SMB HOWTO

Table of Contents

SMB HOWTO

David Wood, dwood@plugged.net.au

v1.3, 20 April 2000

This is the SMB HOWTO. This document describes how to use the Server Message Block (SMB) protocol, also called the Session Message Block, NetBIOS or LanManager protocol, with Linux using Samba.

1.License

2.Introduction

3.Further Information

4.Installation

5.Running The Daemons

6.General Configuration (/etc/smb.conf)

7.Sharing A Linux Drive With Windows Machines

8.Accessing an SMB Share With Linux Machines

9.Sharing A Linux Printer With Windows Machines

10.Sharing A Windows Printer With Linux Machines

11.Backing Up Windows Machines to a Linux Host

12.Using Samba Across Routed Networks

13.Acknowledgements

1.License

2.Introduction

This is the SMB HOWTO. This document describes how to use the Server Message Block (SMB) protocol, also called the Session Message Block, NetBIOS or LanManager protocol, with Linux using Samba. Although this document is Linux−centric, Samba runs on most Unix−like operating systems.

This document is maintained by David Wood (dwood@plugged.net.au). Additions, modifications or corrections may be mailed there for inclusion in the next release.

Much more Samba documentation is available at the Samba Web site, located at http://www.samba.org/. There is a tremendous amount of information there; please have a look before asking for help! You also might try the comp.protocols.smb newsgroup.

The SMB protocol is used by Microsoft Windows 3.11, NT and 95/98 to share disks and printers. Using the Samba suite of tools by Andrew Tridgell (Andrew.Tridgell@anu.edu.au), UNIX (including Linux) machines can share disk and printers with Windows hosts. The smbfs tools by Paal−Kr. Engstad (engstad@intermetrics.com) and Volker Lendecke (lendecke@namu01.gwdg.de) enable Unix machines to mount SMB shares from Windows or Samba hosts.

There are four basic things that one can do with Samba:

1. Share a Linux drive with Windows machines.
2. Access an SMB share with Linux machines.
3. Share a Linux printer with Windows machines.
4. Share a Windows printer with Linux machines.

All of these are covered in this document, plus a few other odds and ends.

Disclaimer: The procedures and scripts either work for the author or have been reported to work by the people that provided them. Different configurations may not work with the information given here. If you encounter such a situation, please e−mail the author with suggestions for improvement in this document.

Please note that for Windows 3.x machines to access SMB shares, they must have a TCP/IP stack and the Win32s DLLs. Both of these are available on Microsoft's Web site (http://www.microsoft.com). As of the writing of this version of the HOWTO, Microsoft are reportedly requiring a subscription to the Microsoft Software Developers Network (MSDN) to download the TCP/IP−32 stack for Windows 3.x from their Web site. Since this software used to be free, many older copies are in existance and may be acquired from friends and user group contacts.

3.Further Information

This HOWTO attempts to explain how to configure basic SMB file and print services on a Linux machine. Samba is a very complex and complete package. There would be no point in attempting to duplicate all of the documentation for Samba here.

For further information, please see the following documents:

- The Samba documentation, available as part of the Samba distribution. The distribution is available at: ftp://ftp.samba.org/
- The Linux Printing HOWTO.
- Protocol Standard For A NetBIOS Service On A TCP/UDP Transport.

RFC 1001

> RFC 1001 − Concepts and Methods.

RFC 1002

> RFC 1002 − Detailed Specifications.

4.Installation

First, in order to use Samba your machines must be on a single ethernet LAN segment using the TCP/IP protocol. Samba will not work using other network protocols. This is generally easy since Linux and Windows 95/98/NT ship with TCP/IP support. However, if you are using Windows 3.X machines TCP/IP support will need to be added. One of the most common questions that I get asked is why Samba "isn't working" when Windows machines are not using TCP/IP.

In order to setup Windows 95/98 to use TCP/IP, select Control Panel | Network, then add and configure Microsoft TCP/IP. Under Windows NT, select Control Panel | Network | Protocols.

To get the latest source version of Samba, go to this URL and pick the closest mirror site to you: ftp://ftp.samba.org/.

In most cases, though, your Linux distribution will already come with an installable package containing a recent version of Samba.

The following two daemons are required for the Samba package. They are typically installed in /usr/sbin and run either on boot from the systems startup scripts or from inetd. Example scripts are shown in Running the Daemons.

```
        smbd (The SMB daemon)
        nmbd (Provides NetBIOS nameserver support to clients)
```

Please note that the name service provided by the nmbd daemon is different from the name service provided by the Domain Name Service (DNS). NetBIOS name service is a 'Windows−style' name service used for SMB. In other words, having DNS name service tells you nothing about the state of the ability for Samba to resolve host names.

Typically, the following Samba binaries are installed in /usr/bin or /usr/local/samba/bin, although the location is optional.

```
        smbclient       (An SMB client for UNIX machines)
        smbprint        (A script to print to a printer on an SMB host)
        smbprint.sysv   (As above, but for SVR4 UNIX machines)
        smbstatus       (Lists the cuurent SMB connections for the local host)
        smbrun          (A 'glue' script to facilitate runnning applciations
                         on SMB hosts)
```

The binaries for smbfs file system support are discussed later in this document.

Additionally, a script called 'print' is included with this HOWTO, which serves as a useful front end to the smbprint script.

The Samba package is simple to install. Simply retrieve the source from the location mentioned above, and read the file README in the distribution. There is also a file called docs/INSTALL.txt in the distribution that provides a simple step−by−step set of instructions.

Following installation, place the daemons in /usr/sbin and the binaries in /usr/bin. Install the man pages in

/usr/local/man.

When you made the Samba package, you would have specified in the Makefile the location for the configuration file, smb.conf. This is generally in /etc, but you can put it anywhere you like. For these directions, we will presume that you specified the location of the configuration file as /etc/smb.conf, the log file location as log file = /var/log/samba−log.%m and the lock directory as lock directory = /var/lock/samba.

