1 BOOT_IMAGE=linux"` (assuming that the root ext2 filesystem is on the first partition). Alternatively, if you did setup your configuration on a computer without bootprom, just boot let it boot using the loader you installed (lilo, ...). But in the later case, if you want the filecache to work, you should have explicitely installed your kernel with filecache support at the right place.

Wait until the system comes up. If you installed the filecache, you can check that `/usr` has exploded into a directory with some symlinks and some already-exploded directories. Now start the programs that the end-users will use most of the time, in order to load them once for all to the hard disk.

You can still make adjustements to your configuration, like on any stand-alone linux station.

4.2.3 Building the Disk Image

When you are happy with your configuration, login as `root`, go to the `/tmp` directory and run our `mrzip` program. `MrZip` is a command interpreter like `BpBatch`, but it can understand more commands than `BpBatch` does. In particular, it can understand the following commands:

```
showlog
filter -"tmp/*"
filter -"var/log/*"
fullzip "/" "/tmp/linux.imz"
```

This will create a disk image in `/tmp/linux.imz`. Move it to the server `/tftpboot` directory. Then copy the following batch file to `/tftpboot/linux.bpb`:

```
hidelog
```

```
setpartitions "linux-ext2:992 linux-swap:32"
fullunzip "linux.imz" 1
clean 2
linuxboot "linux.krn" "root=/dev/hda1 BOOT_IMAGE=linux"
```

The `BOOT_IMAGE` argument is to stay compatible with `lilo` for RedHat 5.1 and later `rc.sysinit`.

Your remote-boot linux configuration is ready ! You can now either set the BOOTP-option-155 to `"linux"`, or type `include "linux.bpb"` from within BpBatch to test it.

4.2.4 System Maintenance and Upgrades

If you want later to upgrade software, install bug fixes and security fixes, proceed as follow:

- Remote-boot a client computer to get a fresh linux install
- Make your changes
- Redo the disk image
- Copy the new image in place of the old one on the server

That means, you can upgrade software on your server-based configuration as if it were a purely local install.

4.3 Setting up DOS 6 and Windows 3.1

On the client computer, boot on your favorite dos floppy disk (either remove the bootprom or type `FloppyBoot` within BpBatch). Format the dos partition of your hard-drive with the `/S` option, in order to put the operating system on it. The size of the partition is not important, as disk archives created with `MrZip` Create a `DOS` subdirectory, copy DOS in it. Install your favorite network client (for instance Microsoft LanManager), Windows 3.1, and so on. If you use Microsoft LanManager, do not use DHCP for the IP configuration as it is a very poor implementation that will almost surely fail with reasonable network load. To do that, add the following lines in your `protocol.ref` file, in the section that loads `tcptsr` (of course, replaces the `xxx` by your true IP parameters):

```
IPADDRESS0 = xxx xxx xxx xxx
SUBNETMASK0 = 255 255 xxx xxx
DEFAULTGATEWAY0 = xxx xxx xxx xxx
DISABLEDHCP = 1
```

Do not be afraid to use EMM386 to optimize the memory usage, and even to include the area where you put your network adapter ROM, since it is not used anymore at this time. But carefully exclude the network adapter RAM, or you will not be able to connect to your server. Use the `NOEMS` parameter.

If you want to ensure that the client machine cannot be used without a valid login name, download our `nobreak` pseudo-device driver (available at `http://cuiwww.unige.ch/info/pc/remote-boot/soft/nobreak.zip`) and run it at the beginning of your `config.sys`. Then add something like this to your `autoexec.bat`:

```
rem -- we use the dummy file c:\logged as a flag
del c:\logged >nul
:loginneeded
cls
echo Please type in your login name and password
echo.
net logon *
rem -- the login script should have created c:\logged
if not exist c:\logged goto loginneeded
del c:\logged
rem -- now enable break again
echo Yes >NOBRK
```

Ensure that your client boot well by rebooting the client and evaluating the following commands within `BpBatch` interactive mode:

```
HideBootprom
HdBoot
```

4.3.1 Building the Disk Image

On the server, make a share called `admin` for instance, on which you will put some stuff for the system administrator. If the server is a Unix machine, it is a good opportunity to put in `admin` a softlink to the `/tftpboot` subdirectory, so that you can put images in it directly from the client. Within `admin`, create a `/utils` subdirectory and put the following files in it:

- `mrbatch.exe`, the DOS version of `BpBatch`
- `mrzip.exe`, the DOS version of the program for building disk images
- `bpbatch.hlp`, the on-line help file

You might also like to put in the same directory a simple MrZip script named `zipdos.mrz` file that contains the commands needed for building a DOS image, like this one:

```
showlog
filter -"lanman.dos/lmuser.ini"
filter -"temp/*"
filter -"*.swp"
fullzip "c:/" "L:/tftpboot/dos.imz"
```

Now go back to your client, mount the `admin` volume on drive `L:`, go to your `utils` directory and type the following command:

```
mrzip -b zipdos
```

One minute later, you will have a new file in the server `/tftpboot` subdirectory called `dos.imz`, which is a compressed image of your hard disk. Copy the following batch file to `/tftpboot/dos.bpb`:

```
hidelog
setpartitions "bigdos:1024"
setbootpart 1
fullunzip "dos.imz" 1
hidebootprom
hdboot :1
```

Your remote-boot DOS configuration is ready ! You can now either set the BOOTP-option-155 to `"dos"`, or type `include "dos.bpb"` from within BpBatch to test it.

4.3.2 Adapting the configuration for other machines

If you want to customize some settings according to the machine, typically the IP settings since Micro$oft DHCP is buggy, you can setup `BpBatch` to change some files before booting. Firsti go to the `lanman.dos` directory and do

```
copy *.ini *.ref
```

Then edit the `.ref` files and replace all fixed parameters with BOOTP variable names as in the following examples:

```
computername = ${BOOTP-Host-Name}
ipaddress0 = ${MS-IPAddress}
subnetmask0 = ${MS-IPSubnet}
defaultgateway = ${MS-IPRouter}
```

Then rebuild the disk image as previously. Note that for IP parameters, we do not use the BOOTP variables directly because LanManager needs then as space-separated numbers instead of dot-separated numbers. Change `dos.bpb` to the following:

```
hidelog
setpartitions "bigdos:1024"
setbootpart 1
fullunzip "dos.imz" 1
set MS-IPAddress="$BOOTP-Your-IP"/.= /
```

```
set MS-IPSubnet="$BOOTP-Subnet-Mask"/.= /
set MS-IPRouter="$BOOTP-Routers"/.= /
patch "{:1}lanman.dos/protocol.ref" "{:1}lanman.dos/protocol.ini"
patch "{:1}lanman.dos/tcpputils.ref" "{:1}lanman.dos/tcputils.ini"
patch "{:1}lanman.dos/lanman.ref" "{:1}lanman.dos/lanman.ini"
hidebootprom
hdboot :1
```

If you prefer, you can also put the `.ref` files in the server `/tftpboot` directory instead of in the disk image.

We like to be able to easily change the computers configuration without rebuilding the image. To do that, copy your `autoexec.bat` and `config.sys` as `autoexec.ref` and `config.ref` to the server `/tftpboot` and add the following two lines to the batch file above:

```
patch "autoexec.ref" "{:1}autoexec.bat"
patch "config.ref" "{:1}config.sys"
```

You can then freely change the files and even customize them with machine-dependant values obtained from BOOTP.

After making any change to the client machine configuration, do not forget to rebuild the disk image using `mrzip` if you want to preserve your changes.

4.3.3 System Maintenance and Upgrades

If you want later to add new software or change anything else, proceed as follow:

- Remote-boot a client computer to get a fresh install
- Make your changes
- Redo the disk image
- Copy the new image in place of the old one on the server

That means, you can upgrade software on your server-based configuration as if it were a purely local install.

4.4 Setting up Windows 95

In previous versions of this document, we used the Microsoft server-based installation of Windows 95, but it was really too much pain and not much worth:

- It is very, very bogus
- Many software package do not support it and their install will fail. Among them, Microsoft Internet Explorer, OnNet 32, Novell's Protected-mode client (which is MUCH more secure than Microsoft Client for Netware).
- It cannot be used with the Microsoft Network client over TCP/IP, since Microsoft provides no real-mode driver for TCP/IP compatibe with Windows 95. That means, it cannot be used with Samba
- It makes software upgrades almost impossible since every client turned on will lock many DLLs on the server, and thus produce *sharing violations* if you try to upgrade them.

Consequently, we throwed away of this document all the informations and bug-workaround collected during months (you can still find them as a HTML document at http://cuiwww.unige.ch/info/pc/remote-boot/win95old/win95old.html) and turned to our new disk-based remote-boot concept. Basically, the configuration for Windows 95 is now almost as easy the configuration for DOS.

