OS-9 for 68K Samba File Manager

Reference manual

Copyright and Publication Information

This manual reflects version II of OS-9/68k Samba File Manager.

Revision: A
Publication date: December 2000

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, the author will not be liable for any damages including indirect or consequential, from use of the OS-9 Samba File Manager, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction Notice

The software described in this document is intended to be used on a single computer system. Any reproduction of the software on tape, disk, or any other medium except for backup purposes is prohibited.

Trademarks

OS-9 is registered trademark of Microware Systems Corporation. All other product names referenced herein are either trademarks or registered trademarks of their respective owners.

Contacts

Ilja V. Levinson, Yekaterinburg, Russia
E-mail: lev@oduurl.ru
Web: http://members.xoom.com/os9/

Table of Contents

Introduction

This chapter includes the following topics:

- Basic features
- Simple protocol description
- Security model

Basic features

OS-9 Samba File Manager is a software solution that enables OS-9 users to participate in Microsoft networks with file and print services. It provides OS-9 users the ability to mount shared directories or printers on Windows NT (Server and Workstation), Windows 95/98, and Windows for Workgroups computers or compatible systems running TCP/IP.

Samba File Manager gives OS-9 users simultaneous read and write access to files stored on remote computers without first requiring users to download or copy the files to a local drive. Using it, users can mount multiple shared directories and browse for available resources on given server.

The unique feature of Samba File Manager is that it presents the remote file system as it is a local OS-9 file system. This allows an OS-9 user to mount and manipulate remote file system as a normal RBF-like one.

Samba File manager permits printing directly from OS-9 machine to any printer over the network.

Mounting is very similar to Network File System (NFS) mount, but uses a password to prevent unauthorized access to shared resources.

Simple protocol description

Samba File Manager is a client for the SMB (Server Message Block) protocol. All Microsoft Windows family products, OS/2, Unix samba and some others currently use this protocol. With this protocol computers can share files, printers and other type of information, for example lists of available resources.

SMB was born by Microsoft near 1987 and was developed further by Microsoft and others. Recently Microsoft has introduced CIFS - Common Internet File System protocol. At this moment it is just a "substitution" for SMB, or "formerly known" as SMB. In a few words, it is a protocol Windows NT 4.0 uses this time. On the another side, Microsoft provides that the CIFS protocol will be opened and available for all computer users. CIFS 1.0 specification was submitted to the IETF (Internet Engineering Task Force) as an Internet draft document. This means, that SMB/CIFS protocol will be alive in the future for a long time.

SMB/CIFS is a client-server, request-response protocol. Clients connect to servers using transport level protocol, TCP/IP for example (or IPX/SPX). After the connection is established, the client can send commands, also known as SMB messages, to the server. The command set allows client to open files, read and write files, delete them, create directories and so on.

Security model

The SMB defines two models of security:

Share Level

Each resource introduced by a server can have a password, and a client only needs that password to access all files under that resource. This was the first security model SMB had. This model is currently used by Windows for Workgroups and Windows 95/98.

User Level

Protection is applied to each file in each resource and is based on user rights. The client is to be authenticated by a server (using an username and a password). After this is done, the client is given an User ID which it must use on all subsequent accesses to that server. This model is what Windows NT uses this time.

In addition, plain text passwords are never passed across the network. In this case, even using a network sniffer it is impossible to stolen user passwords.

Installing and using OS-9 Samba File Manager

This software is supplied as a single Zip-packed archive file. Just unpack the given archive into directory you like using `unzip` utility. If you do not have it, it is downloadable for free from the os9archive. After unpacking, you need to download the following modules into memory using OS-9 `load` command:

- `samba` file manager object module
- `smbprint` simple print client module
- `smbdrv` driver object module
- `smbmount` the smbmount utility

To do this, change your current data directory to where they reside and issue the following command:

```
load -d samba smbdrv smbmount smbprint
```

After that, you are ready to issue `smbmount` command to mount required network resource.

Technical notes

This chapter quickly describes internal OS-9 Samba File Manager organization, known limitations and possible compatibility problems.

Standard OS-9 utilities

The only commands that do not work with Samba File Manager are those needing information about physical layout of the mounted disk, which are `dcheck`, `format`, `os9gen`, `fsave`, `frestore`, `backup`.