Install the configuration file, smb.conf. Go to the directory where Samba was built. Look in the subdirectory examples/simple and read the file README. Copy the file smb.conf found in that directory to /etc. BE CAREFUL! If you have a Linux distribution that already has Samba installed, you may already have a Samba configuration file in /etc. You should probably start with that one.

If you don't want to have your configuration file in /etc, put it wherever you want to, then put a symlink in /etc:

```
        ln -s /path/to/smb.conf /etc/smb.conf
```

5.Running The Daemons

The two SMB daemons are /usr/sbin/smbd and /usr/sbin/nmbd. Under most Linux distributions, these are started, stoped and restarted via the startup script located in /etc/rc.d/init.d/smb and symlinked to the appropriate runlevels.

If you choose not to use the standard startup script, you can run the Samba daemons from inetd or as stand−alone processes. Samba will respond slightly faster as a standalone daemon than running from inetd.

In either case, you should check the file /etc/services for lines that look like this:

```
netbios-ns      137/tcp         nbns
netbios-ns      137/udp         nbns
netbios-dgm     138/tcp         nbdgm
netbios-dgm     138/udp         nbdgm
netbios-ssn     139/tcp         nbssn
```

Make sure they are all uncommented. Depending on your distribution, you may even need to add them. Samba will not be able to bind to the appropriate ports unless /etc/services has these entries.

To run the daemons from inetd, place the following lines in the inetd configuration file, /etc/inetd.conf:

```
    # SAMBA NetBIOS services (for PC file and print sharing)
    netbios-ssn stream tcp nowait root /usr/sbin/smbd smbd
```

```
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd
```

Then restart the inetd daemon by running the command:

```
kill -HUP `cat /var/run/inetd.pid`
```

To run the daemons from the system startup scripts, put the following script in file called /etc/rc.d/init.d/smb (for most distributions) and symbolically link it to the files specified in the comments:

```
#!/bin/sh

#
# /etc/rc.d/init.d/smb - starts and stops SMB services.
#
# The following files should be synbolic links to this file:
# symlinks: /etc/rc.d/rc1.d/K35smb  (Kills SMB services on shutdown)
#           /etc/rc.d/rc3.d/S91smb  (Starts SMB services in multiuser mode)
#           /etc/rc.d/rc6.d/K35smb  (Kills SMB services on reboot)
#

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

# See how we were called.
case "$1" in
  start)
    echo -n "Starting SMB services: "
    daemon smbd -D
    daemon nmbd -D
    echo
    touch /var/lock/subsys/smb
    ;;
  stop)
    echo -n "Shutting down SMB services: "
    killproc smbd
    killproc nmbd
    rm -f /var/lock/subsys/smb
    echo ""
    ;;
  *)
    echo "Usage: smb {start|stop}"
    exit 1
esac
```

If when starting Samba you get an error that says something about the daemon failing to bind to port 139, then you probably have another Samba process already running that hasn't yet shut down. Check a process list (with 'ps auxww | grep mbd') to determine if another Samba service is running.

6.General Configuration (/etc/smb.conf)

Samba configuration on a Linux (or other UNIX machine) is controlled by a single file, /etc/smb.conf. This file determines which system resources you want to share with the outside world and what restrictions you wish to place on them.

Since the following sections will address sharing Linux drives and printers with Windows machines, the smb.conf file shown in this section is as simple as you can get, just for introductory purposes.

Don't worry about the details, yet. Later sections will introduce the major concepts.

Each section of the file starts with a section header such as [global], [homes], [printers], etc.

The [global] section defines a few variables that Samba will use to define sharing for all resources.

The [homes] section allows a remote users to access their (and only their) home directory on the local (Linux) machine). That is, users trying to connect to this share from Windows machines, will be connected to their personal home directories. Note that to do this, they must have an account on the Linux box.

The sample smb.conf file below allows remote users to get to their home directories on the local machine and to write to a temporary directory. For a Windows user to see these shares, the Linux box has to be on the local network. Then the user simply connects a network drive from the Windows File Manager or Windows Explorer.

Note that in the following sections, additional entries for this file will be given to allow more resources to be shared.

```
; /etc/smb.conf
;
; Make sure and restart the server after making changes to this file, ex:
; /etc/rc.d/init.d/smb stop
; /etc/rc.d/init.d/smb start

[global]
; Uncomment this if you want a guest account
; guest account = nobody
   log file = /var/log/samba-log.%m
   lock directory = /var/lock/samba
   share modes = yes

[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mode = 0750
```

```
[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes
```

Having written a new smb.conf, it is useful to test it to verify its correctness. You can test the correctness of a smb.conf file , using the 'testparm' utility (man page: testparm); if testparm reports no problems, smbd will correctly load the configuration file.

Here's a good trick: If your Samba server has more than one ethernet interface, the smbd may bind to the wrong one. If so, you can force it to bind to the intended one by adding a line that looks like this to the [global] section of /etc/smb.conf:

```
interfaces = 192.168.1.1/24
```

where you replace the IP address above with the one that is assigned to the correct ethernet interface. The "24" is correct for a Class C network, but may have to be recalculated if you have subnetted the network. The number relates to the netmask. Numbers for other classes of networks are given in the IP−Masquerade mini−HOWTO.

There is now a GUI configuration tool for Samba: GtkSamba. See http://www.open−systems.com/gtksamba.html.

7.Sharing A Linux Drive With Windows Machines

As shown in the simple smb.conf above, sharing Linux drives with Windows users is easy. However, like everything else with Samba, you can control things to a large degree. Here are some examples:

To share a directory with the public, create a clone of the [tmp] section above by adding something like this to smb.conf:

```
[public]
   comment = Public Stuff
   path = /home/public
   public = yes
   writable = yes
   printable = no
```

To make the above directory readable by the public, but only writable by people in group staff, modify the entry like this:

```
[public]
   comment = Public Stuff
   path = /home/public
   public = yes
   writable = yes
   printable = no
   write list = @staff
```

It used to be that easy; you would now be able to start Samba and browse the shares from a Windows PC. However, Microsoft has recently made life slightly more difficult for those using Samba. Windows 98, Windows NT (service pack 3 or higher) and later builds of Windows 95 now use encrypted passwords by default. Samba uses unencrypted passwords by default. You can't browse servers when either the client or server is using encrypted passwords, because a connection cannot be made anonymously.