4.4.1 Setting up a Stand-Alone Client

Setup a regular Windows 95 client, either starting from scratch as explained in the configuration of a DOS client, starting from the DOS client and installing over the network (that is what we did). You can also start with a preconfigured Windows machine, but you will probably have less knowledge of what stuff is on the hard disk.

Proceed as described above for a DOS client. It is usually NOT necessary to use EMM386 with Windows 95. If you are using Windows 95 OSR2 (alias MSWIN 4.1, alias Windows 95 service pack 1, alias Windows 95 with Internet Explorer), you should add the following line in the

Options

section of MSDOS.SYS (yes, it is a text file):

```
AUTOSCAN=0
```

This will let Windows know that you do not want ScanDisk to be runned automatically at boot time.

If you want to reduce network and server load (which will improve your system performances) while keeping all softwares on the server, you should consider installing the excellent Shared LAN Cache, from Measurement Techniques, Inc (see http://www.lancache.com). This software runs on each client computer, and caches to the local hard disk every data obtained from the network. Even MS-Office starts much faster the second time you run it... You need one license per client computer, but it is not very expensive, and the firm make special prices for universities and colleges. The best thing to do is to go to their Web site and download the free evaluation copy.

4.4.2 Building the Disk Image

Your MrZip script will be named `zipwin95.mrz` and contain:

```
showlog
filter -"temp/*"
filter -"*.swp"
fullzip "c:/" "L:/tftpboot/win95.imz"
```

To build the image, mount the `admin` volume on drive `L:`, go to your `utils` directory and type the following command:

```
mrzip -b zipwin95
```

A few minutes later, you will have a new file if the server `/tftpboot` subdirectory called `win95.imz`, which is a compressed image of your hard disk. If your compressed image was bigger than 87 MB, it has probably been splitted in two or more fragments. These fragments will automatically loaded one after the other when needed. Note that an image bigger than 87 MB will usually take More than one minute to uncompress and may irritate your users. Our Windows 95 image is only 70 MB big, because most software (except Office and Explorer) completely reside on the server. Only 45 seconds are needed to uncompress the image and restore the full disk.

Copy the following batch file to `/tftpboot/win95.bpb`:

```
hidelog
setpartitions "bigdos:1024"
setbootpart 1
fullunzip "win95.imz" 1
hidebootprom
hdboot :1
```

Your remote-boot Windows 95 configuration is ready ! You can now either set the BOOTP-option-155 to `"win95"`, or type `include "win95.bpb"` from within BpBatch to test it.

4.4.3 Adapting the configuration for other Machines

The big difference between Windows 3.1 and Windows 95 is that the later includes code for Plug-and-play , ie. automatic detection of your hardware. This not a bad thing in itself, but the trouble is that it is often too sensible, and that it sometimes fails.

If you try to start another client with exactly the same boot image, you will probably get several messages during startup telling that Windows has detected new hardware: a new sound card, a new hard-disk, a new network card, and even a new mouse... There can be two reasons for that:

- the devices may not use the same ressources (for instance the mouse is not connected on the same port, or the sound card is not connected in the same slot - yes, that is detected)
- the devices may tell to Windows 95 their personal serial number (for instance, every Windows 95 differenciate every network card on the basis of its world-wide unique ethernet address)

The fact that Windows 95 discover that the hardware has changed may not be a problem if the plug-and-play works as-is, but it become a problem when the plug-and-play does not work. For instance, Windows 95 plug-and-play for our Logitech PS2/aux mouse does not work, and result in no mouse at all. To solve such kind of problems, arrange to have all computers as similar as possible, or make different images for different hardware. Later, you will discover that you can simply use the same image and just have several copies of the registery, that you can copy after having restoring the disk image but before booting.

The thing you cannot avoid to differ between computers is the network card. PCI cards usually do not mind, but ISA Plug and Play do. Bad luck for us, the plug-and-play code for our SMC EtherEZ card hangs the computer. The only solution is to let Windows 95 believe that it already know the network card, and that it is not necessary to trigger plug-and-play. The trick for doing that is to automatically insert an entry for the network card in Windows 95 registery, before starting it. Note that this trick is not any more needed with most PCI cards.

Move the `autoexec.bat` to the server as described above for DOS. Edit it (on the server) and add the following lines:

```
rem --- Patch Windows registery in order to avoid plug-and-play detection
regedit /L:c:\windows\system.dat /R:c:\windows\user.dat c:\temp\patch.reg
```

`regedit` is a standard Windows 95 program that let you browse the registery if you start it from within Windows 95, or do simple operations on the registery if you call it from DOS. Run `regedit` under Windows 95, search for your network card, usually under

```
HKEY_LOCAL_MACHINE\Enum\ISAPNP
```

and export the branch using the *File* menu. This will create a text file, that you should same as `patch.ref` in the server `/tftpboot` diretory. Edit this file and find out where the card ethernet address is stored (do that on two different machines and compare the files if you can't find it by yourself). Replace it by a pettern in the form `${MACID}`. Then add lines to the `win95.bpb` script like this:

```
set macid = "$BOOTP-Client-ID"
patch "patch.ref" "{:1}temp/patch.reg"
```

(do any necessary string manipulation for setting `MACID` if it is not exactly the client Ethernet address). That's all, your clients should not any more try to autodect the network card.

Once again, this whole trick is not necessary when using PCI network adapters. Incidentally, we can use the same mechanism for automatically configuring the hostname, which Windows 95 does not seem to take into account when configuring through DHCP. We just add the following line to our `patch.ref` file:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"ComputerName"="${BOOTP-Host-Name}"

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP]
"HostName"="${BOOTP-Host-Name}"

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\ComputerName\ComputerName]
"ComputerName"="${BOOTP-Host-Name}"
```

Using this small registery trick, your configuration should normally be portable for all machines with similar configurations. If you cannot avoid that Windows detect some hardware as new on one machine, try to rebuild the disk image from this machine. This will include the registery configuration specific to this machine into the image, and hopefully supress the problem.

4.4.4 System Maintenance and Upgrades

If you want later to upgrade software, install bug fixes and security fixes, proceed as follow:

- Remote-boot a client computer to get a fresh install
- Make your changes
- Redo the disk image
- Copy the new image in place of the old one on the server

That means, you can upgrade software on your server-based configuration as if it were a purely local install.

4.5 Setting up Windows NT

We do not use Windows NT for remote-boot client computers but we have tested our system to ensure that it work as well. And it works.

As our utilities currently have no support for NTFS (we neither have the documentation nor the time to do that, but I would be happy to help anyone who is interested in doing it), you will have to install NT on FAT16 (simply do not convert your partitions to NTFS during the setup).

Copy your `win95.bpb` boot script to `winnt.bpb`. Change the `setpartitions` line in `winnt.bpb` to the following:

```
setpartitions "BIGDOS:512 BIGDOS:512"
```

Then boot Windows 95 using this script, and install your NT client on drive C. Do not worry about the second partition for now. Do not install too much stuff, or you will get a really large and slow-to-uncompress image. Remove Windows 95 from the disk disk C, you do not need it in a Windows NT image (the boot menu is handled by the bootprom, not by NT boot loader).

Reboot your computer in without overwriting the hard disk, ie. do not execute the `winnt` script but just

```
hidebootprom
hdboot
```

Your NT station should start-up correctly. Make any necessary customization.

4.5.1 Building the Disk Image

The trouble with Windows NT is that direct disk access is prohibed by the kernel. That means, `MrZip` will not even be able to read the boot sectors. The best way to do an image is then to boot Windows 95 and to run `MrZip` from a DOS window. To do that, change the `winnt.bpb` script so that the Windows 95 image is not restored on the first but on the second partition:

```
hidelog
setpartitions "BIGDOS:512 BIGDOS:512"
setbootpart 2
fullunzip "win95.imz" 2
hidebootprom
hdboot :2
```

(if you have any supplementary patch, change the "{:1}" to "{:2}"). Boot with this script; you should have Windows 95 running, but a new drive `D:` should be available, with Windows NT inside.

Make your disk image as usual (but on `D:`, of course), and save it as `winnt.imz` on the server `/tftpboot` directory. Edit one last time the `winnt.bpb` script like this:

```
hidelog
setpartitions "BIGDOS:512 BIGDOS:512"
setbootpart 1
fullunzip "winnt.imz" 1
clean 2
#fullunzip "win95.imz" 2
hidebootprom
hdboot :1
```

Your Windows NT remote-boot configuration is ready. Of course, if you do not like to have two partitions, you can setup a single partition instead. But when you have to rebuild the image, you will have to setup the second partition again for booting Windows 95.

4.5.2 System Maintenance and Upgrades

If you want later to upgrade software, install bug fixes and security fixes, proceed as follow:

- Remote-boot a client computer to get a fresh install

- Make your changes
- Edit `winnt.bpb`: comment the `clean` and winnt `fullunzip`, uncomment win95 `fullunzip`
- Redo the disk image
- Copy the new image in place of the old one on the server

That's all, folks !

4.6 Troubleshooting

This section lists most frequently encountered problems.

The image download never ends
: You are probably using a standard TFTP server, and it cannot handle more than 65535 packets of 512 bytes (or even 32767 packets for the Solaris server). That is, your image must be fragmented in pieces of no more than 30 MB (or 15 MB for Solaris). See under *CopyArchive* for instructions on fragmenting an existing image. But you should seriously thing about using InCom's extended TFTP server, as it is much more efficient (it uses packets of 1408 bytes instead of 512 bytes).

The archive decompression fails immediately
: There are two possibilities. Either the image is really corrupted on the server (try use MrZip to see if it is the case), or the file transfer has failed because of TFTP timeout. TFTP timeout occurs when the network is too heavily loaded (for instance if you try to download a huge image with more than four clients at a time). In this case, `BpBatch` does not retry indefinitely because it would not help. Shut down a few computers and retry with no more than four computers (or maybe even three). If you often need to download images for a lot of computers, you can try our special Broadcast TFTP server (see the section dedicated to it).

The computer hangs instead of downloading/unzipping
: May be your computer has a bad VESA support. Try giving the `-v` command-line argument or setting the VESA variable to `"OFF"`.

VESA scrolling is broken
: We use a VESA 1.1 function for scrolling. If your video adapter does not support VESA 1.1, forget it. If the scrolling works for one page, but then produces a strange strippled pattern, do not worry. This is a known bug, I will fix it as soon as I have time for it (VESA scrolling is not really essential...)

There is a corrupted file in the cache
: When a file in the cache is corrupted by an external program, it is automatically removed from the cache. When a file in the cache is not fully written (because the computer is turned off during the file transfer), it is also automatically removed. But if the server transmits a corrupted file or if the transfer aborts from the server side, it is possible that this file stays in the cache. You can clean-up the cache simply by holding both shift down while `BpBatch` access it for the first time.

The Dos-based file I/O does not work under MrBatch
: Get the latest version, this bug has been fixed.

The EXIT command does not work in a batch file
: This is not a bug. Exit is not a command. There is no exit or quit command because it does not make any sense to exit from a boot script without booting. And MrBatch is really the same program as BpBatch. What you can do instead is calling HdBoot. This makes sense, and the DOS version will cleanly exit instead of rebooting. Note that you can exit from the DOS version at any time by pressing Ctrl-Break. This will restore all hooked interrupts before leaving.

The Print command does not print
: If you try to print something and immediately enter interactive mode, you may not see your text. This is because your text was written on the *runtime* screen and the `Interact` command has switched the display to the *Log* screen. Just put a `GetKey` after the print commands and you will see the text output.

MrZip says `Malloc failed`
: MrZip needs a lot of conventional memory to run. If you encounter this problem, first ensure that you have unloaded the bootprom either using `HideBootprom` or using InCom's `bputil`. If you run MrZip from bare MS-DOS (not within Windows 95 DOS box), you should use EMM386 to load the network drivers high in order to get as much conventional memory as possible. From a Windows 95 DOS box, there is usually no problem (as long as you have not left your old 16-bit stuff in your `autoexec.bat` when you installed Windows 95).

MrZip aborts while reading directories

This bug has already been fixed once. Get the latest release of `MrZip`. If the problem persists, try to build your image with `Trace` set to `"ON"` (and usually `PauseLog` set to `"OFF"`); this will let you discover which file causes the problem. Send a detailled bug report.

MrZip cannot access some file

MrZip is probably trying to read a locked, open or special file, such as Windows swap file. Such files should usually not be included in the image and should be filtered out (using the `filter` command). It is also possible that the operating system is playing you a trick. If MrZip does not tell you what file causes the problem, try to build your image with `Trace` set to `"ON"` (and usually `PauseLog` set to `"OFF"`). You can also try to use direct disk access (that is, do not refer the source partition as `"C:"` or `"/"` but as `"{:1}"` or whatever partition it is). Using direct disk access is usually slower because we have less buffers than the operating system, but it may be sometimes more reliable.

Disk images are always reloaded from the server

Disk images are stored in the special cache area and should not be reloaded if they have not changed on the server. However, as the cache area always starts after the last used partition, changing the total size of partitions will move the location of the cache and thus destroy its content. Another possible reason for a file disappearing from the cache is that the previous file has grown more than one-and-an-half times its initial size. The file would then have been overwritten and need to be downloaded once again. This should almost never occurs. A third possible reason is a too small cache area. If the free space left outside the partitions is less than one-and-an-half times the sum of all compressed image sizes, only the most recently used images will be present in the cache and the other will have to be reloaded on demand.

Red Hat Linux 5.1 does not boot properly

This distribution assumes Linux was booted using `lilo` and checks for the `BOOT_IMAGE` command line argument (in `/etc/rc.d/rc.sysinit`). Simply add it in the `linuxboot` call, or change your `rc.sysinit`.

The broadcast TFTP ramdisk hangs (*Got in bound state*)

Linux dhcp client is a program that dynamically changes the IP address of the client according to DHCP offers. If the address is offered forever (infinite lease time), the DHCP client just set the address and returns (this is what we expect). However, if the lease time is limited, the DHCP client must remain loaded and ask for new addresses every few minutes. And if the DHCP client does not return, MrBatch will never be loaded... The solution is to give an infinite lease time (sometimes encoded as -1).

5 Remote-Boot Tools Reference Manual

This section provides detailled informations on the use of the tools we developped at the CUI, University of Geneva for this remote-boot configuration.

5.1 BpBatch, MrBatch and MrZip

These three names stand for three variants of the same program, with the following characteristics:

- `BpBatch` is a special program that can be started from the BootProm before the operating system is loaded. It is made of two parts: `bpbatch.P`, the dynamic loader, and `bpbatch.ovl`, the program itself. `BpBatch` has full disk I/O capabilities through our own implementation of FAT16, FAT32 and Ext2fs, as well as remote network I/O capabilities through the BootProm TFTP API. `BpBatch` was compiled under DOS using Borland C 5.0 and Turbo Assembler 3.2.
- `MrBatch` is the DOS/Linux version of `BpBatch`. All commands recognized by `BpBatch` are recognized by `MrBatch` and vice versa. This is very usefull if you want to test your batch scripts from a DOS/Linux session. Under DOS, `MrBatch` emulates remote I/O by OS-based file access if the bootprom is not available. Under Linux, the bootprom cannot be seen anymore but `MrBatch` can emulate it using Linux IP support, or use OS-based file access. `MrBatch` was compiled under Linux using GCC 2.7.2.1 and under DOS using Borland C 5.0 and Turbo Assembler 3.2.
- `MrZip` is an interpreter that recognizes a superset of `MrBatch` language, and that serves to build disk images. In `MrZip`, the limited remote file I/O is replaced by a full-featured OS-based file access. `MrZip` does not include VESA support. `MrZip` was compiled under Linux using GCC 2.7.2.1 and under DOS using Borland C 5.0 and Turbo Assembler 3.2.

5.1.1 Command Line Arguments

All programs accept the same syntax of arguments. `MrBatch` and `MrZip` take them from the command line, while `BpBatch` look for them in the BOOTP option 155 (decimal). Here is the syntax of the arguments:

