D

Summary of Samba Daemons and Commands

This appendix is a reference listing of command-line options and other information to help you use the executables that come with Samba distribution.

Samba Distribution Programs

The following sections provide information about the command-line parameters for Samba programs.

smbd

The *smbd* program provides Samba's file and printer services, using one TCP/IP stream and one daemon per client. It is controlled from the default configuration file, *`samba_dir`/lib/smb.conf*, and can be overridden by command-line options.

The configuration file is automatically re-evaluated every minute. If it has changed, most new options are immediately effective. You can force Samba to immediately reload the configuration file if you send a SIGHUP to *smbd*. Reloading the configuration file, however, will not affect any clients that are already connected. To escape this "grandfather" configuration, a client would need to disconnect and reconnect, or the server itself would have to be restarted, forcing all clients to reconnect.

Other signals

To shut down a *smbd* process, send it the termination signal SIGTERM (-15) which allows it to die gracefully instead of a SIGKILL (-9). To increment the debug logging level of *smbd* at runtime, send the program a SIGUSR1 signal. To decrement it at runtime, send the program a SIGUSR2 signal.

Command-line options

`-D` The *smbd* program is run as a daemon. This is the recommended way to use *smbd* (it is also the default action). In addition, *smbd* can also be run from *inetd*.

`-d` *`debuglevel`*
Sets the debug (sometimes called logging) level. The level can range from 0 all the way to 10. Specifying the value on the command line overrides the value specified in the *smb.conf* file. Debug level 0 logs only the most important messages; level 1 is normal; levels 3 and above are primarily for debugging and slow *smbd* considerably.

`-h` Prints command-line usage information for the *smbd* program.

Testing/debugging options

`-a` If this is specified, each new connection to the Samba server will append all logging messages to the log file. This option is the opposite of `-o`, and is the default.

`-i` *`scope`*

This sets a NetBIOS scope identifier. Only machines with the same identifier will communicate with the server. The scope identifier was a predecessor to workgroups, and this option is included only for backwards compatibility.

`-l` *`log_file`*
Send the log messages to somewhere other than the location compiled in or specified in the *smb.conf* file. The default is often */usr/local/samba/var/log.smb*, */usr/samba/var/log.smb,* or */var/log/log.smb*. The first two are strongly discouraged on Linux, where */usr* may be a read-only filesystem.

`-O` *`socket_options`*
This sets the TCP/IP socket options, using the same parameters as the `socket options` configuration option. It is often used for performance tuning and testing.

`-o` This option is the opposite of `-a`. It causes log files to be overwritten when opened. Using this option saves hunting for the right log entries if you are performing a series of tests and inspecting the log file each time.

`-P` This option forces *smbd* not to send any network data out. This option is typically used only by Samba developers.

`-P` This option forces *smbd* not to send any network data out. This option is typically used only by Samba developers.

`-p` *`port_number`*
This sets the TCP/IP port number that the server will accept requests from. Currently, all Microsoft clients send only to the default port: 139.

,appd.28002 Page 361 Friday, November 19, 1999 3:31 PM

`-s` *`configuration_file`*
: Specifies the location of the Samba configuration file. Although the file defaults to */usr/local/samba/lib/smb.conf*, you can override it here on the command line, typically for debugging.

nmbd

The *nmbd* program is Samba's NetBIOS name and browsing daemon. It replies to broadcast NetBIOS over TCP/IP (NBT) name-service requests from SMB clients and optionally to Microsoft's Windows Internet Name Service (WINS) requests. Both of these are versions of the name-to-address lookup required by SMB clients. The broadcast version uses UDP/IP broadcast on the local subnet only, while WINS uses TCP/IP, which may be routed. If running as a WINS server, *nmbd* keeps a current name and address database in the file *wins.dat* in the `samba_dir`*/var/locks* directory.

An active *nmbd* program can also respond to browsing protocol requests used by the Windows Network Neighborhood. Browsing is a combined advertising, service announcement, and active directory protocol. This protocol provides a dynamic directory of servers and the disks and printers that the servers are providing. As with WINS, this was initially done by making UDP/IP broadcasts on the local subnet. Now, with the concept of a local master browser, it is done by making TCP/IP connections to a server. If *nmbd* is acting as a local master browser, it stores the browsing database in the file *browse.dat* in the `samba_dir`*/var/locks* directory.