You can tell if you have a password type mismatch between client and server if when you try to connect to a share you see a dialog box which reads something like "You are not authorized to access that account from this machine".

You can either configure your Samba server to use encrypted passwords, or configure the Windows machines to use unencrypted passwords.

To get Windows to work with encrypted SMB passwords:

Windows 95/98 =============

Using the registry editor (regedit), create the registry setting HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP Add a new DWORD value: Value Name: EnablePlainTextPassword Data: 0x01.

Windows NT ==========

Using the registry editor (regedit), create the registry setting HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Rdr\Parameters Add a new DWORD value: Value Name: EnablePlainTextPassword Data: 0x01

Windows 2000 ============

Using the registry editor (regedit), create the registry setting HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkStation\Parameters Add a new DWORD value: Value Name: EnablePlainTextPassword Data: 0x01

Once these registry changes have been made, reboot the Windows machine and try to map a network drive on the Samba server again. It should work as long as the Samba server is using plain text passwords.

To configure Samba to use encrypted passwords:

In the [global] section of /etc/smb.conf, add the following lines:

```
encrypt passwords = yes
smb passwd file = /etc/smbpasswd
```

You are highly encouraged to read the files ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation before doing this!

If your clients and server are using encrypted passwords, you will not be able to browse the available shares on the server until an initial connection has been made with the appropriate authentication. To get the initial connection, enter the share name manually in the Windows File Manager or Explorer dialog box, in the form '\\<hostname>\<sharename>'. Log onto the server with a username and password that is valid on the server!

If you suspect that your NetBIOS name service is not correctly configured (perhaps because you get 'host not found' errors when trying to connect), try using just the IP address of the server: '\\<host ip address>\<sharename>'.

In order to get filenames to appear correctly, you may also need to set some options in the appropriate share section. These work for Windows 95/98/NT clients, but may need to be modified if you have Windows 3.X clients:

```
    ; Mangle case = yes seems to give the correct filenames
    ; for Win95/98/NT.
    mangle case = yes

    ; If samba is case sensitive when looking for files
    case sensitive = no

    ; Default case of files that are created
    default case = lower

    ; Preserve case for all filenames
    preserve case = yes

    ; Preserve case for dos (8.3) filenames
    short preserve case = no
```

For other tricks to play with drive shares, see the Samba documentation or man pages.

```
interfaces = 192.168.1.1/24
```

Note: The bit after the / is a reference to the subnet mask. "24" is the value to use for an unsegmented Class C network. For more information on subnet calculations, you might want to see http://www.ralphb.net/IPSubnet/.

There is a lot more to Samba configuration than this, but this will get you started. If you want to do something more advanced, I refer you to the Samba Web site mentioned above.

8.Accessing an SMB Share With Linux Machines

Linux (UNIX) machines can also browse and mount SMB shares. Note that this can be done whether the server is a Windows machine or a Samba server!

An SMB client program for UNIX machines is included with the Samba distribution. It provides an ftp−like interface on the command line. You can use this utility to transfer files between a Windows 'server' and a Linux client.

Most Linux distributions also now include the useful smbfs package, which allows one to mount and umount SMB shares. More on smbfs below.

To see which shares are available on a given host, run:

```
    /usr/bin/smbclient −L host
```

where 'host' is the name of the machine that you wish to view. this will return a list of 'service' names − that is, names of drives or printers that it can share with you. Unless the SMB server has no security configured, it will ask you for a password. Get it the password for the 'guest' account or for your personal account on that machine.

For example:

```
    smbclient −L zimmerman
```

The output of this command should look something like this:

```
Server time is Sat Aug 10 15:58:27 1996
Timezone is UTC+10.0
Password:
Domain=[WORKGROUP] OS=[Windows NT 3.51] Server=[NT LAN Manager 3.51]

Server=[ZIMMERMAN] User=[] Workgroup=[WORKGROUP] Domain=[]

        Sharename      Type      Comment
        ---------      ----      -------
        ADMIN$         Disk      Remote Admin
```

```
        public          Disk      Public
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        OReilly         Printer   OReilly
        print$          Disk      Printer Drivers


This machine has a browse list:

        Server               Comment
        ---------            -------
        HOPPER               Samba 1.9.15p8
        KERNIGAN             Samba 1.9.15p8
        LOVELACE             Samba 1.9.15p8
        RITCHIE              Samba 1.9.15p8
        ZIMMERMAN
```

The browse list shows other SMB servers with resources to share on the network.

To use the client, run:

```
    /usr/bin/smbclient service <password>
```

where 'service' is a machine and share name. For example, if you are trying to reach a directory that has been shared as 'public' on a machine called zimmerman, the service would be called \\zimmerman\public. However, due to shell restrictions, you will need to escape the backslashes, so you end up with something like this:

```
    /usr/bin/smbclient \\\\zimmerman\\public mypasswd
```

where 'mypasswd' is the literal string of your password.

You will get the smbclient prompt:

```
Server time is Sat Aug 10 15:58:44 1996
Timezone is UTC+10.0
Domain=[WORKGROUP] OS=[Windows NT 3.51] Server=[NT LAN Manager 3.51]
smb: \>
```

Type 'h' to get help using smbclient:

```
smb: \> h
ls             dir            lcd            cd             pwd
get            mget           put            mput           rename
```

```
more            mask            del             rm              mkdir
md              rmdir           rd              prompt          recurse
translate       lowercase       print           printmode       queue
cancel          stat            quit            q               exit
newer           archive         tar             blocksize       tarmode
setmode         help            ?               !
smb: \>
```

If you can use ftp, you shouldn't need the man pages for smbclient.

