,ch02.26865 Page 31 Friday, November 19, 1999 3:28 PM

2

Installing Samba on a Unix System

Now that you know what Samba can do for you and your users, it's time to get your own network set up. Let's start with the installation of Samba itself on a Unix system. When dancing the samba, one learns by taking small steps. It's just the same when installing Samba; we need to teach it step by step. This chapter will help you to start off on the right foot.

For illustrative purposes, we will be installing the 2.0.4 version of the Samba server on a Linux* system running version 2.0.31 of the kernel. However, the installation steps are the same for all of the platforms that Samba supports. A typical installation will take about an hour to complete, including downloading the source files and compiling them, setting up the configuration files, and testing the server.

Here is an overview of the steps:

1. Download the source or binary files.
2. Read the installation documentation.
3. Configure a makefile.
4. Compile the server code.
5. Install the server files.
6. Create a Samba configuration file.
7. Test the configuration file.
8. Start the Samba daemons.
9. Test the Samba daemons.

* If you haven't heard of Linux yet, then you're in for a treat. Linux is a freely distributed Unix-like operating system that runs on the Intel x86, Motorola PowerPC, and Sun Sparc platforms. The operating system is relatively easy to configure, extremely robust, and is gaining in popularity. You can get more information on the Linux operating system at *http://www.linux.org/*.

,ch02.26865 Page 32 Friday, November 19, 1999 3:28 PM

Downloading the Samba Distribution

If you want to get started quickly, the CD-ROM packaged with this book contains both the sources and binaries of Samba that were available as this book went to print. The CD is a mirror image of the files and directories on the Samba download server: *ftp.samba.org*.

On the other hand, if you want to download the latest version, the primary web site for the Samba software is *http://www.samba.org*. Once connected to this page, you'll see links to several Samba mirror sites across the world, both for the standard Samba web pages and sites devoted exclusively to downloading Samba. For the best performance, choose a site that is closest to your own geographic location.

The standard Samba web sites have Samba documentation and tutorials, mailing list archives, and the latest Samba news, as well as source and binary distributions of Samba. The download sites (sometimes called *FTP sites*) have only the source and binary distributions. Unless you specifically want an older version of the Samba server or are going to install a binary distribution, download the latest source distribution from the closest mirror site. This distribution is always named:

```
samba-latest.tar.gz
```

If you choose to use the version of Samba that is located on the CD-ROM packaged with this book, you should find the latest Samba distribution in the base directory.

Binary or Source?

Precompiled packages are also available for a large number of Unix platforms. These packages contain binaries for each of the Samba executables as well as the standard Samba documentation. Note that while installing a binary distribution can save you a fair amount of trouble and time, there are a couple of issues that you should keep in mind when deciding whether to use the binary or compile the source yourself:

- The binary packages can lag behind the latest version of the software by one or two (maybe more) minor releases, especially after a series of small changes and for less popular platforms. Compare the release notes for the source and binary packages to make sure that there aren't any new features that you need on your platform. This is especially true of the sources and binaries on the CD-ROM: at the time this book went to print, they were from the latest production release of Samba. However, development is ongoing, so the beta-test versions on the Internet will be newer.

,ch02.26865 Page 33 Friday, November 19, 1999 3:28 PM

- If you use a precompiled binary, you will need to ensure that you have the correct libraries required by the executables. On some platforms the executables are statically linked so this isn't an issue, but on modern Unix operating systems (e.g., Linux, SGI Irix, Solaris, HP-UX, etc.), libraries are often dynamically linked. This means that the binary looks for the right version of each library on your system, so you may have to install a new version of a library. The *README* file or *makefile* that accompanies the binary distribution should list any special requirements.*

  Many machines with shared libraries come with a nifty tool called *ldd*. This tool will tell you which libraries a specific binary requires and which libraries on the system satisfy that requirement. For example, checking the *smbd* program on our test machine gave us:

  ```
  $ ldd smbd
  libreadline.so.3 => /usr/lib/libreadline.so.3
  libdl.so.2 => /lib/libdl.so.2
  libcrypt.so.1 => /lib/libcrypt.so.1
  libc.so.6 => /lib/libc.so.6
  libtermcap.so.2 => /lib/libtermcap.so.2
  /lib/ld-linux.so.2 => /lib/ld-linux.so.2
  ```

  If there are any incompatibilities between Samba and specific libraries on your machine, the distribution-specific documentation should highlight those.

- Keep in mind that each binary distribution carries preset values about the target platform, such as default directories and configuration option values. Again, check the documentation and the makefile included in the source directory to see which directives and variables were used when the binary was compiled. In some cases, these will not be appropriate for your situation.

  A few configuration items can be reset with command-line options at runtime instead of at compile time. For example, if your binary tries to place any log, lock, or status files in the "wrong" place (for example, in */usr/local*), you can override this without recompiling.

One point worth mentioning is that the Samba source requires an ANSI C compiler. If you are on a platform with a non-ANSI compiler, such as the *cc* compiler on SunOS version 4, you'll have to install an ANSI-compliant compiler such as *gcc* before you do anything else.† If installing a compiler isn't something you want to wrestle with, you can start off with a binary package. However, for the most flexibility and compatibility on your system, we always recommend compiling from the latest source.