```
[-x] [-l] [-b] [-v] [-w] [-i] [script-basename]
```

where:

- `-x` disable the use of extended memory
- `-l` disable the use of ISO-latin-8859-1 as default character set
- `-b` cancel the bootprom detection (which cause a floppy seek under DOS)
- `-v` cancel the VESA detection (which cause a switch to full screen under Windows 95)
- `-w` enable direct disk write access (disabled by default under DOS and Linux)
- `-i` enable interactive mode even if a script name is provided

The `script-basename` is optional. If provided, `MrBatch` and `BpBatch` load the file with the `.bpb` extension, and `MrZip` loads the file with the `.mrz` extension. If not provided, `MrBatch` and `MrZip` run in interactive mode while `BpBatch` loads the file with the same basename as the BOOTP Boot file and a `.bpb` extension.

5.1.2 Syntax rules

The following rules apply when `BpBatch` parses an input line.

- Commands are parsed line by line. Lines are separated by CR and/or LF.
- The maximal line length is currently 255 characters.
- Keywords and variable names are case-insensitive.
- " is interpreted as the special string delimiter
- When ${variable} or $variable is encountred, it is substituted by the value of the variable, or by an empty string if the variable is undefined. The substitution also occurs within a string. Moreover, the resulting substituted value must be explicitely enclosed between double quotes if used as a string value (ie. one should merely speak of macro expansion than of a variables).
-
  - `\a` is substituted by the audible-bell character (ASCII 7)
  - `\b` is substituted by the backspace character (ASCII 8)
  - `\n` is substituted by the newline character (ASCII 10)
  - `\r` is substituted by the return character (ASCII 13)
  - `\t` is substituted by the tabulation character (ASCII 9)
  - `\v` is substituted by the vertical-tab character (ASCII ...)
  - `\nnn` where n is a 3-digit octal number between 000 and 377 is substituted by the character with ascii code specified
  - `\X` where X is any other character not listed above is substituted by X itself. In particular,
    - `\"` is substituted by a regular double-quote (not a string-delimiter)
    - `\$` is substituted by a regular dollar sign (not variable substitution)
    - `\\` is substituted by a regular backslash (not a special character)
- The character "end of string" (ASCII code 0) CANNOT be used anywhere as it is used internally as end-of-string delimiter
- The character "floating diaeresis" (ASCII code dec 249, hex F9, octal 371) CANNOT be used in any string as it is used internally as string delimiter in the input parsing routine.
- The character "block space" (ASCII code dec 255, hex FF, octal 377) CANNOT be used in any variable value as it is used internally as variable delimiter.

Empty lines are ignored. Lines starting with a sharp (`#`) are treated as comments and are not interpreted. Lines starting with a column (`:`) are treated as labels and are not interpreted.

String expressions

Strings are delimited by opening and closing double-quotes:

```
"Hello world"
```

To include double-quotes within a string, quote them using a backslash:

```
"I said: \(\backslash\)"Hello world\(\backslash\)""
```

Strings can be postfixed with a few operators.

- The character substitution operator:

```
"Hello world"/o=u/      ==      "Hellu wurld"
"198.76.54.32"/.= /     ==      "198 76 54 32"
```

- The word selection operator (zero-based):

```
"Hello world"{0}        ==      "Hello"
"198 76 54 32"{1-3}     ==      "76 54 32"
```

- The substring selection operator (zero-based):

```
"Hello world"[4]        ==      "o"
"Hello world"[4-7]      ==      "o wo"
```

Operators can be chained by postfixing one after the other. For informations about the string length and word count operators, see under "Numerical expressions".

Numerical expressions

Numerical expressions work on 32-bits integer numbers (from -2,147,483,646 to 2,147,483,647). Hexadecimal octal and binary numbers are not understood. Whenever a numerical expression is expected, the following are recognized:

- A positive or negative integer number
- An expression in the form (*expr1 op expr2*) where *op* can be either +, -, * (multiply), `/` (divide) or `%` (modulo) and `expr` is a numerical expression. Note that EACH operation MUST be enclosed between parenthesis :

```
((3 * 5)+2)             == 17
```

- The string-length operator (@), followed by a string :

```
@"Hello world"          == 11
```

- The word-count operator (#) followed by a string :

```
\#"Hello world"          == 2
```

Durations

A few commands expect durations as arguments. Durations are measured in seconds, with a precision of up to a tenth of second:

```
Delay 3                 waits for 3 seconds
Delay 0.3               waits for 3/10 seconds
```

Colors

Whenever a color is expected, you can either use the numeric value of the color or its symbolic name (case-insensitive). The following colors are recognized

```
Black           0
Blue            1
Green           2
Cyan            3
Red             4
Magenta         5
Brown           6
LightGray       7
DarkGray        8
LightBlue       9
LightGreen     10
LightCyan      11
LightRed       12
LightMagenta   13
Yellow         14
White          15
```

File References

File names are strings. They must therefore always be enclosed between double-quotes. File names are case-sensitive on case-sensitive filesystems, case-insensitive on case-insensitive filesystems. Slash and backslash can be freely used one in place of the other. Do not forget to double backslash since a single backslash is an escape character.

There are two kinds of file references:

- Direct disk files
- Foreign files

Direct disk files are referenced using the following notation:

```
"{disk:partition}/absolute/filename"
```

The disk number can be omitted and defaults to zero. For instance, `"{:1}/usr/bin"` points to `/usr/bin` assuming there is such a directory on the first partition. Direct file I/O is solely based on our own file access routines (we do not use the operating system).

There are two *special* partitions. Partition zero corresponds to the hard disk master boot record (MBR) and has a pseudo file-system which let you access the boot code. Partition minus-one (-1) corresponds to the cache filesystem (see below).

Under BpBatch/MrBatch, foreign files correspond to remote files on the TFTP server when the BootProm is available:

```
"help.bpb"           is the file help.bpb in the /tftpboot directory
"gifs/MyImage.gif"   is a file in /tftpboot/gifs
```

Other TFTP servers can be referenced :

```
"198.76.54.32:help.bpb"
```

If the other server is behind a gateway :

```
"198.70.0.1/198.76.54.31:help.bpb"
```

One can also specify a specific port for the TFTP connection :

```
"198.76.54.32@89:getpasswd/smith"
```

There can be only one open remote file at a time. If the BootProm is not available, remote files are emulated using the operating system file I/O, but the same restriction apply.

Under MrZip, foreign files correspond to files as seen by the operating system. There is no limitation, and foreign files can be used wherever direct disk files can be. Foreign files are usually faster than direct disk files, because the operating system has more buffers. Foreign files can refer to network files if supported by the operating system.

```
"C:\(\backslash\)\(\backslash\)autoexec.bat"
"C:/config.sys"
"/mnt/net/usr"
```

5.1.3 The Cache Filesystem

In order to reduce network load and to fasten the boot process, disk archives, linux kernels and possibly other files are cached on the hard disk. This disk cache is located at the end of the hard disk, between the last cylinder allocated in the partition table and the last physical cylinder of the disk (out of any allocated partition). There MUST be room between the last partition and the end of the disk if you want the cache filesystem to work. The cache filesystem MUST work if you want to restore a disk image.

The disk cache is organised in a volatile, CRC-validated filesystem : Each directory entry and each 32 KB data block is validated by a 32-bits CRC. Whenever a directory entry or a data block unexpectedly changes, the file is automatically removed from the cache and downloaded again upon the next request.

You can freely access the cache filesystem from within BpBatch, MrBatch and MrZip using direct disk access on the special partition `"{:-1}"`. To see the content of the cache, just type :

```
logdir "{:-1}"
```

If the cache ever gets corrupted and is not automatically cleaned (which should never occurs), you can either type :

```
clean -1
```

(in interactive mode) or hold both shifts down when BpBatch access the cache for the first time.

5.1.4 Special variables

Some variable are initially set and/or have special meanings. Some of them exist within all programs, other are only available under MrZip and other are only available when a BOOTP/DHCP reply has been received.

General variables

- `$Program` is set to "BpBatch" within BpBatch, "MrBatch" within MrBatch and "MrZip" within MrZip
- `$Basename` is set to the basename of the script on which the batch interpreter was started
- `$HelpFile` is the name of the file loaded when `Help` is invoked. Default: `"${Basename}.hlp"`
- `$BOOTP-...` are variables set from the BOOTP/DHCP reply (see the paragraph on BOOTP/DHCP variables for more details)
- `$DHCP-...` are variables set from the DHCP reply (see the paragraph on BOOTP/DHCP variables for more details)
- `$Disks` is set to the space-separated list of sizes for each disk. That means, `#"$Disks"` represent the number of disks and `"$Disks"{0}` is the size of the first disk
- `$Keypressed` is set to the next ready-to-read key available in the keyboard buffer (if available)
- `$LBA` controls the use of LBA to access disks ¿ 2Gb. Default: "ON"
- `$FDA` controls the use of fast disk access (write accross cylinders). Default: "ON"
- `$VESA` controls the use of VESA graphics. Default: "ON" if available
- `$Trace` controls the display of each command before execution. It also controls the display of file names when creating new archives. Default: "OFF"
- `$AutoShowLog` controls the automatic switch to the text log whenever the ESC key is pressed. Default: "ON"
- `$PauseLog` controls the pause between each page of log when the log is visible. Default: "ON"
- `$CacheDisk` is set to the disk used for caching remote files. Default: empty == 0, the first hard disk
- `$CacheAlways` controls the automatic caching of remote files copied, patched or drawn as GIF. Default: "OFF"
- `$CacheNever` prevents any file from being cached. Turn this variable on for diskless Linux boot. Default: "OFF"
- `$CacheReserve` controls the preventive allocation of 25 percent more space than necessary in the cache partition, to let the files grow. Turn this variable off if you are short of disk space. Default: "ON"
- `$ExtMemory` controls the use of Extended Memory (or XMS). Once deactivated, extended memory cannot be reactivated. Default: "ON" if available
- `$IsoLatin` controls the interpretation of upper ASCII codes in included and patched files. The IsoLatin settings are processed at the time the file is loaded, not at the time the file is processed. Default: "ON"
- `$ProgressX` and `$ProgressY` controls the position of the progress window displayed in VESA graphics during archive download and decompression. Default: 200 200
- `$EXT2-Backup` controls the update of superblock backups in Linux ext2 filesystem. Superblock backups take a few seconds to do and are never used by current kernels (only by e2fsck).
- `$Security-Gateway` controls the gateway-server used for user authentication. Our special authentication gateway must be running on the target computer. Default: `"${BOOTP-Server-IP}@89"` (ie. the TFTP server, on port 89)
- `$Security-Check` contains the answer of the security server for the last check performed, either PASSED or FAILED. Default: "FAILED"
- `$Security-Passwd, $HelpTopic, $OnExit, $OnKey-...` are used internally.

See also BOOTP variables and MrZip-specific variables.

MrZip-specific variables

The following variables are only used within MrZip.

- `$TempPath` controls the directory where temporary files will be stored. Default: <empty> == current directory

- `$DumpFormat` controls the way archives are dumped to the log when requested. It is a string containing
  - "h"/"H" to display the archive header
  - "b"/"B" to summarize/dump boot sectors
  - "s"/"S" to display a short/long allocation summary
  - "d"/"D" to display a short/long directory listing
  - "f"/"F" to summarize/dump files

  Default: "hbD"
- `$FragmentSize` controls the size of archive pieces. If you do not use InCom's extended TFTP server, you should set this to "30 MB". Default: "87 MB"
- `$SourceArchive, $DestArchive, $Filter...` are used internally.

BOOTP variables

The following BOOTP-... and DHCP-... variables are recognized, as long as a BOOTP/DHCP reply has been received (TCP/IP Bootprom must be reported as detected):