Although you can use smbclient for testing, you will soon tire of it for real work. For that you will probably want to use the smbfs package. Smbfs comes with two simple utilties, smbmount and smbumount. They work just like mount and umount for SMB shares.

One important thing to note: You must have smbfs support compiled into your kernel to use these utilities!

The following shows a typical use of smbmount to mount an SMB share called "customers" from a machine called "samba1":

```
[root@postel]# smbmount "\\\\samba1\\customers" -U rtg2t -c 'mount /customers -u 500 -g 100'
Added interface ip=192.168.35.84 bcast=192.168.255.255 nmask=255.255.0.0
Got a positive name query response from 192.168.168.158 ( 192.168.168.158 )
Server time is Tue Oct  5 10:27:36 1999
Timezone is UTC-4.0
Password:
Domain=[IPM] OS=[Unix] Server=[Samba 2.0.3]
security=user
```

Issuing a mount command will now show the share mounted, just as if it were an NFS export:

```
[root@postel]# mount
/dev/hda2 on / type ext2 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,mode=622)
//SAMBA1/CUSTOMERS on /customers type smbfs (0)
```

Please see the manual pages for smbmount and smbumount for details on the above operation.

9.Sharing A Linux Printer With Windows Machines

To share a Linux printer with Windows machines, you need to make certain that your printer is set up to work under Linux. If you can print from Linux, setting up an SMB share of the printer is stright forward.

Note that Windows users must have an account on the Linux/Samba server in order to print. Windows 95/98 will attempt to authenticate to the print server using the username and password used on login to the Windows box.This means that if you clicked 'Cancel' when logging onto Windows, you can't print (or connect to other SMB services)! Windows NT allows one to explicitely provide a username and password when connecting to a printer.

See the Printing HOWTO to set up local printing.

Add printing configuration to your smb.conf:

```
[global]
   printing = bsd
   printcap name = /etc/printcap
   load printers = yes
   log file = /var/log/samba-log.%m
   lock directory = /var/lock/samba

[printers]
   comment = All Printers
   security = server
   path = /var/spool/lpd/lp
   browseable = no
   printable = yes
   public = yes
   writable = no
   create mode = 0700

[ljet]
   security = server
   path = /var/spool/lpd/lp
   printer name = lp
   writable = yes
   public = yes
   printable = yes
   print command = lpr -r -h -P %p %s
```

Make certain that the printer path (in this case under [ljet]) matches the spool directory in /etc/printcap!

The lines:

```
   printcap name = /etc/printcap
   load printers = yes
```

controls whether all the printers in /etc/printcap should be loaded by default. If you do this, there is no reason to set up printers individually. The section [printers] specifies options for the printers that you wish to explicitly difine. If the printing subsystem you are using doesn't work this way (BSD), you need to set up a fake printcap file (or to use the 'print command' technique, see below). For more information on the printcap system see the Printing HOWTO.

A useful technique to test the network connection is to change the print command to:

```
print command = cp %S /tmp/print.%P.%S
```

The resulting file can then be analyzed.

NOTE: There are some problems sharing printers on UNIX boxes with Windows NT machines using Samba. One problem is with NT seeing the shared printer properly. To fix this, see the notes in the Samba distribution in the file docs/WinNT.txt. The other deals with password problems. See the comments in the same file for an annoying gain of understanding and failure to fix the problem.

Oleg L. Machulskiy (machulsk@shade.msu.ru) suggests that a better print command to use in the above example would be:

```
print command = smb2ps %s | lpr -r -h -P %p
```

where 'smb2ps' is a script which transforms the spool file received from Windows into usual a usable Postscript file. It must cut off first 3 lines and last 2 lines, because these lines contain some PJL or PCL codes.

That approach is only needed if your Windows machine is printing PCL and not real Postscript. I have found that Windows 95/98/NT don't have a generic Postscript driver per se, but the "Digital turbo Printserver 20" driver acts as a good general Postscript driver for most setups. I have also heard that the "Apple LaserWriter II NTX" driver works for this purpose.

If you are creating a printer spool directory instead of using one created by a Linux distribution's installation utility, be careful of permissions! Neil Fraser (neilf@necon.co.za) suggested setting the permissions of the spool directory (in his case, /var/spool/lpd/lpr) to 4755 (note the suid bit). This worked for him when the owner of the directory was 'root' and the group was 'lp'.

Jeff Stern (jstern@eclectic.ss.uci.edu) reported that he had to set the permissions on his spool directory to 777 in order for non−priviledged users to print, although he notes that he could have also added users to the 'lp' group. This is a decision for local systems administrators; if printing security is an issue, then lock it down. In home environments, you will probably want everyone to be able to print.

Dr. Michael Langner (langner@fiz−chemie.de) points out that write permission problems on the /var/spool/lpd/ tree could be avoided by use something like "path = /tmp" and "print command = lpr −r −P%p %s" instead.

Sometimes, a Postscript parsing error will occur with Postscript printing from Windows machines that causes an extra page to be printed at the end of every print job. The last page will always have "%%[Lastpage]%%" at the top of it. This seems to happen with Windows 95 and 98 only and is because the Postscript is malformed.

One way to handle that is to use a script to remove that bit of bad Postscript from the spooled jobs. Another way is to try to find a better Windows Postscript driver. Probably the best way is to us LPRng instead of Postscript to print to a Samba server.

Erik Ratcliffe (erik@caldera.com) Caldera tells me that using LPRng means that any printer driver can be used from Windows machines. On the Samba server, they used an /etc/printcap entry that looked like this:

```
raw:\
        :rw:sh:
        :lp=/dev/lp1
        :sd=/var/spool/lpd/raw
        :fx=flp
```

LPRng doesn't require :\ at the end of every line. A printer entry will still need to be made in /etc/smb.conf for the physical printer. The print command line needs to use the "raw" entry in /etc/printcap and data must be sent to the printer in binary form. Try a print command line like this:

```
print command = lpr -b -Praw %s
```

You may also need to set the spooling on the Windows95 end to print directly to the printer instead of spooling.