---

* This is especially true with programs that use *glibc-2.1* (which comes standard with Red Hat Linux 6). This library caused quite a consternation in the development community when it was released because it was incompatable with previous versions of *glibc*.

† *gcc* binaries are available for almost every modern machine. See *http://www.gnu.org/* for a list of sites with *gcc* and other GNU software.

,ch02.26865 Page 34 Friday, November 19, 1999 3:28 PM

Read the Documentation

This sounds like an obvious thing to say, but there have probably been times where you have uncompressed a package, blindly typed `configure`, `make`, and `make install`, and walked away to get another cup of coffee. We'll be the first to admit that we do that, many more times than we should. It's a bad idea—especially when planning a network with Samba.

Samba 2.0 automatically configures itself prior to compilation. This reduces the likelihood of a machine-specific problem, but there may be an option mentioned in the *README* file that you end up wishing for after Samba's been installed. With both source and binary packages you'll find a large number of documents in the *docs* directory, in a variety of formats. The most important files to look at in the distribution are:

```
WHATSNEW.txt
docs/textdocs/UNIX_INSTALL.txt
```

These files tell you what features you can expect in your Samba distribution, and will highlight common installation problems that you're likely to face. Be sure to look over both of them before you start the compilation process.

Configuring Samba

The source distribution of Samba 2.0 and above doesn't initially have a makefile. Instead, one is generated through a GNU *configure* script, which is located in the *samba-2.0.x/source/* directory. The *configure* script, which must be run as root, takes care of the machine-specific issues of building Samba. However, you still may want to decide on some global options. Global options can be set by passing options on the command-line:

```
# ./configure --with-ssl
```

For example, this will configure the Samba makefile with support for the Secure Sockets Layer (SSL) encryption protocol. If you would like a complete list of options, type the following:

```
#./configure --help
```

Each of these options enable or disable various features. You typically enable a feature by specifying the `--with-`*`feature`* option, which will cause the feature to be compiled and installed. Likewise, if you specify a `--without-`*`feature`* option, the feature will be disabled. As of Samba 2.0.5, each of the following features is disabled by default:

`--with-smbwrapper`
: Include SMB wrapper support, which allows executables on the Unix side to access SMB/CIFS filesystems as if they were regular Unix filesystems. We

,ch02.26865 Page 35 Friday, November 19, 1999 3:28 PM

recommend using this option. However, at this time this book went to press, there were several incompatibilities between the *smbwrapper* package and the GNU *libc* version 2.1, and it would not compile on Red Hat 6.0. Look for more information on these incompatibilities on the Samba home page.

`--with-afs`
: Include support of the Andrew Filesystem from Carnegie Mellon University. If you're going to serve AFS files via Samba, we recommend compiling Samba once first without enabling this feature to ensure that everything runs smoothly. Once that version is working smoothly, recompile Samba with this feature enabled and compare any errors you might receive against the previous setup.

`--with-dfs`
: Include support for DFS, a later version of AFS, used by OSF/1 (Digital Unix). Note that this is *not* the same as Microsoft DFS, which is an entirely different filesystem. Again, we recommend compiling Samba once first without this feature to ensure that everything runs smoothly, then recompile with this feature to compare any errors against the previous setup.

`--with-krb4=`*`base-directory`*
: Include support for Kerberos version 4.0, explicitly specifying the base directory of the distribution. Kerberos is a network security protocol from MIT that uses private key cryptography to provide strong security between nodes. Incidentally, Microsoft has announced that Kerberos 5.0 will be the standard authentication mechanism for Microsoft Windows 2000 (NT 5.0). However, the Kerberos 5.0 authentication mechanisms are quite different from the Kerberos 4.0 security mechanisms. If you have Kerberos version 4 on your system, the Samba team recommends that you upgrade and use the `--with-krb5` option (see the next item). You can find more information on Kerberos at *http://web.mit.edu/kerberos/www*.

`--with-krb5=`*`base-directory`*
: Include support for Kerberos version 5.0, explicitly specifying the base directory of the distribution. Microsoft has announced that Kerberos 5.0 will be the standard authentication mechanism for Microsoft Windows 2000 (NT 5.0). However, there is no guarantee that Microsoft will not extend Kerberos for their own needs in the future. Currently, Samba's Kerberos support only uses a plaintext password interface and not an encrypted one. You can find more information on Kerberos at its home page: *http://web.mit.edu/kerberos/www*.

`--with-automount`
: Include support for automounter, a feature often used on sites that offer NFS.

,ch02.26865 Page 36 Friday, November 19, 1999 3:28 PM

`--with-smbmount`
: Include *smbmount* support, which is for Linux only. This feature wasn't being maintained at the time the book was written, so the Samba team made it an optional feature and provided *smbwrapper* instead. The *smbwrapper* feature works on more Unix platforms than *smbmount*, so you'll usually want to use `--with-smbwrapper` instead of this option.

`--with-pam`
: Include support for pluggable authentication modules (PAM), an authentication feature common in the Linux operating system.

`--with-ldap`
: Include support for the Lightweight Directory Access Protocol (LDAP). A future version of LDAP will be used in the Windows 2000 (NT 5.0) operating system; this Samba support is experimental. LDAP is a flexible client-server directory protocol that can carry information such as certificates and group memberships.*

`--with-nis`
: Include support for getting password-file information from NIS (network yellow pages).