```
$BOOTP-Client-ID
$BOOTP-Your-IP
$BOOTP-Server-IP
$BOOTP-Gateway-IP
$BOOTP-Bootfile
$BOOTP-Server-Name
$BOOTP-Subnet-Mask
$BOOTP-Time-Offset
$BOOTP-Routers
$BOOTP-Time-Servers
$BOOTP-Name-Servers
$BOOTP-Domain-name-Servers
$BOOTP-BOOTP-Log-Servers
$BOOTP-Cookie-Servers
$BOOTP-Lpr-Servers
$BOOTP-Impress-Servers
$BOOTP-Resource-Location-Servers
$BOOTP-Host-Name
$BOOTP-Boot-Size
$BOOTP-Merit-Dump
$BOOTP-Domain-Name
$BOOTP-Swap-Servers
$BOOTP-Root-Path
$BOOTP-Extensions-Path
$BOOTP-IP-Forwarding
$BOOTP-Interface-MTU
$BOOTP-All-Subnets-Are-Local
$BOOTP-Broadcast-Address
$BOOTP-NIS-Domain
$BOOTP-NIS-Servers
$BOOTP-NTP-Servers
$BOOTP-Font-Servers
$BOOTP-X-Display-Manager
$DHCP-IP-Address-Lease-Time
$DHCP-Message-Type
$DHCP-Server-Identifier
$DHCP-Message
$DHCP-Renewal-Time
$DHCP-Rebinding-Time
$BOOTP-NIS+-Domain
$BOOTP-NIS+-Servers
$BOOTP-Server-Name
$BOOTP-Bootfile
```

```
$BOOTP-Mobile-IP-Agent
$BOOTP-SMTP-Servers
$BOOTP-POP3-Servers
$BOOTP-NNTP-Servers
$BOOTP-WWW-Servers
$BOOTP-Finger-Servers
$BOOTP-IRC-Servers
$BOOTP-StreetTalk-Servers
$BOOTP-STDA-Servers
```

Other BOOTP/DHCP parameters can be used under the name

```
$BOOTP-Option-n
```

where n is the decimal representation of the BOOTP option number.
Do not mix-up `BOOTP-Gateway-IP`, which is the gateway to use for TFTP and should be 0.0.0.0 if the TFTP server is in the same subnet, and `BOOTP-Routers`, which contains the default IP gateway(s). The TCP/IP Bootprom sometimes seems to set the value of `BOOTP-Gateway-IP` from the value in `BOOTP-Routers`, causing each TFTP ack packet to be sent to the router first. To avoid such behaviour, if your TFTP server is in the same subnet as the client, force `BOOTP-Gateway-IP` to `0.0.0.0` (thanks to Maciek Uhlig for having pointed out this problem).

5.1.5 Monitoring commands

This section lists commands for monitoring the system state. Optional arguments are listed between parenthesis (I would have prefered square brackets, but LaTeX do not like them at this place...)

Interact
Show the log and turn to interactive mode until QUIT or EXIT is entered. Type HideLog before quitting if you want to avoid disturbing log messages during batch execution.

Help (topic)
Load the on-line help file (`bpbatch.hlp`) and display the description of the given topic. If no topic is provided, or if the topic is unknown, display the help index.

Log "text"
Display the string on the log. No return/linefeed is implicitely added.

Echo "text"
Display the string on the log and go to the next line. Equivalent to

```
Log "text\r\n".
```

LogVars ("pattern")
Log (ie. display on the log) all variables matching the given pattern. The pattern can contain wildcards (? and *).

```
Example: LogVars "BOOTP-*"              list all BootP variables
```

LogDir "path/pattern"
Log (ie. display on the log) all files from the given path that match the pattern. The pattern can contain wildcards (? and *).

```
Example: LogDir "/usr/g*p"              list files names like g...p
```

LogTree "path"
Log the directory tree starting with the given path as root.

LogFile "filename"
Log the content of the file. The file must be no more than 64 KB big.

ShowLog
Make the log visible if it was hidden. Automatically performed when ESC is pressed with "$AutoShowLog" == "ON" and when entering interactive mode.

HideLog
Prevent log messages to appear on the screen. Default state when BpBatch, MrBatch and MrZip are started on a script file.

CaptureLog

Record all log output to a 64 KB buffer until EndCapture is issued. Wrap around buffer if the log output is more than 64 KB big. This command can be used to create a text file with an arbitrary content. The EndCapture MUST occurs within the same batch file.

EndCapture ("filename")

End up the capture of the log. If a filename is given, store the captured text to a file. Otherwise, discard it.

5.1.6 Control commands

This section lists commands that control the batch execution. Optional arguments are listed between parenthesis.

Include "filename"

Load the given file and start up the parser on it. Go back to the current point when the include file processing is done. The interpretation of characters above ASCII 127 within the include file depends on the value of $IsoLatin at the time the file is included.

OnExit *command*

Setup an exit-handler that will automatically be evaluated at the end of current batch file.

Goto *label*

Move the execution cursor to the given label (ie. the line starting with :*label*)

Eval "command"

Perform all substitutions on the "command" and run the parser on it.

If ...

```
If (not) <expr1> (==|!=|<|>|>=|<=|=>|=<|<>) <expr2> <command>
If (not) (ci) "str1" (==|!=|<|>|>=|<=|=>|=<|<>) "str2" <command>
If (not) (ci) "str1" Match-Expr "pattern" <command>
If (not) (ci) "str1" Match-Passwd "unix-passwd" <command>
If (not) (ci) "str1" In "wordlist" <command>
If (not) (ci) "str1" In-File "filename" <command>
If (not) exist "filename" <command>
```

These commands execute *command;* if the test succeeds. The 1st form compares two numerical expressions. The 2nd form compares two strings, optionally case-insensitive. The 3rd form tests if "str1" matches the given pattern (wildcards allowed). The 4th form tests if the clear password "str1" matches the Unix-crypted password. The 5th form tests if "str1" is included in the word list. The 6th form tests if "str1" is included in the word file. The 7th form tests if the given file exists.

Set ...