If you constantly get a extra page printing at the end of print jobs from Windows clients, try adding an "sf" directive in /etc/printcap. This will suppress form feeds separating jobs, but will not effect form feeds within documents.

10.Sharing A Windows Printer With Linux Machines

To share a printer on a Windows machine, you must do the following:

1. You must have the proper entries in /etc/printcap and they must correspond to the local directory structure (for the spool directory, etc).
2. You must have the script /usr/bin/smbprint. This comes with the Samba source, but not with all Samba binary distributions. A slightly modifed copy is discussed below.

3. If you want to convert ASCII files to Postscript, you must have nenscript, or its equivalent. nenscript is a Postscript converter and is generally installed in /usr/bin.
4. You may wish to make Samba printing easier by having an easy−to−use front end. A simple perl script to handle ASCII, Postscript or created Postscript is given below.
5. You could also use MagicFilter to do the above. The details on setting up MagicFilter are given below the perl script. MagicFilter has advantages because it knows how to automatically convert a lot of file formats.

The /etc/printcap entry below is for an HP 5MP printer on a Windows NT host. The entries are as follows:

```
        cm - comment
        lp - device name to open for output
        sd - the printer's spool directory (on the local machine)
        af - the accounting file
        mx - the maximum file size (zero is unlimited)
        if - name of the input filter (script)
```

For more information, see the Printing HOWTO or the man page for printcap.

```
# /etc/printcap
#
# //zimmerman/oreilly via smbprint
#
lp:\
        :cm=HP 5MP Postscript OReilly on zimmerman:\
        :lp=/dev/lp1:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/spool/lpd/lp/acct:\
        :mx#0:\
        :if=/usr/bin/smbprint:
```

Make certain that the spool and accounting directories exist and are writable. Ensure that the 'if' line holds the proper path to the smbprint script (given below) and make sure that the proper device is pointed to (the /dev special file).

Next is the smbprint script itself. It is usually placed in /usr/bin and is attributable to Andrew Tridgell, the person who created Samba as far as I know. It comes with the Samba source distribution, but is absent from some binary distributions, so I have recreated it here.

You may wish to look at this carefully. There are some minor alterations that have shown themselves to be useful.

```
#!/bin/sh -x

# This script is an input filter for printcap printing on a unix machine. It
# uses the smbclient program to print the file to the specified smb-based
# server and service.
```

```
# For example you could have a printcap entry like this
#
# smb:lp=/dev/null:sd=/usr/spool/smb:sh:if=/usr/local/samba/smbprint
#
# which would create a unix printer called "smb" that will print via this
# script. You will need to create the spool directory /usr/spool/smb with
# appropriate permissions and ownerships for your system.

# Set these to the server and service you wish to print to
# In this example I have a WfWg PC called "lapland" that has a printer
# exported called "printer" with no password.

#
# Script further altered by hamiltom@ecnz.co.nz (Michael Hamilton)
# so that the server, service, and password can be read from
# a /usr/var/spool/lpd/PRINTNAME/.config file.
#
# In order for this to work the /etc/printcap entry must include an
# accounting file (af=...):
#
#   cdcolour:\
#       :cm=CD IBM Colorjet on 6th:\
#       :sd=/var/spool/lpd/cdcolour:\
#       :af=/var/spool/lpd/cdcolour/acct:\
#       :if=/usr/local/etc/smbprint:\
#       :mx=0:\
#       :lp=/dev/null:
#
# The /usr/var/spool/lpd/PRINTNAME/.config file should contain:
#   server=PC_SERVER
#   service=PR_SHARENAME
#   password="password"
#
# E.g.
#   server=PAULS_PC
#   service=CJET_371
#   password=""

#
# Debugging log file, change to /dev/null if you like.
#
logfile=/tmp/smb-print.log
# logfile=/dev/null


#
# The last parameter to the filter is the accounting file name.
#
spool_dir=/var/spool/lpd/lp
config_file=$spool_dir/.config

# Should read the following variables set in the config file:
#   server
#   service
#   password
#   user
eval `cat $config_file`

#
# Some debugging help, change the >> to > if you want to same space.
#
echo "server $server, service $service" >> $logfile
```

```
(
# NOTE You may wish to add the line `echo translate' if you want automatic
# CR/LF translation when printing.
        echo translate
        echo "print -"
        cat
) | /usr/bin/smbclient "\\\\$server\\$service" $password -U $user -N -P >> $logfile
```

Most Linux distributions come with nenscript for converting ASCII documents to Postscript. The following perl script makes life easier be providing a simple interface to Linux printing via smbprint.

```
Usage: print [-a|c|p] <filename>
       -a prints <filename> as ASCII
       -c prints <filename> formatted as source code
       -p prints <filename> as Postscript
        If no switch is given, print attempts to
        guess the file type and print appropriately.
```

Using smbprint to print ASCII files tends to truncate long lines. This script breaks long lines on whitespace (instead of in the middle of a word), if possible.

The source code formatting is done with nenscript. It takes an ASCII file and foramts it in 2 columns with a fancy header (date, filename, etc). It also numbers the lines. Using this as an example, other types of formatting can be accomplished.