Signals

Like *smbd*, the *nmbd* program responds to several Unix signals. Sending *nmbd* a SIGHUP signal will cause it to dump the names it knows about to the file *namelist.debug* in the `samba_dir`*/locks* directory and its browsing database to the *browse.dat* file in the same directory. To shut down a *nmbd* process send it a SIGTERM (-15) signal instead of a SIGKILL (-9) to allow it to die gracefully. You can increment the debug logging level of *nmbd* by sending it a SIGUSR1 signal; you can decrement it by sending a SIGUSR2 signal.

Command-line options

`-D` Instructs the *nmbd* program to run as a daemon. This is the recommended way to use *nmbd*. In addition, *nmbd* can also be run from *inetd*.

`-d` *`debuglevel`*
: Sets the debug (sometimes called logging) level. The level can range from 0, all the way to 10. Specifying the value on the command line overrides the value specified in the *smb.conf* file. Debug level 0 logs only the most impor-

tant messages; level 1 is normal; level 3 and above are primarily for debugging, and slow *nmbd* considerably.

`-h` Prints command-line usage information for the *nmbd* program (also `-?`).

Testing/debugging options

`-a` If this is specified, each new connection to the Samba server will append all logging messages to the log file. This option is the opposite of `-o`, and is the default.

`-H` *`hosts_file`*
This option loads a standard *hosts* file for name resolution.

`-i` *`scope`*
This sets a NetBIOS scope identifier. Only machines with the same identifier will communicate with the server. The scope identifier was a predecessor to workgroups, and this option is included only for backward compatibility.

`-l` *`log_file`*
Sends the log messages to somewhere other than the location compiled-in or specified in the *smb.conf* file. The default is often */usr/local/samba/var/log.nmb*, */usr/samba/var/log.nmb,* or */var/log/log.nmb*. The first two are strongly discouraged on Linux, where */usr* may be a read-only filesystem.

`-n` *`NetBIOS_name`*
This option allows you to override the NetBIOS name by which the daemon will advertise itself. Specifying the option on the command line overrides the `netbios name` option in the Samba configuration file.

`-O` *`socket_options`*
This sets the TCP/IP socket options, using the same parameters as the `socket options` configuration option. It is often used for performance tuning and testing.

`-o` This option is the opposite of `-a`. It causes log files to be overwritten when opened. Using this option saves hunting for the right log entries if you are performing a series of tests and inspecting the log file each time.

`-p` *`port_number`*
This sets the UDP/IP port number from which the server will accept requests. Currently, all Microsoft clients send only to the default port: 137.

`-s` *`configuration_file`*
Specifies the location of the Samba configuration file. Although the file defaults to */usr/local/samba/lib/smb.conf*, you can override it here on the command line, typically for debugging.

`-v`
This option prints the current version of Samba.

Samba Startup File

Samba is normally started by running it from your Unix system's *rc* files at boot time. For systems with a System V-like set of */etc/rcN.d* directories, this can be done by placing a suitably named script in the */rc* directory. Usually, the script starting Samba is called *S91samba*, while the script stopping or "killing" Samba is called *K91samba*. On Linux, the usual subdirectory for the scripts is */etc/rc2.d*. On Solaris, the directory is */etc/rc3.d*. For machines with */etc/rc.local* files, you would normally add the following lines to that file:

```
/usr/local/samba/bin/smbd -D
/usr/local/samba/bin/nmbd -D
```