```
Set variable = "string-value"
Set variable = <expr>
```

Setup a value for the given variable. If the given value is a numerical expresison, it will be implicitely converted to a string. A variable can be used anywhere by refering it as $variable or ${variable}. If the resulting reference is to be interpreted as a string, it should be enclosed between double quotes: "$variable" or "${variable}".

Delay *duration*

Waits until the specified duration (expressed in seconds) expired. See also the paragraph on the format of durations.

GetTime *variable*, GetDate *variable*

Get the CMOS time and store it into *variable*in the form HH:MM:SS. Get the CMOS date and store it into *variable*in the form YY/MM/DD. This can be used to customize the behavior of your boot scripts depending on the time of day or on the date.

SetTime "HH:MM:SS", SetDate "YY/MM/DD"

Set the computer CMOS time or date to the given value. If you have a security gateway (our special TFTP server) running, you can automatically adjust the CMOS time and date of the client computers at each boot by evaluating the following command:

```
include "$Security-Gateway:gettime"
```

If you want to understand what this command does, just type:

```
logfile "$Security-Gateway:gettime"
```

5.1.7 Keyboard-related commands

This section lists commands that let you monitor the keyboard input. Optional arguments are listed between parenthesis. See also under *National Language Support*.

GetKey (*variable*)
: Indefinitely wait until a key is pressed and store it in the *variable*.

WaitForKey *duration* (*command*)
: Wait until a key is pressed for no more than *duration* seconds. If no key has been pressed after the given time, evaluate the *command*. Otherwise, leave the key in the keyboard buffer. See also the paragraph on the format of durations.

Input (*variable* (*max-length*))
: Read a return-terminated string from the keyboard and store the result string in *variable* (without the terminating return). If *max-length* is given, do not allow the user to enter more than this number of characters. See also `GetPasswd` under *Security-related commands*.

OnKey "c" *command*
: Setup a key handler that will automatically evaluate the given *command* when the key "c" is pressed (except is explicitely waited by a GetChar or an Input command). If the string `"default"` is used instead of a single character, the command is executed if any other key is pressed.

5.1.8 Text output commands

This section lists commands used to perform regular text output. All these commands can be used in graphic mode also, with the same behaviour (except that text mode provides 80x25 characters while graphic mode provides 100x37, because graphic mode characters are of size 8x16). Optional arguments are listed between parenthesis. See also under *National Language Support*.

Print *"text"/expr*
: Print the specified string/expression at current cursor position and using current text attributes, then move the cursor. Add "\r\n" to the end of the string to go to the next line.

TextAttr *fg-color bg-color*
: Setup the text attributes. One can also put a single numeric value representing both colors and defined as 16**bg-color*+*fg-color*.
If you need more fantasy, you can use `LoadFont`. See under *National Language Support*.

At *line*,*col* (*command*)
: Move the cursor position to the specified position and evaluate the command if provided.

```
Example: At 10,20 Print "Gnats and rats !"
```

Clear (*color* (*pattern-char* (*top*,*left*,*bottom*,*right*)))
: Fill the given text area with the given *pattern-char* (either a string or the decimal ascii code). The area defaults to the full screen, the pattern char defaults to the full block (ASCII dec 219) and the color defaults to black (clear screen). Move the cursor to the upper left corner of the cleared area.

BpMenu backward compatibility commands

```
.ATT (<attribute>)
.CLS (<attribute>)
.DEF <key> (<timeout_val>)
.KEY <key> <filename>
.POS ((<x>) <y>)
.PWD <key> <cpasswd>
.WLN (<text>)
.WRT <text>
```

See InCom's manual for more infos. We wrote some time ago a program program for editing menu files using this syntax, but it is preferable to make your menus using the new explicit syntax. Note that the .PWD command is not implemented because we do not now the password crypting algorithm used by InCom GmbH.

5.1.9 Graphics output commands

This section lists commands used to perform graphic-mode output. For the functions listed in this section, coordinates are given in pixels. You can also use all text output commands (see above) in graphic mode. Optional arguments are listed between parenthesis.

Note that the graphic mode is automatically turned on whenever a graphic command is used, unless the variable `VESA` is set to `"OFF"`.

InitGraph
: Turn on VESA graphics (800 pixels width, 600 pixels height, 256 colors). The origin is on the upper-left corner of the screen (0 0). VESA graphics may hang some computers under Windows 95. Run MrBatch with the -v option to avoid such problems.

CloseGraph
: Close VESA graphic mode and go back to text mode.

DrawBar *x-pos y-pos width height color*
: VESA graphics. Draw a filled bar of the given size and colors.

DrawWindow *x-pos y-pos width height* (*bg-color* (*title-color*))
: VESA graphics. Draw a window of the given size and colors. The background color defaults to LightGray and the title color defaults to Blue.

Drawtext *x-pos y-pos* "text" (*fg-color*)
: VESA graphics. Draw the text string at the given position with a transparent background. The color defaults to text foreground color.

DrawGif "gif-filename" (*x-pos y-pos* (*color-strategy*))
: VESA graphics. Load the given GIF-87a file and draw it on the screen. The file can be interlaced, but must be in GIF-87a (not GIF-89a). The image size should be no more than 800x600 with 256 colors. The GIF position defaults to the top left corner of the screen (0 0).
The *color-strategy* defines the allocation of colors in the palette when more than 256 colors are needed (for instance when two 256 colors GIF files are displayed simultaneously):

- `Best-Colors` use best possible colors for the most recent GIF
- `Spare-Colors` try to avoid allocating colors, change existing colors
- `Share-Colors` try to avoid allocating colors, use existing colors
- `Reuse-Colors` allocate no new color, only use existing colors

The default strategy is `Best-Colors`.

5.1.10 Security-related commands

This section lists commands that help you authenticate a user. Optional arguments are listed between parenthesis.

Some of these functions cooperate with a *Security gateway*, that you should first install. See the section on *Special TFTP servers* for more infos.

GetPasswd (*variable* (*max-length*))
: Same as Input, but echo stars instead of the typed characters.

Crypt "text" "salt" *variable*
: Apply the Unix crypt function to the given 8-chars text and store the resulting crypted string into *variable*. The "salt" is usually a two-character string that will be found as the first two characters of the crypted string. Note that Unix crypt is a one-way function. It is not possible to decode the crypted string. One can only try to crypt another string with the same salt and compre the resulting crypted string.

DESCrypt "text" "key" *variable*
: Crypt the given text using the given 8-chars key and store the result as an hexadecimal string in *variable*.

DESDecrypt "hexcode" "key" *variable*
: Decrypt the given hexadecimal string using the given 8-chars key and store the result in *variable*.

MD5 "text" *variable*
: Compute the MD5 checksum of the given text and store it as an hexadecimal string in *variable*. Can be used as an alternative to the Unix crypt function to check for passwords bigger than 8 characters.

CheckUser "user" "password" "domain"

Connect to the $Security-Gateway and check if the given user exist in the given radius domain and uses the specified password. If the domain is `"Unix"`, use the Unix user/password definition on the security gateway. For any other domain, use the security gateway domain definition file to determine the real Radius or NT domain to check.
Set the value of $Security-Check to "PASSED" or "FAILED". The password do not transit in clear on the network.

5.1.11 Disk-related commands

This section lists commands for preparing the hard-disk. Optional arguments are listed between parenthesis.

GetPartitions *variable* (*disk*)
Read the partition table(s) for the given disk and store it as a string into the given *variable*. The result string is a space-separated list of *Type:Size*, where

- *Type* is FAT16, EXT, BIGDOS, NTFS, FAT32, FAT32-LBA, BIGDOS-LBA, EXT-LBA, LINUX-SWAP, LINUX-EXT2 or the decimal filesystem id for unknown types.
- *Size* is the size of the partition in megabytes.

SetPartitions "partitions" (*disk*)
Setup the partition table(s) to the content of the string. The format used is the same that for GetPartitions. This command also reset all boot flags (hint: use SetBootPart).
If you want to use extended partitions, remember that they begin after each four partitions. You can use EMPTY to leave a free entry in the partition table. For instance, if you want to setup a primary dos partition of 100 MB and a logical dos partition of 400 MB, you must execute