Postscript documents are already properly formatted, so they pass through directly.

```
#!/usr/bin/perl

# Script:   print
# Authors:  Brad Marshall, David Wood
#           Plugged In Communications
# Date:     960808
#
# Script to print to a Postscript printer via Samba.
# Purpose:  Takes files of various types as arguments and
# processes them appropriately for piping to a Samba print script.
#
# Currently supported file types:
#
# ASCII      - ensures that lines longer than $line_length characters wrap on
#              whitespace.
# Postscript - Takes no action.
# Code       - Formats in Postscript (using nenscript) to display
#              properly (landscape, font, etc).
#

# Set the maximum allowable length for each line of ASCII text.
$line_length = 76;

# Set the path and name of the Samba print script
```

```
$print_prog = "/usr/bin/smbprint";

# Set the path and name to nenscript (the ASCII-->Postscript converter)
$nenscript = "/usr/bin/nenscript";

unless ( -f $print_prog ) {
        die "Can't find $print_prog!";
}
unless ( -f $nenscript ) {
        die "Can't find $nenscript!";
}

&ParseCmdLine(@ARGV);

# DBG
print "filetype is $filetype\n";

if ($filetype eq "ASCII") {
        &wrap($line_length);
} elsif ($filetype eq "code") {
        &codeformat;
} elsif ($filetype eq "ps") {
        &createarray;
} else {
        print "Sorry..no known file type.\n";
        exit 0;
}
# Pipe the array to smbprint
open(PRINTER, "|$print_prog") || die "Can't open $print_prog: $!\n";
foreach $line (@newlines) {
        print PRINTER $line;
}
# Send an extra linefeed in case a file has an incomplete last line.
print PRINTER "\n";
close(PRINTER);
print "Completed\n";
exit 0;

# --------------------------------------------------- #
#        Everything below here is a subroutine        #
# --------------------------------------------------- #

sub ParseCmdLine {
        # Parses the command line, finding out what file type the file is

        # Gets $arg and $file to be the arguments (if the exists)
        # and the filename
        if ($#_ < 0) {
                &usage;
        }
        # DBG
#       foreach $element (@_) {
#               print "*$element* \n";
#       }

        $arg = shift(@_);
        if ($arg =~ /\-./) {
                $cmd = $arg;
        # DBG
#       print "\$cmd found.\n";

                $file = shift(@_);
```

```
        } else {
                $file = $arg;
        }

        # Defining the file type
        unless ($cmd) {
                # We have no arguments

                if ($file =~ /\.ps$/) {
                        $filetype = "ps";
                } elsif ($file =~ /\.java$|\.c$|\.h$|\.pl$|\.sh$|\.csh$|\.m4$|\.inc$|\.html$|\.htm$/) {
                        $filetype = "code";
                } else {
                        $filetype = "ASCII";
                }

                # Process $file for what type is it and return $filetype
        } else {
                # We have what type it is in $arg
                if ($cmd =~ /^-p$/) {
                        $filetype = "ps";
                } elsif ($cmd =~ /^-c$/) {
                        $filetype = "code";
                } elsif ($cmd =~ /^-a$/) {
                        $filetype = "ASCII"
                }
        }
}

sub usage {
        print "
Usage: print [-a|c|p] <filename>
       -a prints <filename> as ASCII
       -c prints <filename> formatted as source code
       -p prints <filename> as Postscript
        If no switch is given, print attempts to
        guess the file type and print appropriately.\n
";
        exit(0);
}

sub wrap {
        # Create an array of file lines, where each line is < the
        # number of characters specified, and wrapped only on whitespace

        # Get the number of characters to limit the line to.
        $limit = pop(@_);

        # DBG
        #print "Entering subroutine wrap\n";
        #print "The line length limit is $limit\n";

        # Read in the file, parse and put into an array.
        open(FILE, "<$file") || die "Can't open $file: $!\n";
        while(<FILE>) {
                $line = $_;

                # DBG
                #print "The line is:\n$line\n";

                # Wrap the line if it is over the limit.
                while ( length($line) > $limit ) {
```

```
                        # DBG
                        #print "Wrapping...";

                        # Get the first $limit +1 characters.
                        $part = substr($line,0,$limit +1);

                        # DBG
                        #print "The partial line is:\n$part\n";

                        # Check to see if the last character is a space.
                        $last_char = substr($part,-1, 1);
                        if ( " " eq $last_char ) {
                            # If it is, print the rest.

                            # DBG
                            #print "The last character was a space\n";

                            substr($line,0,$limit + 1) = "";
                            substr($part,-1,1) = "";
                            push(@newlines,"$part\n");
                        } else {
                             # If it is not, find the last space in the
                             # sub-line and print up to there.

                            # DBG
                            #print "The last character was not a space\n";

                             # Remove the character past $limit
                             substr($part,-1,1) = "";
                             # Reverse the line to make it easy to find
                             # the last space.
                             $revpart = reverse($part);
                             $index = index($revpart," ");
                             if ( $index > 0 ) {
                               substr($line,0,$limit-$index) = "";
                               push(@newlines,substr($part,0,$limit-$index)
                                   . "\n");
                             } else {
                               # There was no space in the line, so
                               # print it up to $limit.
                               substr($line,0,$limit) = "";
                               push(@newlines,substr($part,0,$limit)
                                   . "\n");
                             }
                        }
                }
                push(@newlines,$line);
        }
        close(FILE);
}

sub codeformat {
        # Call subroutine wrap then filter through nenscript
        &wrap($line_length);

        # Pipe the results through nenscript to create a Postscript
        # file that adheres to some decent format for printing
        # source code (landscape, Courier font, line numbers).
        # Print this to a temporary file first.
        $tmpfile = "/tmp/nenscript$$";
        open(FILE, "|$nenscript -2G -i$file -N -p$tmpfile -r") ||
```

```
                die "Can't open nenscript: $!\n";
        foreach $line (@newlines) {
                print FILE $line;
        }
        close(FILE);

        # Read the temporary file back into an array so it can be
        # passed to the Samba print script.
        @newlines = ("");
        open(FILE, "<$tmpfile") || die "Can't open $file: $!\n";
        while(<FILE>) {
                push(@newlines,$_);
        }
        close(FILE);
        system("rm $tmpfile");
}

sub createarray {
        # Create the array for postscript
        open(FILE, "<$file") || die "Can't open $file: $!\n";
        while(<FILE>) {
                push(@newlines,$_);
        }
        close(FILE);
}
```

Now the MagicFilter way. Thanks to Alberto Menegazzi (flash.egon@iol.it) for this information.

Alberto says:

−−− 1) Install MagicFilter with the filter for the printers you need in /usr/bin/local but DON'T fill /etc/printcap with the suggestion given by the documentation from MagicFilter.