`--with-nisplus`
: Include support for obtaining password-file information from NIS+, the successor to NIS.

`--with-ssl`
: Include experimental support for the Secure Sockets Layer (SSL), which is used to provide encrypted connections from client to server. Appendix A, *Configuring Samba with SSL*, describes setting up Samba with SSL support.

`--with-nisplus-home`
: Include support for locating which server contains a particular user's home directory and telling the client to connect to it. Requires `--with-nis` and, usually, `--with-automounter`.

`--with-mmap`
: Include experimental memory mapping code. This is not required for fast locking, which already uses mmap or System V shared memory.

`--with-syslog`
: Include support for using the SYSLOG utility for logging information generated from the Samba server. There are a couple of Samba configuration options that you can use to enable SYSLOG support; Chapter 4, *Disk Shares*, discusses these options.

---

* By *directory*, we don't mean a directory in a file system, but instead an indexed directory (such as a phone directory). Information is stored and can be easily retrieved in a public LDAP system.

,ch02.26865 Page 37 Friday, November 19, 1999 3:28 PM

`--with-netatalk`
: Include experimental support for interoperating with the (Macintosh) Netatalk file server.

`--with-quotas`
: Include disk-quota support.

Because each of these options is disabled by default, none of these features are essential to Samba. However, you may want to come back and build a modified version of Samba if you discover that you need one at a later time.

In addition, Table 2-1 shows some other parameters that you can give the *configure* script if you wish to store parts of the Samba distribution in different places, perhaps to make use of multiple disks or partitions. Note that the defaults sometimes refer to a prefix specified earlier in the table.

Table 2-1. Additional Configure Options

Option	Meaning	Default
`--prefix=`*`directory`*	Install architecture-independent files at the base directory specified.	*/usr/local/samba*
`--eprefix=`*`directory`*	Install architecture-dependent files at the base directory specified.	*/usr/local/samba*
`--bindir=`*`directory`*	Install user executables in the directory specified.	*eprefix/bin*
`--sbindir=`*`directory`*	Install administrator executables in the directory specified.	*eprefix/bin*
`--libexecdir=`*`directory`*	Install program executables in the directory specified.	*eprefix/libexec*
`--datadir=`*`directory`*	Install read-only architecture independent data in the directory specified.	*prefix/share*
`--libdir=`*`directory`*	Install program libraries in the directory specified.	*eprefix/lib*
`--includedir=`*`directory`*	Install package include files in the directory specified.	*prefix/include*
`--infodir=`*`directory`*	Install additional information files in the directory specified.	*prefix/info*
`--mandir=`*`directory`*	Install manual pages in the directory specified.	*prefix/man*

Again, before running the *configure* script, it is important that you are the root user on the system. Otherwise, you may get a warning such as:

```
configure: warning: running as non-root will disable some tests
```

You don't want any test to be disabled when the Samba makefile is being created; this leaves the potential for errors down the road when compiling or running Samba on your system.

,ch02.26865 Page 38 Friday, November 19, 1999 3:28 PM

Here is a sample execution of the *configure* script, which creates a Samba 2.0.4 makefile for the Linux platform. Note that you must run the configure script in the *source* directory, and that several lines from the middle of the excerpt have been omitted:

```
# cd samba-2.0.4b/source/
# ./configure | tee mylog

loading cache ./config.cache
checking for gcc... (cached) gcc
checking whether the C compiler (gcc -O ) works... yes
checking whether the C compiler (gcc -O ) is a cross-compiler... no
checking whether we are using GNU C... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking for a BSD compatible install... (cached) /usr/bin/install -c

...(content omitted)...

checking configure summary
configure OK
creating ./config.status
creating include/stamp-h
creating Makefile
creating include/config.h
```

In general, any message from *configure* that doesn't begin with the words `checking` or `creating` is an error; it often helps to redirect the output of the configure script to a file so you can quickly search for errors, as we did with the `tee` command above. If there was an error during configuration, more detailed information about it can be found in the *config.log* file, which is written to the local directory by the *configure* script.

If the configuration works, you'll see a `checking configure summary` message followed by a `configure OK` message and four or five file creation messages. So far, so good.... Next step: compiling.

Compiling and Installing Samba

At this point you should be ready to build the Samba executables. Compiling is also easy: in the *source* directory, type `make` on the command line. The *make* utility will produce a stream of explanatory and success messages, beginning with:

```
Using FLAGS = -O -Iinclude ...
```

This build includes compiles for both *smbd* and *nmbd*, and ends in a linking command for *bin/make_printerdef*. For example, here is a sample make of Samba version 2.0.4 on a Linux server:

,ch02.26865 Page 39 Friday, November 19, 1999 3:28 PM


```
# make
Using FLAGS =  -O -Iinclude -I./include -I./ubiqx -I./smbwrapper  -DSMBLOGFILE="/
usr/local/samba/var/log.smb" -DNMBLOGFILE="/usr/local/samba/var/log.nmb" -
DCONFIGFILE="/usr/local/samba/lib/smb.conf" -DLMHOSTSFILE="/usr/local/samba/lib/
lmhosts"   -DSWATDIR="/usr/local/samba/swat" -DSBINDIR="/usr/local/samba/bin" -
DLOCKDIR="/usr/local/samba/var/locks" -DSMBRUN="/usr/local/samba/bin/smbrun" -
DCODEPAGEDIR="/usr/local/samba/lib/codepages" -DDRIVERFILE="/usr/local/samba/lib/
printers.def" -DBINDIR="/usr/local/samba/bin" -DHAVE_INCLUDES_H -DPASSWD_
PROGRAM="/bin/passwd" -DSMB_PASSWD_FILE="/usr/local/samba/private/smbpasswd"
Using FLAGS32 =  -O -Iinclude -I./include -I./ubiqx -I./smbwrapper  -
DSMBLOGFILE="/usr/local/samba/var/log.smb" -DNMBLOGFILE="/usr/local/samba/var/log.
nmb" -DCONFIGFILE="/usr/local/samba/lib/smb.conf" -DLMHOSTSFILE="/usr/local/samba/
lib/lmhosts"   -DSWATDIR="/usr/local/samba/swat" -DSBINDIR="/usr/local/samba/bin"
-DLOCKDIR="/usr/local/samba/var/locks" -DSMBRUN="/usr/local/samba/bin/smbrun" -
DCODEPAGEDIR="/usr/local/samba/lib/codepages" -DDRIVERFILE="/usr/local/samba/lib/
printers.def" -DBINDIR="/usr/local/samba/bin" -DHAVE_INCLUDES_H -DPASSWD_
PROGRAM="/bin/passwd" -DSMB_PASSWD_FILE="/usr/local/samba/private/smbpasswd"
Using LIBS = -lreadline -ldl  -lcrypt -lpam
Compiling smbd/server.c
Compiling smbd/files.c
Compiling smbd/chgpasswd.c

...(content omitted)...

Compiling rpcclient/cmd_samr.c
Compiling rpcclient/cmd_reg.c
Compiling rpcclient/cmd_srvsvc.c
Compiling rpcclient/cmd_netlogon.c
Linking bin/rpcclient
Compiling utils/smbpasswd.c
Linking bin/smbpasswd
Compiling utils/make_smbcodepage.c
Linking bin/make_smbcodepage
Compiling utils/nmblookup.c
Linking bin/nmblookup
Compiling utils/make_printerdef.c
Linking bin/make_printerdef
```

If you encounter problems when compiling, check the Samba documentation to see if it is easily fixable. Another possibility is to search or post to the Samba mailing lists, which are given at the end of Appendix D, *Summary of Samba Daemons and Commands*, and on the Samba home page. Most compilation issues are system specific and almost always easy to overcome.

Now that the files have been compiled, you can install them into the directories you identified with the command:

```
# make install
```

If you happen to be upgrading, your old Samba files will be saved with the extension *.old*, and you can go back to that previous version with the command `make revert`. After doing a `make install`, you should copy the *.old* files (if they exist)

,ch02.26865 Page 40 Friday, November 19, 1999 3:28 PM

to a new location or name. Otherwise, the next time you install Samba, the original *.old* will be overwritten without warning and you could lose your earlier version. If you configured Samba to use the default locations for files, the new files will be installed in the directories listed in Table 2-2. Remember that you need to perform the installation from an account that has write privileges on these target directories; this is typically the root account.

Table 2-2. Samba Installation Directories

Directory	Description
/usr/local/samba	Main tree
/usr/local/samba/bin	Binaries
/usr/local/samba/lib	*smb.conf*, *lmhosts*, configuration files, etc.
/usr/local/samba/man	Samba documentation
/usr/local/samba/private	Samba encrypted password file
/usr/local/samba/swat	SWAT files
/usr/local/samba/var	Samba log files, lock files, browse list info, shared memory files, process ID files

Throughout the remainder of the book, we occasionally refer to the location of the main tree as ***samba_dir***. In most configurations, this is the base directory of the installed Samba package: */usr/local/samba*.

Watch out if you've made */usr* a read-only partition. You will want to put the logs, locks, and password files somewhere else.

Here is the installation that we performed on our machine. You can see that we used */usr/local/samba* as the base directory for the distribution (e.g., ***samba_dir***):

```
# make install
Using FLAGS = -O -Iinclude -I./include -I./ubiqx -I./smbwrapper  -DSMBLOGFILE="/
usr/local/samba/var/log.smb" -DNMBLOGFILE="/usr/local/samba/var/log.nmb" -
DCONFIGFILE="/usr/local/samba/lib/smb.conf" -

...(content omitted)...

The binaries are installed. You may restore the old binaries
(if there were any) using the command "make revert". You may
uninstall the binaries using the command "make uninstallbin"
or "make uninstall" to uninstall binaries, man pages and shell
scripts.

...(content omitted)...
```

,ch02.26865 Page 41 Friday, November 19, 1999 3:28 PM


```
============================================================
The SWAT files have been installed. Remember to read the
README for information on enabling and using SWAT.
============================================================
```

If the last message is about SWAT, you've successfully installed all the files. Congratulations! You now have Samba on your system!

Final Installation Steps

There are a couple of final steps to perform. Specifically, add the Samba Web Administration Tool (SWAT) to the */etc/services* and */etc/inetd.conf* configuration files. SWAT runs as a daemon under *inetd* and provides a forms-based editor in your web browser for creating and modifying SMB configuration files.