The following example script supports two extra commands, `status` and `restart`, in addition to the normal `start` and `stop` for System V machines:

```
#!/bin/sh
#
# /etc/rc2.d./S91Samba  --manage the SMB server in a System V manner
#
OPTS="-D"
#DEBUG=-d3
PS="ps  ax"
SAMBA_DIR=/usr/local/samba
case "$1" in
'start')
    echo "samba "
    $SAMBA_DIR/bin/smbd $OPTS $DEBUG
    $SAMBA_DIR/bin/nmbd $OPTS $DEBUG
    ;;
'stop')
    echo "Stopping samba"
    $PS | awk '/usr.local.samba.bin/ { print $1}' |\
    xargs kill
    ;;
'status')
    x=`$PS | grep -v grep | grep '$SAMBA_DIR/bin'`
    if [ ! "$x" ]; then
        echo "No samba processes running"
    else
        echo "  PID TT STAT  TIME COMMAND"
        echo "$x"
    fi
    ;;
'restart')
    /etc/rc2.d/S91samba stop
    /etc/rc2.d/S91samba start
    /etc/rc2.d/S91samba status
    ;;
*)
    echo "$0: Usage error -- you must say $0 start,  stop, status or restart ."
    ;;
esac
exit
```

You'll need to set the actual paths and `ps` options to suit the machine you're using. In addition, you might want to add additional commands to tell Samba to reload its *smb.conf* file or dump its *nmbd* tables, depending on your actual needs.

smbsh

The *smbsh* program lets you use a remote Windows share on your Samba server as if the share was a regular Unix directory. When it's run, it provides an extra directory tree under */smb*. Subdirectories of */smb* are servers, and subdirectories of the servers are their individual disk and printer shares. Commands run by *smbsh* treat the */smb* filesystem as if it were local to Unix. This means that you don't need *smbmount* in your kernel to mount Windows filesystems the way you mount with NFS filesystems. However, you do need to configure Samba with the `--with-smbwrappers` option to enable *smbsh*.

Options

`-d` *`debuglevel`*
: Sets the debug (sometimes called logging) level. The level can range from 0, the default, all the way to 10. Debug level 0 logs only the most important messages; level 1 is normal; level 3 and above are primarily for debugging, and slow *smbsh* considerably.

`-l` *`logfile`*
: Sets the name of the logfile to use.

`-P` *`prefix`*
: Sets the root directory to mount the SMB filesystem. The default is */smb*.

`-R` *`resolve order`*
: Sets the resolve order of the name servers. This option is similar to the `resolve order` configuration option, and can take any of the four parameters, `lmhosts`, `host`, `wins`, and `bcast`, in any order.

`-U` *`user`*
: Supports *`user%password.`*

`-W` *`workgroup`*
: Sets the NetBIOS workgroup to which the client will connect.

smbclient

The *smbclient* program is the maid-of-all-work of the Samba suite. Initially intended as a testing tool, it has become a full command-line Unix client, with an

FTP-like interactive client. Some of its options are still used for testing and tuning, and it makes a simple tool for ensuring that Samba is running on a server.

It's convenient to look at *smbclient* as a suite of programs:

- FTP-like interactive file transfer program
- Interactive printing program
- Interactive tar program
- Command-line message program
- Command-line *tar* program (but see *smbtar* later)
- "What services do you have" query program
- Command-line debugging program

General command-line options

The program has the usual set of *smbd*-like options, which apply to all the interactive and command-line use. The syntax is:

```
smbclient //server_name/share_name [password] [-options]
```

Here is an explanation of each of the command-line options:

`-d` *`debug_level`*
: Sets the debug (logging) level, from 0 to 10, with `A` for all. Overrides the value in *smb.conf*. Debug level 0 logs only the most important messages; level 1 is normal; debug level 3 and above are for debugging, and slow *smbclient* considerably.

`-h` Prints the command-line help information (usage) for smbclient.

`-n` *`NetBIOS_name`*
: Allows you to override the NetBIOS name by which the program will advertise itself.

Smbclient operations

Running `smbclient //`*`server_name`*`/`*`share`* will cause it to prompt you for a username and password. If the login is successful, it will connect to the share and give you a prompt much like an FTP prompt (the backslash in the prompt will be replaced by the current directory within the share as you move around the filesystem):

```
smb:\>
```

From this command line, you can use several FTP-like commands, as listed in Table D-1. Arguments in square brackets are optional.