The `attr` utility works, but there are some limitations due to the nature of server's file system, which can be as old as Microsoft DOS. The only file attribute you can change is *S_IWRITE* on the client's side, so on the server's side the file will be marked as read-only.

The Microware's `rename` utility opens the file to be renamed with the access modes as 3 (i.e. *S_IREAD*|*S_IWRITE*). On the other side, Samba servers cannot rename opened files. I hacked `rename` code to make it open files with access modes as 0, but due to the copyright problems cannot distribute the modified version to all the world. If you need this functionality, contact me via e-mail for instructions.

In compare with Microware's PCF (PC File Manager) the following utilities work fine with full functionality: `dsave`, `pd`, `free`.

File permissions

The SMB protocol limitation is the lack of uid, gid and permission information per file. Samba File Manager have to assign those values once for a complete mounted resource (values are inherited from who is `smbmount` running).

File names

Incompatibility between DOS and OS-9 file names is a real problem. The PC file name may have up to 255 characters ("Long File Names" feature is available from Windows 95, before maximum file name length was 12 characters only). OS-9 allows for as many as 28 characters in a file name. However, Samba servers can supply the special short (i.e. 12 characters length) file name for clients those cannot handle the new long one. So, if a DOS file name follows OS-9 length for it, you will see it as it is, otherwise it becomes a bit special (with '~' near the end).

Samba file names are case insensitive. Therefore Samba File Manager follows OS-9 agreement to make the directory names uppercase.

DOS file names can contain characters that are not legal in RBF files. Samba File Manager implements its own file name parsing procedure, so it can handle DOS names successfully, but some standard OS-9 utilities like `dir`, `rename` use kernel's *_os_prsnam()* call directly and will fail on such file names. So, try to avoid such characters for files you plan to use with Samba File Manager.

Besides can emerge problems when DOS file name contains characters from the second half of the ASCII table (codes > 0x7F).

File locking

DOS and OS-9 have a different file locking mechanism. At this moment Samba File Manager supports Exclusive Lock, when opening files with *S_ISHARE* bit set in access mode field (non-sharable files).

Also *I$SetStt/SS_Lock* call is supported with the following limitations:

- This call only can be used to lock/unlock any portion of the file. Read or write of zero bytes (as for Microware's RBF) to unlock is not implemented.

- If a read/write/lock request is issued for a part of a file that is already locked out, Samba File Manager returns immediately with the error code *E$Lock*. You cannot use *I$SetStt/SS_Ticks* to set the delay as RBF does. This occurs because of locking is implemented by the server, not by the client. Possible workaround is to repeat locked request every small period of time (0.5 sec, for example) to retrieve latest locking information from the server, but in this case the network traffic will be increased by far. This also can be done either inside the file manager, or directly by an user application.

- Locking mechanism is implemented on a File basis.

- To ensure better OS-9 compatibility, there is no way to gain file lock using only read or write system calls, although Windows NT based Samba servers permit shared locks, when a record is locked out only for writing (other processes may read locked record as usual).

File buffering

Samba File Manager does not perform file buffering inside itself at all. Instead, it supports large block transfer (up to 50 kB) for user's calls (read and write). You can see this behavior using standard `copy` utility with `-b` option set to 50K, for example.

File descriptors

DOS file system does not have a special file descriptor's storage per file as OS-9 has. Instead, information about a file is stored in a directory entry directly. Samba File Manager has to emulate OS-9 file descriptors (FD's) for all files on mounted volume to make standard utilities happy.

Directories on Samba volumes are always null-sized. When a directory is opened for reading, the value 4096 is used instead. However, `dir` listing still displays the value 0. Due to this behavior, alternate `dir` utilities (like one written by K.Schmitt, available from os9archive) may work in a wrong way.

Also, the SMB protocol does not include such concept as a "current working directory". All requests, which the client dispatches to the server, should include a complete file name (from root level). Thus, it is the file manager's duty to support a special data chunk for all OS-9 processes (but out of the OS-9 process descriptor) to permit them to save information about the current directory using emulated FD (see above).