1. To add SWAT, add the following line to the end of the */etc/services* file:

   ```
   swat   901/tcp
   ```

2. Add these lines to */etc/inetd.conf*. (Check your *inetd.conf* manual page to see the exact format of the *inetd.conf* file if it differs from the following example.) Don't forget to change the path to the SWAT binary if you installed it in a different location from the default */usr/local/samba*.

   ```
   swat   stream  tcp  nowait.400  root  /usr/local/samba/bin/swat  swat
   ```

And that's pretty much it for the installation. Before you can start up Samba, however, you need to create a configuration file for it.

A Basic Samba Configuration File

The key to configuring Samba is its lone configuration file: *smb.conf*. This configuration file can be very simple or extremely complex, and the rest of this book is devoted to helping you get deeply personal with this file. For now, however, we'll show you how to set up a single file service, which will allow you to fire up the Samba daemons and see that everything is running as it should be. In later chapters, you will see how to configure Samba for more complicated and interesting tasks.

The installation process does not automatically create an *smb.conf* configuration file, although several example files are included in the Samba distribution. To test the server software, though, we'll use the following file. It should be named *smb.conf* and placed in the */usr/local/samba/lib* directory.*

```
[global]
    workgroup = SIMPLE
```

* If you did not compile Samba, but instead downloaded a binary, check with the documentation for the package to find out where it expects the *smb.conf* file. If Samba came preinstalled with your Unix system, there is probably already an *smb.conf* file somewhere on your system.

,ch02.26865 Page 42 Friday, November 19, 1999 3:28 PM


```
[test]
    comment = For testing only, please
    path = /export/samba/test
    read only = no
    guest ok = yes
```

This brief configuration file tells the Samba server to offer the directory */export/samba/test* on the server as an SMB/CIFS share called `test`. The server also becomes part of the named workgroup SIMPLE, which each of the clients must also be a part of. (Use your own workgroup here if you already know what it is.) We'll use the `[test]` share in the next chapter to set up the Windows clients. For now, you can complete the setup by performing the following commands as root on your Unix server:

```
# mkdir /export/samba/test
# chmod 777 /export/samba/test
```

We should point out that in terms of system security, this is the worst setup possible. For the moment, however, we only wish to test Samba, so we'll leave security out of the picture. In addition, there are some encrypted password issues that we will encounter with Windows clients later on, so this setup will afford us the least amount of headaches.

If you are using Windows 98 or Windows NT Service Pack 3 or above, you must add the following entry to the `[global]` section of the Samba configuration file: `encrypt passwords = yes`. In addition, you must use the *smbpassword* program (typically located in */usr/local/samba/bin/*) to reenter the username/password combinations of those users on the Unix server who should be able to access shares into Samba's encrypted client database. For example, if you wanted to allow Unix user `steve` to access shares from an SMB client, you could type: `smbpassword -a steve`. The first time a user is added, the program will output an error saying that the encrypted password database does not exist. Don't worry, it will then create the database for you. Make sure that the username/password combinations that you add to the encrypted database match the usernames and passwords that you intend to use on the Windows client side.

Using SWAT

With Samba 2.0, creating a configuration file is even easier than writing a configuration file by hand. You can use your browser to connect to *http://localhost:901*, and log on as the root account, as shown in Figure 2-1.

,ch02.26865 Page 43 Friday, November 19, 1999 3:28 PM

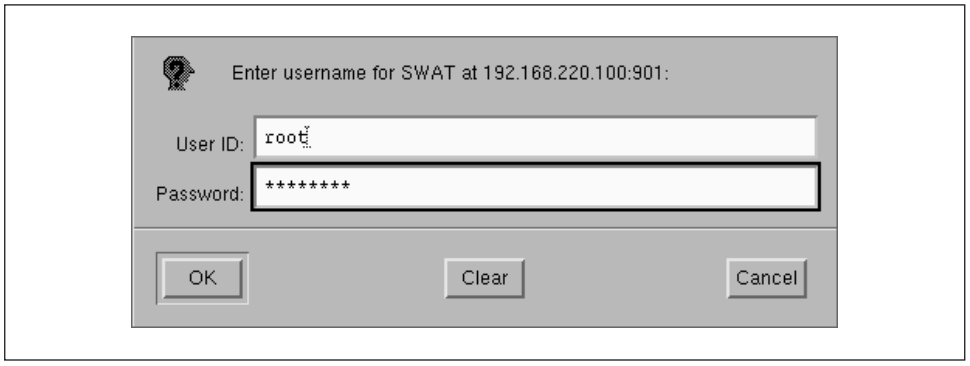

Figure 2-1. SWAT login

After logging in, press the GLOBALS button at the top of the screen. You should see the Global Variables page shown in Figure 2-2.