Table D-1. smbclient Commands

Command	Description
`?` *`command`*	Provides list of commands or help on specified command.
`help` [*`command`*]	Provides list of commands or help on specified command.
`!` [*`command`*]	If a command is specified, it will be run in a local shell. If not, you will be placed into a local shell on the client.
`dir` [*`filename`*]	Displays any files matching *`filename`* in the current directory on the server, or all files if *`filename`* is omitted.
`ls` [*`filename`*]	Displays any files matching *`filename`* in the current directory on the server, or all files if *`filename`* is omitted.
`cd` [*`directory`*]	If *`directory`* is specified, changes to the specified directory on the remote server. If not, reports the current directory on the remote machine.
`lcd` [*`directory`*]	If *`directory`* is specified, the current directory on the local machine will be changed. If not, the name of the current directory on the local machine will be reported.
`get` *remotefile* [*`localfile`*]	Copies the file *`remotefile`* to the local machine. If a *`localfile`* is specified, uses that name to copy the file to. Treats the file as binary; does *not* do LF to CR/LF conversions.
`put` *localfile* [*`remotefile`*]	Copies *`localfile`* to the remote machine. If a *`remotefile`* is specified, uses that as the name to copy to on the remote server. Treats the file as binary; does *not* do LF to CR/LF conversions.
`mget` *`pattern`*	Gets all files matching *`pattern`* from the remote machine.
`mput` *`pattern`*	Places all local files matching *`pattern`* on the remote machine.
`prompt`	Toggles interactive prompting on and off for `mget` and `mput`.
`lowercase ON` (or `OFF`)	If lowercase is on, *smbclient* will convert filenames to lowercase during an `mget` or `get` (but not a `mput` or `put`).
`del` *`filename`*	Delete a file on the remote machine.
`md` *`directory`*	Create a directory on the remote machine.
`mkdir` *`directory`*	Create a directory on the remote machine.
`rd` *`directory`*	Remove the specified directory on the remote machine.
`rmdir` *`directory`*	Remove the specified directory on the remote machine.
`setmode` *`filename`* `[+|-]rsha`	Set DOS filesystem attribute bits, using Unix-like modes. `r` is read-only, `s` is system, `h` is hidden, and `a` is archive.
`exit`	Exits *smbclient.*
`quit`	Exits *smbclient.*

There are also mask and recursive commands for large copies; see the *smbclient* manual page for details on how to use these. With the exception of mask, recursive, and the lack of an ASCII transfer mode, *smbclient* works exactly the same as FTP. Note that because it does binary transfers, Windows files copied to Unix will have lines ending in carriage-return and linefeed (`\r\n`), not Unix's linefeed (`\n`).

Printing commands

The *smbclient* program can also be used for access to a printer by connecting to a print share. Once connected, the commands shown in Table D-2 can be used to print.

Table D-2. smbclient Printing Commands

Command	Description
`print` *`filename`*	Prints the file by copying it from the local machine to the remote one and then submitting it as a print job there.
`printmode` *`text`* `|` *`graphics`*	Instructs the server that the following files will be plain text (ASCII) or the binary graphics format that the printer requires. It's up to the user to ensure that the file is indeed the right kind.
`queue`	Displays the queue for the print share you're connected to, showing job ID, name, size, and status.

Finally, to print from the *smbclient*, use the `-c` option:

```
cat printfile | smbclient //server/printer_name  -c "print -"
```

Tar commands

smbclient can tar up files from a file share. This is normally done from the command line using the *smbtar* command, but the commands shown in Table D-3 are also available interactively.

Table D-3. smbclient Printing Commands

Command	Description
`tar c|x[IXbgNa]` *`operands`*	Performs a creation or extraction *tar* similar to the command-line program.
`blocksize` *`size`*	Sets the block size to be used by *tar*, in 512-byte blocks.
`tarmode full|inc|reset|noreset`	Makes *tar* pay attention to DOS archive bit for all following commands. In `full` mode (the default), *tar* will back up everything. In `inc` (incremental) mode, *tar* will back up only those files with the archive bit set. In `reset` mode, *tar* will reset the archive bit on all files it backs up. (this requires the share to be writable), and in `noreset` mode the archive bit will not be reset even after the file has been backed up.