```
SetPartitions "BIGDOS:100 EXT:400 EMPTY EMPTY BIGDOS:400"
```

GetBootPart *variable* (*disk*)
Get the partition number with the boot flag turned on (DOS says: the activated primary partition) and store it to the *variable*. The first partition is numbered 1. If no partitions has the boot flag turned on, answers zero.

SetBootPart *partition* (*disk*)
Set the boot flag to the given partition. The boot flag let the master boot record (MBR) choose which partition to boot on. The first partition is numbered 1.

Blank *partition* (*disk*)
Fill the given partitions with zeroes. Can take quite a lot of time for big partitions. Do not format the partition for any operating system. See also `Clean`.

Clean *partitions* (*disk*) ("label")
Fast-format the given partition(s) according to the type declared in the partition table. If a label is given and the filesystem supports it, setup the partition label. For a paranoiac full format, call `Blank` on the partition first.
Clean is supported for (FAT16) BIGDOS, FAT32, EXT, LINUX-EXT2 and LINUX-SWAP partitions. To clean the master boot record (MBR), use `Clean 0`.
Clean should be used on data partitions and on MBR/EXT partitions. It is totally useless to clean a partition before unzipping a filesystem on it using `FullUnzip`.

FullUnzip "full-archive" *partition* (*disk*)
Decompress a full disk archive to the given partition, overwriting any existing file (clean-up on the fly).
FullUnzip is supported for (FAT16) BIGDOS, FAT32 and LINUX-EXT2. Unfortunately, images restored from FAT32 do not seem to be bootable, although all files are restored correctly. This should be fixed in a future release.
This commands turn on VESA graphics to display a progress banner, unless `VESA` has been turned `OFF`.

IncrUnzip "incr-archive" "destpath"
Decompress an incremental disk archive to the given path. Files in the archive replace those with the same name on the target path, but other files are not deleted.
IncrUnzip is supported for (FAT16) BIGDOS, FAT32 and LINUX-EXT2. This command is far less efficient than FullUnzip since the existing filesystem structure must be preserved. However, it avoids multiplying the number of different disk images by storing the differences only.

FileUnzip "source-filename" "dest-filename"
Uncompress a file previously compressed with `MrZip` FileZip command. The file is validated by a 32-bits CRC.

Copy "source-filename" "dest-filename"

Copy the source file to the destinaton file, byte-to-byte. Can be used after a FullUnzip for instance to update configuration files from the server without rebuilding the image. Better to use `FileUnzip` for big and easy-to-compress files.

Append "src-filename-1" "src-filename-2" "dest-filename"

Copy the first, then the second file to the destination file, byte-to-byte. Can be used on arbitrary large files. The destination file cannot be one of the two source files.

Patch "source-filename" "dest-filename" ("prefix" ("postfix"))

Read the source file and perform variable substitution before writing it to the destination file. The interpretation of characters above ASCII 127 depends on the value of $IsoLatin.

By default, variables are recognized when prefixed by "${" and postfixed by "}". This can be changed to any other non-empty string. remember that if you want to use a dollar sign within the prefix or suffix, you must escape it or it will get macro-evaluated. For instance, if you want to explicitely use the default prefix and postfix, use:

```
Patch "source-file" "dest-file" "\${" "}"
```

MkDir "path"

Recursively create directories from the root to the given full path. If the path already exists, this command has no effect.

Delete "filename", Del "filaname"

Remove the given file. The file must exist.

DelTree "path"

Recursively remove all files and directories under the given path, and remove the directory itself.

5.1.12 Boot commands

This section lists commands for continuing the boot process. Optional arguments are listed between parenthesis.

HideBootProm

Restore the memory and the interrupt vectors allocated by the bootprom. All attempts to make TFTP transfers will fail after calling this command. It is usually a good idea to call this command before HdBoot, or you might run short of memory under DOS/Windows. This command is implicitely called by FloppyBoot.

Note that although this function restore all vectors "officially" rerouted by the BootProm, it does not seems to restore everything. But it works well enough for DOS and Windows.

LoadRamDisk "ramdisk-filename"

Load a floppy disk image into the extended memory and redirect the BIOS Disk Services to make floppy disk calls use this image instead. This command implicitely calls `HideBootProm`. Call `FloppyBoot` to boot on the ramdisk you just loaded.

This kind of ramdisk may not be as robust as what you get when you use the TFTPBoot command. The only advantage is that it only steals a few hundred bytes of conventional memory instead of the >64 KB reserved by the BootPROM. Warning, nothing secures the extended memory in which the ramdisk resides. There is no way to uninstall such a ramdisk.

LoadZRamDisk "ramdisk-filename"

Do the same as `LoadRamDisk`, but for an image that has been compressed using `MrZip` FileZip command. Compressed ramdisks are protected against data corruption (and uncomplete download) by a byte count and a 32-bits CRC.

TFTPBoot "remote-bootfile"

Chain to another boot file (for instance a floppy image made with InCom's BpShell program). See the file referencing conventions for accessing a file on another TFTP server.

FloppyBoot

Hide the BootPROM, load the floppy disk boot sector and boot on it. Unfortunately, this command does not seems to work well enough for LDLINUX.SYS (used on RedHat Linux setup disk) to work.

HdBoot (*disk*)(*:partition*)

Load the given boot sector and boot from it. The disk default to zero, the first hard disk, and the partition defaults to zero, ie. the master boot record. You can boot from any partition, but be warned that Windows 95 may not let you boot a partition that has not been set as the boot partition (hint: use SetBootPart).

This command does not implicitely call HideBootProm, so you might want to call it before.

LinuxBoot "kernelfile" ("command-line" ("ramdisk-file"))

Load the given kernel and ramdisk into the high memory, setup the command line and boot the kernel. It is a good idea to put at least a minimal command line with the location of the root filesystem (like "root=dev/hda1"/). If you are using a linux system that heavily relies on `lilo` (like RedHat Linux 5.1), it may be necessary to add to the command line something like `BOOT_IMAGE=linux`. Note that the kernel can be loaded by TFTP (automatically cached on the hard disk) or directly from the target root partition.
This command works for small and big kernels (`zImage` and `bzImage`).

5.1.13 National language support

This section lists commands related not national language support. Optional arguments are listed between parenthesis.

RemapKeys "original-keys" "remapped-keys"

National keyboard support. Remap given keys to other characters. For instance, to swap the Y and Z keys, use

```
Remapkeys "yzYZ" "zyZY"
```

It is a good idea to use the quoted octal notation when using characters not included in the minimal ASCII character set, in order to avoid a dependency to the iso-latin modal settings.
For international keyboards, there are two keys that produce a backslash in non-remapped (US) mode. Each of them can be independantly remapped, thanks to the fact that `BpBatch` sees one of them as a key answering ASCII code 252 (octal) or ASCII code 335 (octal) when shifted.
If you send me a sample script that does keyboard mapping for your national keyboard, I will make it available under `http://cuiwww.unige.ch/info/pc/remote-boot/soft/keyboards` To help you make your own keyboard mapping, I suggest pressing all *special* keys without remapping the keyboard and writing down the character they produce. These will be the `original-keys`. The `remapped-keys` simply are the key you would have liked to see, in the same order. If some keys (either original or remapped) produce characters above ASCII dec 127, use the quoted octal notation. You can easily get the octal code for any given character by looking in the ASCII table of HelpPC for instance (HelpPC is a shareware hypertext on-line help program by David Jurgens).

RemapAltkeys "original-keys" "remapped-keys"

National keyboard support. Remap the given keys when ALT is depressed For instance, to map Alt-2 to the ampersand sign, use

```
RemapAltKeys "2" "@"
```

Note that dead keys are not supported.

LoadCodePage "cpxxx.bin"

Load and activate the given binary Codepage file. Codepages are used for the translation of Unicode characters (present on VFAT valumes for instance) into 8-bits characters. If you do not have the right Codepage loaded, you will get FAT warnings while accessing the filesystem when special characters are encountred.
All binary codepage files are available at `http://cuiwww.unige.ch/info/pc/remote-boot/soft/codepage.zip`
The default codepage is 850, a reordered superset of ISO-Latin-1. If you load a more exotic codepage, you should usually turn the variable `$IsoLatin` to `"off"` or you might get meaningless implicit conversions. Moreover, if you want to display exotic characters, you should also load the proper screen font (use `"LoadFont"`).

LoadFont "fontfile"

Load and activate a VGA/VESA font, both in text and graphic mode. The font file must be a binary file of 16 bztes/characters (8x16 bitmap). This command can be used for National Language Support as well as for Fantasy support.
An archive with several fantasy fonts is available at `http://cuiwww.unige.ch/info/pc/remote-boot/soft/fonts.zip`.
This archive also contains a program to extract fonts for your codepage from the DOS `.CPI` file.

5.1.14 Commands specific to MrZip

Source...

```
Source (i)archive "filename"
Source path "path"
```

Set the source for the archive manipulation to the given (incremental) archive file or disk path.

Dest...

```
Dest (i)archive "filename"
Dest (i)dump
Dest path "path"
```

Set the destination for the archive manipulation to the given (incremental) archive file, dump or disk path. To control the quantity of data displayed during dump, use the $DumpFormat special variable.

FileZip "source-filename" "dest-filename"

Compress a file for further decompression with FileUnzip or for using as ZRamDisk. The file is validated by a 32-bits CRC.

Filter...