2) Write the /etc/printcap like this way (it's done for my LaserJet 4L):

lp|ljet4l:\ :cm=HP LaserJet 4L:\ :lp=/dev/null:\ # or /dev/lp1 :sd=/var/spool/lpd/ljet4l:\ :af=/var/spool/lpd/ljet4l/acct:\ :sh:mx#0:\ :if=/usr/local/bin/main−filter:

You should explain that the lp=/dev/... is opened for locking so "virtual" devices one for every remote printer should be used.

Example creating with : touch /dev/ljet4l

3) Write the filter /usr/local/bin/main−filter the same you suggest using the ljet4l−filter instead of cat.

Here's mine.

#! /bin/sh logfile=/var/log/smb−print.log spool_dir=/var/spool/lpd/ljet4l (echo "print −" /usr/local/bin/ljet4l−filter) | /usr/bin/smbclient "\\\\SHIR\\HPLJ4" −N −P >> $logfile

P.S. : here is the quote from the Print2Win mini−Howto about locking and why creating virtual printers

−−−Starts here

Hint from Rick Bressler :

Good tip sheet. I use something very similar. One helpful tip, this is not a particularly good idea:

:lp=/dev/null:\

lpr does an 'exclusive' open on the file you specify as lp=. It does this in order to prevent multiple processes from trying to print to the dame printer at the same time.

The side effect of this is that in your case, eng and colour can't print at the same time, (usually more or less transparent since they probably print quickly and since they queue you probably don't notice) but any other process that tries to write to /dev/null will break!

On a single user system, probably not a big problem. I have a system with over 50 printers. It would be a problem there.

The solution is to create a dummy printer for each. Eg: touch /dev/eng.

I have modified the lp entries in the printcap file above to take into account Rick's suggestion. I did the following:

#touch /dev/eng #touch /dev/colour

−−−Ends here

−−

11.Backing Up Windows Machines to a Linux Host

Adam Neat (adamneat@ipax.com.au) kindly contributed the following script to back up Windows machines to a Linux host, using the smbclient utility. Adam says that it is used to backup Windows 3.x and NT machines to a Linux based DAT SCSI Drive.

Adam is not proud of the coding style used here, but it works. As I like to say, "If it works and its stupid, then it is not stupid".

Another Windows backup script, contributed by Dan Tager (dtager@marsala.com), is provided below. Dan's script also backs up Unix machines via rsh, although that could be modified to use ssh rather easily.

In this script, the string 'agnea1' is the username on the Linux machine that does the backups.

```
#!/bin/bash

clear
echo Initialising ...
checkdate=`date | awk '{print $1}'`

if [ -f "~agnea1/backup-dir/backup-data" ]; then

        echo "ERROR: No config file for today!"
        echo "FATAL!"
        exit 1
fi

if [ -d "~agnea1/backup-dir/temp" ]; then

        echo "ERROR: No tempoary directory found!"
        echo
        echo "Attempting to create"
        cd ~agnea1
        cd backup-dir
        mkdir temp
        echo "Directory Made - temp"
fi

if [ "$1" = "" ]; then

        echo "ERROR: enter in a machine name (ie: cdwriter)"
        exit 1
fi

if [ "$2" = "" ]; then

        echo "ERROR: enter in a SMB (Lan Manager) Resource (ie: work)"
        exit 1
fi

if [ "$3" = "" ]; then

        echo "ERROR: enter in an IP address for $1 (ie:
        130.xxx.xxx.52)" exit 1
fi


##############################################################################
# Main Section
#
##############################################################################

cd ~agnea1/backup-dir/temp
rm -r ~agnea1/backup-dir/temp/*
cd ~agnea1/backup-dir/

case "$checkdate"
in
        Mon)
                echo "Backuping for Monday"
                cat backup-data | /usr/local/samba/bin/smbclient
                \\\\$1\\$2 -I$3 -N echo "Complete"

                        if [ -d "~agnea1/backup-dir/Monday" ]; then
                                echo "Directory Monday Not found ...
                                making" mkdir
```

```
                        ~agnea1/backup-dir/Monday
                fi

        echo "Archiving ..."
        cd ~agnea1/backup-dir/temp
        tar -cf monday.tar *                echo "done ..."
        rm ~agnea1/backup-dir/Monday/monday.tar
        mv monday.tar ~agnea1/backup-dir/Monday
        ;;


Tue)
        echo "Backuping for Tuesday"
        cat backup-data | /usr/local/samba/bin/smbclient
        \\\\$1\\$2 -I$3 -N echo "Complete"

                if [ -d "~agnea1/backup-dir/Tuesday" ]; then
                        echo "Directory Tuesday Not found ...
                        making" mkdir
                        ~agnea1/backup-dir/Tuesday
                fi
        echo "Archiving ..."
        cd ~agnea1/backup-dir/temp
        tar -cf tuesday.tar *
        echo "done ..."
        rm ~agnea1/backup-dir/Tuesday/tuesday.tar
        mv tuesday.tar ~agnea1/backup-dir/Tuesday
        ;;

Wed)
        echo "Backuping for Wednesday"
        cat backup-data | /usr/local/samba/bin/smbclient
        \\\\$1\\$2 -I$3 -N echo "Complete"

                if [ -d "~agnea1/backup-dir/Wednesday" ]; then
                        echo "Directory Wednesday Not found
                        ... making" mkdir
                        ~agnea1/backup-dir/Wednesday
                fi
        echo "Archiving ..."
        cd ~agnea1/backup-dir/temp
        tar -cf wednesday.tar *
        echo "done ..."
     rm ~agnea1/backup-dir/Wednesday/wednesday.tar
        mv wednesday.tar ~agnea1/backup-dir/Wednesday
        ;;

Thu)
        echo "Backuping for Thrusday"
        cat backup-data | /usr/local/samba/bin/smbclient
        \\\\$1\\$2 -I$3 -N echo "Complete"

                if [ -d "~agnea1/backup-dir/Thursday" ]; then
                        echo "Directory Thrusday Not found ...
                        making" mkdir
                        ~agnea1/backup-dir/Thursday
                fi
        echo "Archiving ..."
        cd ~agnea1/backup-dir/temp
        tar -cf thursday.tar *
        echo "done ..."
        rm ~agnea1/backup-dir/Thursday/thursday.tar
```

```
                mv thursday.tar ~agnea1/backup-dir/Thursday
                ;;


        Fri)
                echo "Backuping for Friday"
                cat backup-data | /usr/local/samba/bin/smbclient
                \\\\$1\\$2 -I$3 -N echo "Complete"

                        if [ -d "~agnea1/backup-dir/Friday" ]; then
                                echo "Directory Friday Not found ...
                                making" mkdir
                                ~agnea1/backup-dir/Friday
                        fi
                echo "Archiving ..."
                cd ~agnea1/backup-dir/temp
                tar -cf friday.tar *
                echo "done ..."
                rm ~agnea1/backup-dir/Friday/friday.tar
                mv friday.tar ~agnea1/backup-dir/Friday
                ;;

        *)
                echo "FATAL ERROR: Unknown variable passed for day"
                exit 1;;

esac
###########
```