Command-line message program options

`-M` *`NetBIOS_machine_name`*
: This option allows you to send immediate messages using the WinPopup protocol to another computer. Once a connection is established, you can type your message, pressing control-D to end. If WinPopup is not running on the receiving machine, the program returns an error.

`-U` *`user`*
: This option allows you to indirectly control the FROM part of the message.

Command-line tar program options

The `-T` (tar), `-D` (starting directory), and `-c` (command) options are used together to tar up files interactively. This is better done with *smbtar*, which will be discussed shortly. We don't recommend using *smbclient* directly as a *tar* program.

`-D` *`initial_directory`*
: Changes to initial directory before starting.

`-c` *`command_string`*
: Passes a command string to the *smbclient* command interpreter, which treats it as a semicolon-separated list of commands to be executed. This is handy to say things such as `tarmode inc`, for example, which forces `smbclient -T` to back up only files with the archive bit set.

`-T` *`command filename`*
: Runs the *tar* driver, which is *gtar* compatible. The two main commands are: `c` (create) and `x` (extract), which may be followed by any of:

  `a` Resets archive bits once files are saved.

  `b` *`size`*
  : Sets blocksize in 512-byte units.

  `g` Backs up only files with the archive bit set.

  `I` *`file`*
  : Includes files and directories (this is the default). Does not do pattern-matching.

  `N` *`filename`*
  : Backs up only those files newer than *`filename.`*

  `q` Does not produce diagnostics.

  `X` *`file`*
  : Excludes files.

Command-line query program

If *smbclient* is run as:

```
smbclient -L server_name
```

it will list the shares and other services that machine provides. This is handy if you don't have *smbwrappers*. It can also be helpful as a testing program in its own right.

Command-line debugging/diagnostic program options

Any of the various modes of operation of *smbclient* can be used with the debugging and testing command-line options:

`-B` *`IP_addr`*
: Sets the broadcast address.

`-d` *`debug_level`*
: Sets the debug (sometimes called logging) level. The level can range from 0 all the way to 10. In addition, you can specify `A` for all debugging options. Debug level 0 logs only the most important messages; level 1 is normal; level 3 and above are primarily for debugging and slow operations considerably.

`-E` Sends all messages to stderr instead of stdout.

`-I` *`IP_address`*
: Sets the IP address of the server to which it connects.

`-i` *`scope`*
: This sets a NetBIOS scope identifier. Only machines with the same identifier will communicate with the server. The scope identifier was a predecessor to workgroups, and this option is included only for backward compatibility.

`-l` *`log_file`*
: Sends the log messages to the specified file.

`-N` Suppresses the password prompt. Unless a password is specified on the command line or this parameter is specified, the client will prompt for a password.

`-n` *`NetBIOS_name`*
: This option allows you to override the NetBIOS name by which the daemon will advertise itself.

`-O` *`socket_options`*
: Sets the TCP/IP socket options using the same parameters as the `socket options` configuration option. It is often used for performance tuning and testing.

`-p` *`port_number`*
: Sets the port number from which the client will accept requests.

`-R` *`resolve_order`*
: Sets the resolve order of the name servers. This option is similar to the `resolve order` configuration option, and can take any of the four parameters, `lmhosts`, `host`, `wins`, and `bcast`, in any order .

`-s` *`configuration_file`*
: Specifies the location of the Samba configuration file. Used for debugging.

`-t` *`terminal_code`*
: Sets the terminal code for Asian languages.

`-U` *`username`*
: Sets the username and optionally password (e.g., `-U fred%secret`).

`-W` *`workgroup`*
: Specifies the workgroup that you want the client to connect as.