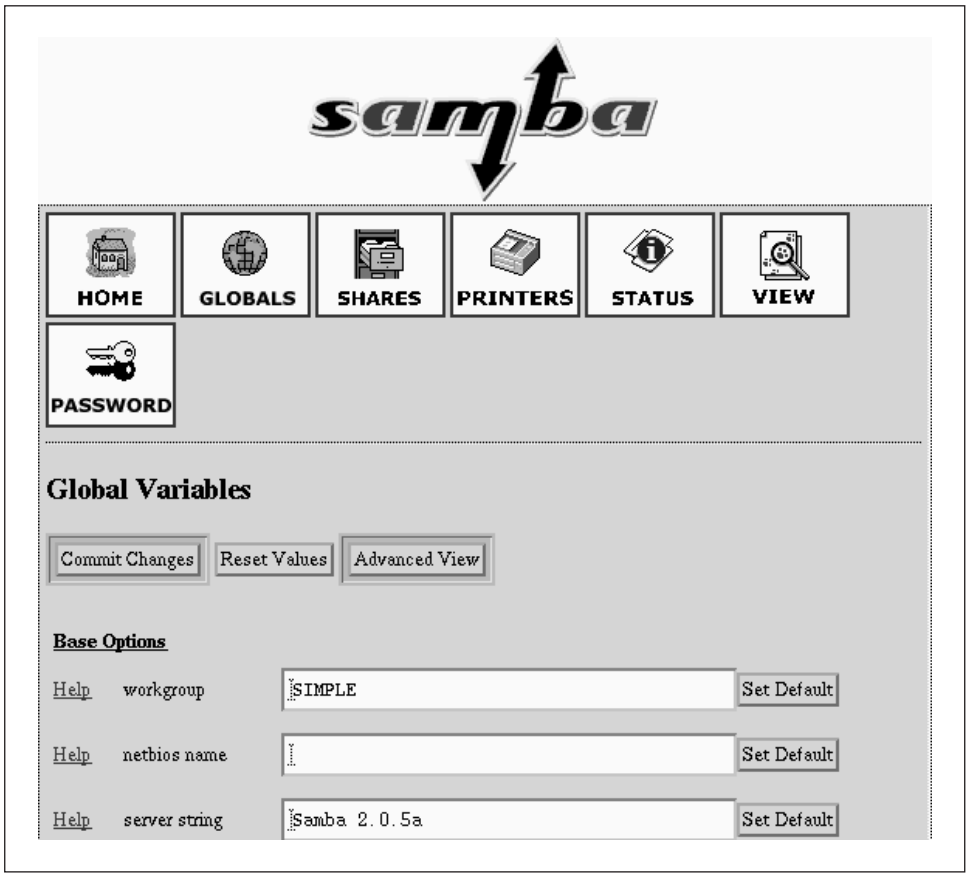

Figure 2-2. SWAT Global Variables page

,ch02.26865 Page 44 Friday, November 19, 1999 3:28 PM

In this example, set the workgroup field to SIMPLE and the security field to USER. The only other option you need to change from the menu is one determining which system on the LAN resolves NetBIOS addresses; this system is called the *WINS server*. At the very bottom of the page, set the wins support field to Yes, unless you already have a WINS server on your network. If you do, put the WINS server's IP address in the wins server field instead. Then return to the top and press the Commit Changes button to write the changes out to the *smb.conf* file.

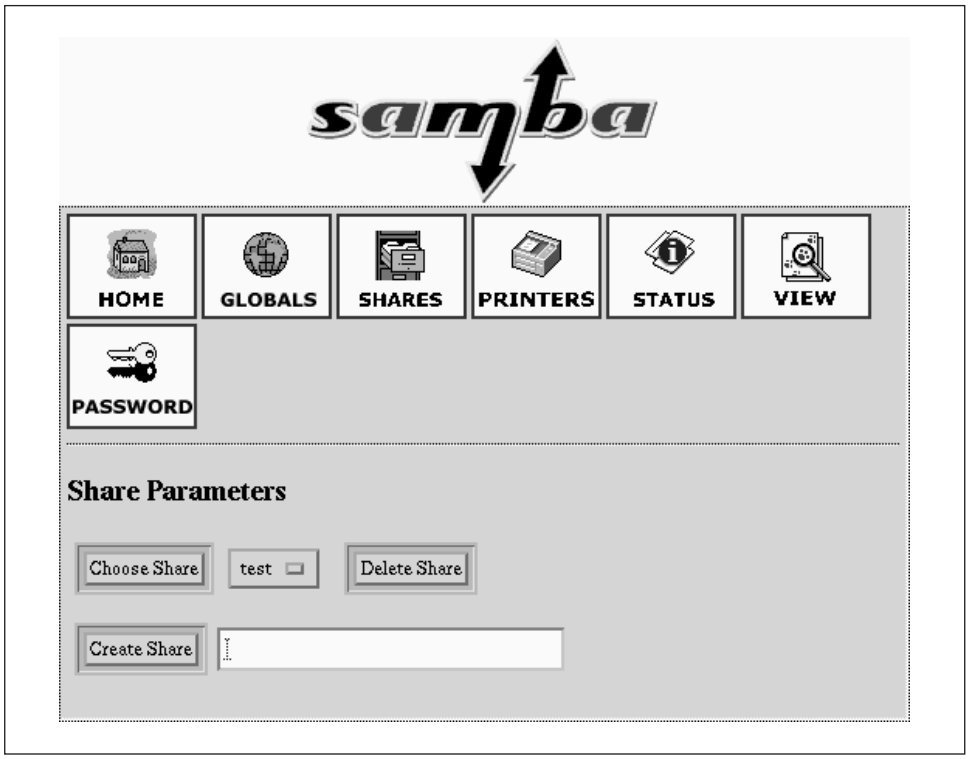

Figure 2-3. SWAT Share Creation screen

Next, press the Shares icon. You should see a page similar to Figure 2-3. Choose Test in the field beside the Choose Share button. You will see the Share Parameters screen, as shown in Figure 2-4. We added a comment to remind us that this is a test share in the *smb.conf* file. SWAT has copies of all that information here.

