# SMB HOWTO

David Wood, dwood@plugged.net.au v1.0, 10 August 1996

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

## Contents



## 1 Introduction

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

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.

The SMB protocol is used by Microsoft Windows 3.11, NT and 95 to share disks and printers. Using the Samba suite of tools by Andrew Tridgell, UNIX (including Linux) machines can share disk and printers with Windows hosts.

There are four things that one can do with Samba:

1. Share a linux drive with Windows machines.
2. Share a Windows drive 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.

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, you may e-mail the author with suggestions for improvement in this document, but the author guarantees nothing. What did you expect? The author is, after all, a consultant...

## 2 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://nimbus.anu.edu.au/pub/tridge/samba/* `<ftp://nimbus.anu.edu.au/pub/tridge/samba/>`
- The linux Printing HOWTO.
- The Print2Win Mini-HOWTO.

## 3 Installation

The latest source version of Samba is available from:

*ftp://nimbus.anu.edu.au/pub/tridge/samba/* `<ftp://nimbus.anu.edu.au/pub/tridge/samba/>`

However, if you have installed the Redhat distribution of linux, you have the option of installing it as a package. Some other distributions also include the Samba binaries.

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 4.

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

Typically, the following Samba binaries are installed in /usr/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)
```

Additionally, a script called 'print' is included with this HOWTO, which serves as a usefull 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
```

# 4 Running The Daemons

The two SMB daemons are /usr/sbin/smbd and /usr/sbin/nmbd.

You can run the Samba daemons from inetd or as stand-alone processes. If you are configuring a permanent file server, they should be run from inetd so that they will be restarted if they die. If you just want to use SMB services occasionally or to assist with systems administration, you can start them with an /etc/rc.d/init.d script or even by hand when you need them.

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 1
```

To run the daemons from the system startup scripts, put the following script in file called /etc/rc.d/init.d/smb 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
```

# 5 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, if a Windows user trys to connect to this share from their Windows machines, they will be connected to their personal home directory. 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
```

# 6 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 = yes
```

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
```

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

# 7 Sharing A Windows Drive With Linux Machines

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 uyse this utility to transfer files between a Windows 'server' and a linux client.

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

```
/usr/sbin/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/sbin/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/sbin/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.

# 8 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.

See the Printing HOWTO to set up local printing.

Since the author uses a printer connected to a Windows NT machine, this section should not be taken as definitive, but merely a suggestion. Anyone with details to share, please send them to dwood@plugged.net.au so this section can be completed.

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!

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.

# 9 Sharing A Windows Printer With Linux Machines

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

a) You must have the proper entries in /etc/printcap and they must correspond to the local directory structure (for the spool directory, etc)

b) 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.

c) 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.

d) 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.

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 speical 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 oreilly which is currently on zimmerman
# 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);
}
```

# 10 Copyright



# 11 Acknowledgements

As soon as you mail me with suggestions, I'll acknowledge you here in the next release.