If you want to test a particular name service, run *smbclient* with `-R` and just the name of the service. This will force *smbclient* to use only the service you gave.

smbstatus

The *smbstatus* program lists the current connections on a Samba server. There are three separate sections. The first section lists various shares that are in use by specific users. The second section lists the locked files that Samba currently has on all of its shares. Finally, the third section lists the amount of memory usage for each of the shares. For example:

```
# smbstatus
Samba version 2.0.3
Service      uid      gid      pid     machine
----------------------------------------------
network      davecb   davecb   7470   phoenix  (192.168.220.101) Sun May 16
network      davecb   davecb   7589   chimaera (192.168.220.102) Sun May 16

Locked files:
Pid    DenyMode   R/W       Oplock           Name
--------------------------------------------------
7589   DENY_NONE  RDONLY   EXCLUSIVE+BATCH  /home/samba/quicken/inet/common/
system/help.bmp   Sun May 16 21:23:40 1999
7470   DENY_WRITE RDONLY    NONE            /home/samba/word/office/findfast.exe
Sun May 16 20:51:08 1999
7589   DENY_WRITE RDONLY   EXCLUSIVE+BATCH  /home/samba/quicken/lfbmp70n.dll
Sun May 16 21:23:39 1999
7589   DENY_WRITE RDWR     EXCLUSIVE+BATCH  /home/samba/quicken/inet/qdata/
runtime.dat   Sun May 16 21:23:41 1999
7470   DENY_WRITE RDONLY   EXCLUSIVE+BATCH  /home/samba/word/office/osa.exe
Sun May 16 20:51:09 1999
7589   DENY_WRITE RDONLY   NONE             /home/samba/quicken/qversion.dll
Sun May 16 21:20:33 1999
```

```
7470   DENY_WRITE RDONLY     NONE            /home/samba/quicken/qversion.dll
Sun May 16 20:51:11 1999

Share mode memory usage (bytes):
   1043432(99%) free + 4312(0%) used + 832(0%) overhead = 1048576(100%) total
```

Options

`-b` Forces *smbstatus* to produce brief output. This includes the version of Samba and auditing information about the users that have logged into the server.

`-d` Gives verbose output, including each of the three reporting sections listed in the previous example. This is the default.

`-L` Forces *smbstatus* to print only the current file locks it has. This corresponds to the second section in a verbose output.

`-p` Prints a list of *smbd* process IDs only. This is often used for scripts.

`-S` Prints only a list of shares and their connections. This corresponds to the first section in a verbose output.

`-s` *`configuration_file`*
Sets the Samba configuration file to use when processing this command.

`-u` *`username`*
Limits the *smbstatus* report to the activity of a single user.

smbtar

The *smbtar* program is a shell script on top of *smbclient* that gives the program more intelligible options when doing tar operations. Functionally, it is equivalent to the Unix *tar* program.

Options

`-a`
Resets the archive bit mode

`-b` *`blocksize`*
Blocking size. Defaults to 20.

`-d` *`directory`*
Changes to initial directory before restoring or backing up files.

`-i` Incremental mode; tar files are backed up only if they have the DOS archive bit set. The archive bit is reset after each file is read.

`-l` *`log_level`*
Sets the logging level.

,appd.28002 Page 372 Friday, November 19, 1999 3:31 PM

-N *filename*
: Backs up only the files newer than the last modification date of *filename*. For incremental backups.

-p *password*
: Specifies the password to use to access a share.

-r Restores files to the share from the tar file.

-s *server*
: Specifies the SMB/CIFS server in which the share resides.

-t *tape*
: Tape device or file. Default is the value of the environment variable $TAPE, or *tar.out* if $TAPE isn't set.

-u *user*
: Specifies the user to connect to the share as. You can specify the password as well, in the format *username%password*.

-v Specifies the use of verbose mode.

-X *file*
: Tells *smbtar* to exclude the specified file from the *tar* create or restore.

-x *share*
: States the share name on the server to connect to. The default is backup, which is a common share name to perform backups with.

For example, a trivial backup command to archive the data for user sue is:

```
# smbtar -s pc_name -x sue -u sue -p secret -t sue.tar
```

nmblookup

The *nmblookup* program is a client program that exercises the NetBIOS-over-UDP/IP name service for resolving NBT machine names into IP addresses. The command works by broadcasting its queries on the local subnet until a machine with that name responds. You can think of it as a Windows *nslookup(1)* or *dig(1)*. This is useful for looking up both normal NetBIOS names, and the odd ones like _ _MSBROWSE_ _ that the Windows name services use to provide directory-like services. If you wish to query for a particular type of NetBIOS name, add the NetBIOS <type> to the end of the name.