Here's Dan's backup script:

```
#!/bin/bash

BACKDIR=3D/backup
WINCMD=3D/usr/bin/smbclient

function CopyWinHost(){

# tars and gzips "windows shares" to a local directory using samba's
# smbclient
# argument 1 is the remote host window's host name
# argument 2 is the share name to be backed up

   echo $1,$2,$3
   REMOTE=3D$1
   SHARE=3D$2
   DEST=3D$3

 # create a tarred gzip file using samba to copy direct from a
 # windows pc
 # 12345 is a password.  Needs some password even if not defined on
 # remote system
   $WINCMD \\\\$REMOTE\\$SHARE 12345 -Tc -|gzip > $DEST
   echo `date`": Done backing up "$REMOTE" to "$DEST
   echo
}
```

```
function CopyUnixHost(){

# tars and gzips a directory using rsh
# argument 1 is the name of the remote source host
# argument 2 is the full path to the remote source directory
# argument 3 is the name of the local tar-gzip file.  day of week
#  plus .tgz will be appended to argument 3

   REMOTE=3D$1
   SRC=3D$2
   DEST=3D$3


   if rsh $REMOTE tar -cf - $SRC |gzip > $DEST; then
      echo `date`": Done backing up "$REMOTE":"$SRC" to "$DEST
   else
     echo `date`": Error backing up "$REMOTE":"$SRC" to "$DEST
   fi

}

# $1: win=3Dbackup windows machine, unix=3Dbackup unix machine
case $1 in
   win)
      # $2=3D remote windows name, $3=3Dremote share name,
      # $4=3Dlocal destination directory
      CopyWinHost $2 $3 $4;;
   unix)
      # $2 =3D remote host, $3 =3D remote directory,
      # $4 =3D destination name
      CopyUnixHost $2 $3 $4;;
esac
```

12.Using Samba Across Routed Networks

Andrew Tridgell states that SMB host browsing across routers is problematic. Here are his suggestions to allow this:

−− For cross−subnet (ie. routed) browsing you should do the following. There are other methods but they are much more complex are error prone:

1) all computers that you want visible should use a single WINS server (Samba or NT can do this)

2) the master browser for each subnet must be either NT or Samba. (Win9X doesn't communicate cross−subnet browse info correctly)

3) You should use the same workgroup name on all subnets. This is not strictly necessary but it is the simplest way to guarantee success. If you can't arrange this then you must organise for a way for browse info

to propogate between subnets. (It does *not* propogate via WINS). It propogates via two mechanisms: i) each browse master notices workgroup announcements from other browse masters on the same broadcast domain ii) each non−Win9X browse master contacts the global DMB for the workgroup (typically the domain controller or a Samba box marked as the domain master) and swaps full browse info periodically. −−

Also, Rakesh Bharania points out that Cisco routers can be configured to forward SMB traffic in a way that allows browsing. His suggestion is to configure the router interface which hosts SMB clients with a command like this:

```
ip helper-address x.x.x.x
```

where x.x.x.x is the IP address of the SMB server.

13.Acknowledgements

Special thanks to Andrew Tridgell (tridge@linuxcare.com) for starting and directing the Samba project and for keeping this document honest.

Brad Marshall (bmarshall@plugged.net.au) and Jason Parker (jparker@plugged.net.au) contributed time, patience, scripting and research.

Adam Neat (adamneat@ipax.com.au) and Dan Tager (dtager@marsala.com) contributed the bash scripts used to back up Windows machines to a Linux host.

Matthew Flint () told me about the use of the 'interfaces' option in smb.conf.

Oleg L. Machulskiy (machulsk@shade.msu.ru), Jeff Stern (jstern@eclectic.ss.uci.edu), Dr. Michael Langner (langner@fiz−chemie.de and Erik Ratcliffe (erik@caldera.com) suggested modifications to the section on Sharing A Linux Printer With Windows Machines.

Alberto Menegazzi (flash.egon@iol.it) contributed the MagicFilter setup to enable a Linux machine to share a Windows printer.

Rakesh Bharania (rbharani@cisco.com) contributed the suggestion for Cisco router configuration.

Rich Gregory (rtg2t@virginia.edu) and others suggested that this document show some details about the smbfs package and its use.

Andrea Girotto (icarus@inca.dei.unipd.it) contributed a number of valuable suggestions throughout the document.

Thanks, also, to all of the international translators that have brought this HOWTO to the non−English speaking world: Takeo Nakano (nakano@apm.seikei.ac.jp), Klaus−Dieter Schumacher (Klaus−Dieter.Schumacher@fernuni−hagen.de), Andrea Girotto (icarus@inca.dei.unipd.it), Mathieu Arnold (arn_mat@club−internet.fr), Stein Oddvar Rasmussen (Stein@kongsberg−energi.no) Nilo Menezes (nmenezes@n3.com.br) and many others for whom I don't have contact details.