If you press the View button, SWAT shows you the following *smb.conf* file:

```
# Samba config file created using SWAT
# from localhost (127.0.0.1)
# Date: 1998/11/27 15:42:40

# Global parameters
        workgroup = SIMPLE
```

,ch02.26865 Page 45 Friday, November 19, 1999 3:28 PM

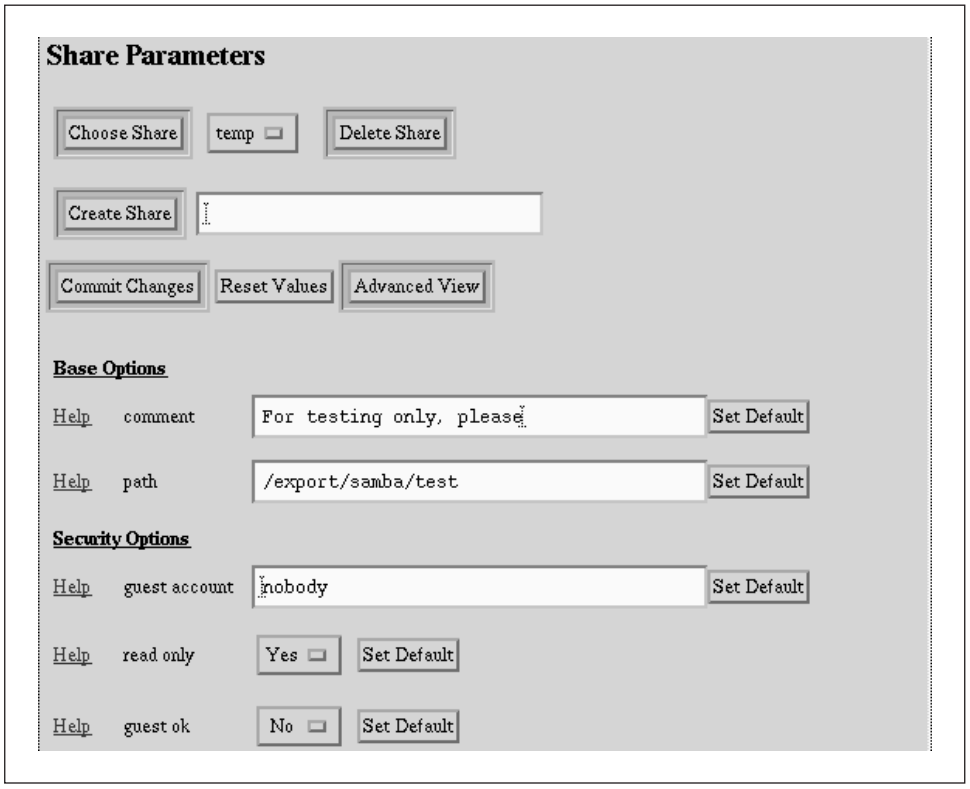

Figure 2-4. SWAT Share Parameters screen

```
[test]
        comment = For testing only, please
        path = /export/samba/test
        read only = no
        guest ok = yes
```

Once this configuration file is completed, you can skip the next step because the output of SWAT is guaranteed to be syntactically correct.

Testing the Configuration File

If you didn't use SWAT to create your configuration file, you should probably test it to ensure that it is syntactically correct. It may seem silly to run a test program against an eight-line configuration file, but it's good practice for the real ones that we'll be writing later on.

The test parser, *testparm*, examines an *smb.conf* file for syntax errors and reports any it finds along with a list of the services enabled on your machine. An example follows; you'll notice that in our haste to get the server running we mistyped

,ch02.26865 Page 46 Friday, November 19, 1999 3:28 PM

`workgroup` as `workgrp` (the output is often lengthy, so we recommend capturing the last parts with the `tee` command):

```
Load smb config files from smb.conf
Unknown parameter encountered: "workgrp"
Ignoring unknown parameter "workgrp"
Processing section "[test]"
Loaded services file OK.
Press enter to see a dump of your service definitions
# Global parameters
[global]
        workgroup = WORKGROUP
        netbios name =
        netbios aliases =
        server string = Samba 2.0.5a
        interfaces =
        bind interfaces only = No

...(content omitted)...

[test]
        comment = For testing only, please
    path = /export/samba/test
        read only = No
        guest ok = Yes
```

The interesting parts are at the top and bottom. The top of the output will flag any syntax errors that you may have made, and the bottom lists the services that the server thinks it should offer. A word of advice: make sure that you and the server have the same expectations.

If everything looks good, then you are ready to fire up the server daemons!

Starting the Samba Daemons

There are two Samba processes, *smbd* and *nmbd*, that need to be running for Samba to work correctly. There are three ways to start:

- By hand
- As stand-alone daemons
- From *inetd*

Starting the Daemons by Hand

If you're in a hurry, you can start the Samba daemons by hand. As root, simply enter the following commands:

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

,ch02.26865 Page 47 Friday, November 19, 1999 3:28 PM

At this point, Samba will be running on your system and will be ready to accept connections.