The command line is:

```
nmblookup [-options] name
```

The options supported are:

-A Interprets *name* as an IP address and do a node-status query on this address.

-B *broadcast _address*
Sends the query to the given broadcast address. The default is to send the query to the broadcast address of the primary network interface.

-d *debuglevel*
Sets the debug (sometimes called logging) level. The level can range from 0 all the way to 10. Debug level 0 logs only the most important messages; level 1 is normal; level 3 and above are primarily for debugging and slow the program considerably.

-h Prints command-line usage information for the program.

-i *scope*
Sets a NetBIOS scope identifier. Only machines with the same identifier will communicate with the server. The scope identifier was a predecessor to workgroups, and this option is included only for backward compatibility.

-M Searches for a local master browser. This is done with a broadcast searching for a machine that will respond to the special name `_ _MSBROWSE_ _`, and then asking that machine for information, instead of broadcasting the query itself.

-R Sets the recursion desired bit in the packet. This will cause the machine that responds to try to do a WINS lookup and return the address and any other information the WINS server has saved.

-r Use the root port of 137 for Windows 95 machines.

-S Once the name query has returned an IP address, does a node status query as well. This returns all the resource types that the machine knows about, with their numeric attributes. For example:

```
% nmblookup -d 4 -S elsbeth
received 6 names
      ELSBETH              <00> - <GROUP> B <ACTIVE>
      ELSBETH              <03> -         B <ACTIVE>
      ELSBETH              <1d> -         B <ACTIVE>
      ELSBETH              <1e> - <GROUP> B <ACTIVE>
      ELSBETH              <20> -         B <ACTIVE>
      .._ _MSBROWSE_ _..   <01> - <GROUP> B <ACTIVE>
```

-s *configuration_file*
Specifies the location of the Samba configuration file. Although the file defaults to */usr/local/samba/lib/smb.conf*, you can override it here on the command-line, normally for debugging.

-T This option can be used to translate IP addresses into resolved names.

-U *unicast_address*
Performs a unicast query to the specified address. Used with -R to query WINS servers.

Note that there is no workgroup option for *nmblookup*; you can get around this by putting `workgroup = `*`workgroup_name`* in a file and passing it to *nmblookup* with the `-s` *`smb.conf_file`* option.

smbpasswd

The *smbpasswd* password has two distinct sets of functions. When run by users, it changes their encrypted passwords. When run by `root`, it updates the encrypted password file. When run by an ordinary user with no options, it connects to the primary domain controller and changes his or her Windows password.

The program will fail if *smbd* is not operating, if the `hosts allow` or `hosts deny` configuration options will not permit connections from localhost (IP address 127.0. 0.1), or the `encrypted passwords = no` option is set.

Regular user options

`-D` *`debug_level`*
: Sets the debug (also called logging) level. The level can range from 0 to 10. Debug level 0 logs only the most important messages; level 1 is normal; level 3 and above are primarily for debugging and slow the program considerably.

`-h` Prints command-line usage information for the program.

`-r` *`remote_machine_name`*
: Specifies on which machine the password should change. The remote machine must be a primary domain controller (PDC).

`-R` *`resolve_order`*
: Sets the resolve order of the name servers. This option is similar to the `resolve order` configuration option, and can take any of the four parameters, `lmhosts`, `host`, `wins`, and `bcast`, in any order.

`-U` *`username`*
: Used only with `-r`, to modify a username that is spelled differently on the remote machine.

Root-only options

`-a` *`username`*
: Adds a user to the encrypted password file.

`-d` *`username`*
: Disables a user in the encrypted password file.

`-e` *`username`*
: Enables a disabled user in the encrypted password file.

`-m` *machine_name*
: Changes a machine account's password. The machine accounts are used to authenticate machines when they connect to a primary or backup domain controller.

`-j` *domain_name*
: Adds a Samba server to a Windows NT Domain.