```
Filter -"pattern"
Filter +"pattern"
```

Avoid/allow files and directories matching the given pattern (wildcards allowed) to be included in the archive. The pattern is matched agains the full pathname. By default, all files are included in the image. You only need to explicitely allow files that where cancelled by a filter. Each negative filter has its own positive filter (allowed) sublist.
For DOS/Windows images, you will typically use

```
Filter -"*.swp"
Filter -"temp/*"
```

and for Unix images, you will typically use

```
Filter -"var/log/*"
Filter -"tmp/*"
```

CopyArchive

Start the archive manipulation operation, according to source, destination and filter settings. Except in a few circumstances, you will probably use the shortcut below instead of explicitely calling `CopyArchive`. One circumstance in which you will use `CopyArchive` explicitely is when you want to change the fragmentation of an image, as follow:

```
set FragmentSize="30 MB"
Source archive "original.imz"
Dest archive "refragmented.imz"
CopyArchive
```

FullZip "path" "full-archive"

Shortcut for

```
Source path "path"
Dest archive "full-archive"
CopyArchive
```

You should usually first setup filters.

IncrZip "path" "incr-archive"

Shortcut for

```
Source path "path"
Dest iarchive "incr-archive"
CopyArchive
```

FullDump "full-archive"

Shortcut for

```
Source archive "full-archive"
Dest dump
CopyArchive
```

IncrDump "incr-archive"
Shortcut for

```
Source iarchive "incr-archive"
Dest dump
CopyArchive
```

XCopy "srcpath" "dstpath"
Shortcut for

```
Source path "srcpath"
Dest path "dstpath"
CopyArchive
```

5.2 NoBreak.sys

`Nobreak.sys` is a very small (about 350 bytes only) driver that you include at the beginning of your `config.sys`. Its goal is to secure the boot process, until the user is logged in. DOS provides a setting for this (namely `BREAK=OFF`), but it is not drastic enough, and has almost no effect in the `autoexec.bat`. Our driver works by modifying the scan-code of the key pressed when a break is requested, directly at the BIOS level. This way, no program at all can receive a break until break is enabled again.

The driver must be loaded from the `config.sys` (or using the `devlod` program from *Undocumented DOS*). Afterwards, break can be enabled by sending `Yes` to the `NOBRK` pseudo-device, and disabled again by sending `No` (in fact, only the first character, `Y` or `N` is significant).

As this driver relies on the BIOS, it does only work for DOS and Windows 3.1. Windows 95 has its own low-level keyboard handling routines.

Assembler source code is available.

6 Special TFTP Servers

As the only network support available in the BootPROM is TFTP, there is a special interest in enhancing TFTP servers for providing new capabilities.

6.1 InCom's Enhanced TFTP Server

InCom GmbH distributes with the BootPROM an enhanced TFTP server that can send packets of up to 1408 bytes instead of the standard 512 bytes. This is a great enhancement that you should use. This server is available for Linux, Solaris, Windows and as Netware NLM.

6.2 The Security Gateway

We wrote a special TFTP server that serves as security gateway for authenticating users. This server runs under Linux or Solaris, and can authenticate users according to a Unix password database (NIS and shadow passwords are supported), a Windows NT (or Samba) server or a Radius server. It is available from `http://cuiwww.unige.ch/info/pc/remote-boot/soft/stdtpd.tar.gz`, with source and precompiled binaries. The precompiled binaries do not include NT password encryption as we cannot distribute `libdes` but compilation is straightforward.

In order to use the security gateway, you just have to setup a trivial *security domains* configuration file that describes to which authentication server each logical security domains maps (the `Unix` domain implicitely maps to the server Unix password database). This is a sample configuration file:

```
#
# STFTPD configuration file
#
# This file specify the server of the "security domains". Two types of
# authentication servers are supported : radius or winnt (winnt includes
```

```
# NT Server and Samba)
#
# Format of radius servers
# radius        <domain>        <serveraddress>         <secret>
#
# secret is the secret word as specified in your /etc/raddb/clients file
#
# Format of SMB servers
# winnt         <domain>        <serveraddress>         <netbiosname>
#
# netbiosname is the NETBIOS name of your server
#
# Examples
radius          sec-dom-rad     radiusserver    testing123
winnt           sec-dom-nt1     192.168.1.1     NTSERVER1
winnt           sec-dom-smb     samba           SAMBA1
```

Note that if you are using Samba, you must set `security = user`.

You can also provide to the security server a file containing a list of users which are not allowed to log on (for which the check will fail anyways).

6.3 The Broadcast TFTP Server

We wrote a special TFTP server that implements a home-made Broadcast variant of TFTP. Using this server, we were able to download images to 25 clients on a heavily loaded 10 Mb ethernet network at 6 Mb/s (it is more efficient than the regular TFTP because it does not need to acknowledge each packets). This server runs under Linux or Solaris. It is available from `http://cuiwww.unige.ch/info/pc/remote-boot/soft/btdtpd.tar.gz`, with source and precompiled binaries.

As the TCP/IP bootprom does not support this protocol, our solution consist in booting a tiny ramdisk-based linux system using the tools described in this document, and running the Linux version of MrBatch which has built-in support for Broadcast TFTP. A simple batch file can the download all files to the cache in a few minutes, simultaneously on all client computers. You do not need to install Linux yourself to use this package, except if you have exotic hardware and cannot directly use the kernel provided in the package.

The process works as follow. First, you startup the broadcast server manually, giving the number of expected client computers as argument (remember, this procedure is not to be used every day but only when you changed an image and want to ensure it is immediately uploaded to all your client computers). Then, you turn on all client computers, which will run the following BpBatch script:

```
#
# This batch is run by bpbatch to launch a mini-linux using an initial
# ramdisk, which will then run mrbatch under linux.
#
# The broadcast TFTP protocol only works with the Linux implementation of
# mrbatch, because of the lack of broadcast support in the bootprom itself.
#
# 1. Setup a tiny partition, to let a lot of space for the cache
setpartitions "BIGDOS:50"
# 2. Clean the MBR
clean 0
# 3. Run a Linux Kernel with initrd (Initial Ramdisk) supprt, and use
#    bcastrd.gz as the initial ramdisk (will be mounted root and then
#    executed via /linuxrc). See initrd.txt for more details about
#    initial ramdisks. You don't have to specify a root device (second
#    parameter is null) to the kernel, it will use the initial ramdisk.
linuxboot "linux.krn" "" "bcastrd.gz"
# 4. The initial ramdisk will run dhcpcd to setup networking using DHCP.
#    It will then run mrbatch -w bcastlx
```

The initial ramdisk contains:

- `dhcpcd`, a DHCP client used to setup networking

- `mrbatch`
- `linuxrc`, a little wrapper automatically started by initrd and that starts `dhcpcd` then `mrbatch`.
- `usr/lib/terminfo/l/linux`, used by MrBatch
- `dev/*`, devices needed to run Linux and mrbatch

All programs are statically linked and stripped, to avoid `libc.so` which is really huge. The resulting ramdisk is Gzipped and takes less than 300 KB. The kernel itself takes 450 KB (with many network cards and initrd support). When Linux is up and running, MrBatch is called with the following script (that you should edit for your needs):

```
# This file is executed when mrbatch is launched by the initial ramdisk
# bcastrd.gz
# It's main purpose is to "broacast copy" files to the cache
#
# 1. Be verbose
showlog
# 2. Don't want a "press a key"
set pauselog="OFF"
# 3. Set partitions at their final values.
#    Important: Since you will copy files into the cache to be used in future
#    boot, you need to specify the same partitions as in the future boots.
setpartitions "BIGDOS:1024"
# 4. Clean the CACHE partition
clean -1
# 5. And the copy files into the cache, using the Broadcast TFTP protocol
#    (port 99)
#
# You can use the script "as is", but you surely need to modify the following
# line ! In our example, we download the file mblinux.imz, which is the image
# file for our installation of Linux.
copy "$BOOTP-Server-IP@99:mblinux.imz" "{:-1}mblinux.imz"
```

When the transfer is done, you can simply turn off all client computers and change their initial boot script to your favorite menu.

7 Discussion

We here discuss some theoretical issues related to our configuration.

7.1 Bootproms and Hard Disks

Bootproms exist for quite a long time, but they are usually used for diskless computers only. In our opinion, bootproms are even more interesting for computers which have a local harddisk, since they allow to take profit of both sides:

- A bootprom make the configurations more robust, since it ensure that the computer will always boot the same way, no matter any virus or partition table crash. It can be used, as we did, to cleanup the harddisk even before the operating system is loaded.
- A local harddisk make the configuration more efficient, since it can reduce the network trafic through caching, and allows for efficient swap.

7.2 Which Bootprom ?

Several bootproms are available for PCs. We had several reason for choosing the TCP/IP Bootprom from Köppen EDV GmbH:

- It is based on the BOOTP/DHCP protocol, which is publicly defined by RFCs. The definition states that when a BOOTP/DHCP server receives a request from a client that he doesn't know, the server will *not* answer. This avoids interferences between multiple servers, as you might sadly experience with the MSD boot server. Moreover, since IP broadcasts are confined to the local subnet, they produce less noise than their IPX counterpart.

- It is not bound to a specific operating system.
- Technical informations and API informations are available on request.
- Home-made boot loader can be written (as we have done)
- Home-made boot scripts can be written (as you can do using BpBatch)
- The boot process can be parametrized on many ways. Specifically, it allowed us to forestall floppy boot on old-fashioned AST computers, which BIOS did not include this feature.