Stand-alone Daemons

To run the Samba processes as stand-alone daemons, you need to add the commands listed in the previous section to your standard Unix startup scripts. This varies depending on whether you have a BSD-style Unix system or a System V Unix.

BSD Unix

WIth a BSD-style Unix, you need to append the following code to the *rc.local* file, which is typically found in the */etc* or */etc/rc.d* directories:

```
if [ -x /usr/local/samba/bin/smbd]; then
    echo "Starting smbd..."
    /usr/local/samba/bin/smbd -D
    echo "Starting nmbd..."
    /usr/local/samba/bin/nmbd -D
fi
```

This code is very simple; it checks to see if the *smbd* file has execute permissions on it, and if it does, it starts up each of the Samba daemons on system boot.

System V Unix

With System V, things can get a little more complex. System V typically uses scripts to start and stop daemons on the system. Hence, you need to instruct Samba how to operate when it starts and when it stops. You can modify the contents of the */etc/rc.local* directory and add something similar to the following program entitled *smb*:

```
#!/bin/sh

# Contains the "killproc" function on Red Hat Linux
./etc/rc.d/init.d/functions

PATH="/usr/local/samba/bin:$PATH"

case $1 in
    'start')
        echo "Starting smbd..."
        smbd -D
        echo "Starting nmbd..."
        nmbd -D
        ;;
    'stop')
        echo "Stopping smbd and nmbd..."
        killproc smbd
        killproc nmbd
        rm -f /usr/local/samba/var/locks/smbd.pid
```

,ch02.26865 Page 48 Friday, November 19, 1999 3:28 PM


```
            rm -f /usr/local/samba/var/locks/nmbd.pid
            ;;
        *)
            echo "usage: smb {start|stop}"
            ;;
esac
```

With this script, you can start and stop the SMB service with the following commands:

```
# /etc/rc.local/smb start
Starting smbd...
Starting nmbd...
# /etc/rc.local/smb stop
Stopping smbd and nmbd...
```

Starting From Inetd

The *inetd* daemon is a Unix system's Internet "super daemon." It listens on TCP ports defined in */etc/services* and executes the appropriate program for each port, which is defined in */etc/inetd.conf*. The advantage of this scheme is that you can have a large number of daemons ready to answer queries, but they don't all have to be running. Instead, the *inetd* daemon listens in places of all the others. The penalty is a small overhead cost of creating a new daemon process, and the fact that you need to edit two files rather than one to set things up. This is handy if you have only one or two users or your machine has too many daemons already. It's also easier to perform an upgrade without disturbing an existing connection.

If you wish to start from *inetd*, first open */etc/services* in your text editor. If you don't already have them defined, add the following two lines:

```
netbios-ssn     139/tcp
netbios-ns      137/udp
```

Next, edit */etc/inetd.conf*. Look for the following two lines and add them if they don't exist. If you already have `smbd` and `nmbd` lines in the file, edit them to point at the new *smbd* and *nmbd* you've installed. Your brand of Unix may use a slightly different syntax in this file; use the existing entries and the *inetd.conf* manual page as a guide:

```
netbios-ssn stream tcp nowait root /usr/local/samba/bin/smbd smbd
netbios-ns  dgram  udp wait   root /usr/local/samba/bin/nmbd nmbd
```

Finally, kill any *smbd* or *nmbd* processes and send the *inetd* process a hangup (HUP) signal. (The *inetd* daemon rereads its configuration file on a HUP signal.) To do this, use the `ps` command to find its process ID, then signal it with the following command:

```
# kill -HUP process_id
```

After that, Samba should be up and running.

,ch02.26865 Page 49 Friday, November 19, 1999 3:28 PM

Testing the Samba Daemons

It's hard to believe, but we're nearly done with the Samba server setup. All that's left to do is to make sure that everything is working as we think it should. A convenient way to do this is to use the *smbclient* program to examine what the server is offering to the network. If everything is set up properly, you should be able to do the following:

```
# smbclient -U% -L localhost

Added interface ip=192.168.220.100 bcast=192.168.220.255 nmask=255.255.255.0
Domain=[SIMPLE] OS=[Unix] Server=[Samba 2.0.5a]

        Sharename      Type      Comment
        ---------      ----      -------
        test           Disk      For testing only, please
        IPC$           IPC       IPC Service (Samba 2.0.5a)

        Server                   Comment
        ---------                -------
        HYDRA                    Samba 2.0.5a

        Workgroup                Master
        ---------                -------
        SIMPLE                   HYDRA
```

If there is a problem, don't panic! Try to start the daemons manually, and check the system output or the debug files at */usr/local/samba/var/log.smb* to see if you can determine what happened. If you think it may be a more serious problem, skip to Chapter 7, *Printing and Name Resolution*, for help on troubleshooting the Samba daemons.

If it worked, congratulations! You now have successfully set up the Samba server with a disk share. It's a simple one, but we can use it to set up and test the Windows 95 and NT clients in the next chapter. Then we will start making it more interesting by adding services such as home directories, printers, and security, and seeing how to integrate the server into a larger Windows domain.