`-n` Sets no password for the user.

`-s` *username*
: Causes *smbpasswd* to be silent and to read its old and new passwords from standard input, rather than from */dev/tty*. This is useful for writing scripts.

testparm

The *testparm* program checks an *smb.conf* file for obvious errors and self-consistency. Its command line is:

```
testparm [options] configfile_name [hostname IP_addr]
```

If the configuration file is not specified, the file at *`samba_dir/lib/smb.conf`* is checked by default. If you specify a hostname and an IP address, an extra check will be made to ensure that the specified machine would be allowed to connect to Samba. If a hostname is specified, an IP address should be present as well.

Options

`-h` Prints command-line information for the program.

`-L` *server_name*
: Resets the `%L` configuration variable to the specified server name.

`-s` This option prevents the *testparm* program from prompting the user to press the Enter key before printing a list of the configuration options for the server.

testprns

The *testprns* program checks a specified printer name against the system printer capabilities (*printcap*) file. Its command line is:

```
testprns printername [printcapname]
```

If the `printcapname` isn't specified, Samba attempts to use one located in the *smb.conf* file. If one isn't specified there, Samba will try */etc/printcap*. If that fails, the program will generate an error.

rpcclient

This is a new client that exercises the RPC (remote procedure call) interfaces of an SMB server. Like *smbclient*, *rpcclient* started its life as a test program for the Samba developers and will likely stay that way for a while. Its command line is:

```
rpcclient //server/share
```

The command-line options are the same as the Samba 2.0 *smbclient*, and the operations you can try are listed in Table D-4.

Table D-4. rpcclient commands

Command	Description
`regenum keyname`	Registry Enumeration (keys, values)
`regdeletekey keyname`	Registry Key Delete
`regcreatekey keyname [keyvalue]`	Registry Key Create
`regquerykey keyname`	Registry Key Query
`regdeleteval valname`	Registry Value Delete
`regcreateval valname valtype value`	Registry Key Create
`reggetsec keyname`	Registry Key Security
`regtestsec keyname`	Test Registry Key Security
`ntlogin [username] [password]`	NT Domain Login Test
`wksinfo`	Workstation Query Info
`srvinfo`	Server Query Info
`srvsessions`	List Sessions on a Server
`srvshares`	List shares on a server
`srvconnections`	List connections on a server
`srvfiles`	List files on a server
`lsaquery`	Query Info Policy (domain member or server)
`lookupsids`	Resolve names from SIDs
`ntpass`	NT SAM Password Change

tcpdump

The *tcpdump* utility, a classic system administration tool, dumps all the packet headers it sees on an interface that match an expression. The version included in the Samba distribution is enhanced to understand the SMB protocol. The *expression* is a logical expression with “and,” “or,” and “not,” although sometimes it’s very simple. For example, `host escrime` would select every packet going to or from `escrime`. The expression is normally one or more of:

- `host` *`name`*
- `net` *`network_number`*
- `port` *`number`*
- `src` *`name`*
- `dst` *`name`*

The most common options are `src` (source), `dst` (destination), and `port`. For example, in the book we used the command:

```
tcpdump port not telnet
```

This dumps all the packets except telnet; we were logged-in via telnet and wanted to see only the SMB packets.

Another *tcpdump* example is selecting traffic between server and either `sue` or `joe`:

```
tcpdump host server and \( sue or joe \)
```

We recommend using the `-s 1500` option so that you capture all of the SMB messages sent, instead of just the header information.

Options

There are many options, and many other kinds of expressions that can be used with *tcpdump*. See the manual page for details on the advanced options. The most common options are as follows:

`-c` *`count`*
  Forces the program to exit after receiving the specified number of packets.

`-F` *`file`*
  Reads the expression from the specified file and ignores expressions on the command line.

`-i` *`interface`*
  Forces the program to listen on the specified interface.

`-r` *`file`*
  Reads packets from the specified file (captured with `-w`).

`-s` *`length`*
  Saves the specified number of bytes of data from each packet (rather than 68 bytes).

`-w` *`file`*
  Writes the packets to the specified file.