File times

OS-9 support for file times is a bit poor. It saves the file modification time on one-minute basis, and the file creation time is only the date. On the other hand, Samba server software supports file creation time only from Windows'95 and higher versions. In this case, Samba File Manager try to translate file times as exactly as possible depending on the server software. The additional trouble is Samba server does not have correct date/time for its root directory (i.e. physical root, disk C: on FAT filesystem for example). Currently there is no workaround for this problem.

Also it is impossible to change date/time for directories at all. These values are set only once when directory was created.

Memory usage and possible limitations

Due to necessity to emulate OS-9 like file descriptors as mentioned above, Samba File Manager allocates special chunk of memory (directory cache) for these purposes at startup. The size of this memory depends of emulating FD's count (320 by default, may be changed using special `smbmount` utility option), with 64 bytes for each directory cache entry.

Because remote file system does not have a right directory entry for each file, Samba File Manager needs to emulate it for each *readdir()* call. This memory is allocated once on each *opendir()* operation and occupies 128 (also changeable with `smbmount` utility) directory entries (the maximum count of files in a directory cannot exceed this limit).

Additional memory (about 8 Kbytes) is allocated also at startup for special Samba transfer process's purposes.

Directories can be opened only for reading (you cannot write to a directory at all). *I$GetStat/SS_FDInf* call has no global scope as for Microware's RBF. The FD sector address is valid only for directory path opened at this moment.

Because of Samba server's behavior the directory information may be updated with some delays. The Samba File Manager does not use any flushing to prevent increasing network traffic and overloading Samba servers.

If nothing has been received from a client for a while, the Samba server will assume that the client is no longer running and disconnect the client after configurable amount of time which is about one hour. You must unmount and mount the resource again by hand from the client's side. Also, when you have opened files on a server, it will never disconnect you. Version II (from edition #201) uses a special keep alive timer to prevent the server to drop the connection in this situation.

`smbmount` utility does not check files opened on Samba volume when unmounting. Be sure, that all used files are closed, otherwise it may crash the entire system.

Multiplexing

Version II of OS-9 Samba File Manager allows multiple asynchronous input/output requests to be sent to the server at one time. Typical servers like Windows 95/98 support 2 simultaneous requests, advanced servers like Windows NT - up to 50 requests.

At the same time, there is some unpleasant limitation at use of this feature. Client cannot use large block transfers (see above) and multiplexing at one time. So, if you want just to transfer some large files from/to server, Version I of OS-9 Samba File Manager will be the best

choice. On the other hand, if your applications use mounted disk as a general-purpose storage (open/close/read/write a lot of files simultaneously), Version II will supply the best productivity.

Printing

Version II of OS-9 Samba File Manager includes simple print client. Due to Windows printing implementation the mounted printer device is not an real SCF device from the OS-9 side of view. There are not line-by-line printing features, character substitutions and so on. Windows printing system just spools all client output into special "printer" file, and when the client ends its job (i.e. closes the file descriptor), the server begins to print.

Another limitation is that server does not take care about client output at all. All client data are going directly to the physical printer. So it is the client duty to implement so-called "driver" functions to support different printer models, as well as various tasks like printing text or image, and so on.

In a few words, the Samba file manager gives only "transport" service to transfer user data from an OS-9 machine to the printer in Samba network.

All print jobs supplied with simple print client, are named "Untitled" when viewing print job list (either local from Windows control panel, or remotelly using `smbjobs` script from OS-9 side). This is known limitation in the current version.

The information returned with `smbjobs` script is described below.

Job is a 16-bit integer that uniquely specifies a print job within a printer queue. The job identifier is unique on a server. A combination of the server name and job Id is sufficient to uniquely identify a particular print job.

User specifies the name of the user who submitted the print job (not implemented yet).

Position specifies the position of the print job within the print queue. If the value is 1, this print job prints next.

Date and time fields describe when the user submitted the job.

Size contains an integer that specifies the size of the print job in terms of number of kilobytes.

Status is an integer used as a status flag. The values and meanings of the various bits are:

Tab. 1

Bits	Code	Value	Description
0-1	PRJ_QS_QUEUED	0	Print job is queued
0-1	PRJ_QS_PAUSED	1	Print job is paused
0-1	PRJ_QS_SPOOLING	2	Print job is spooling
0-1	PRJ_QS_PRINTING	3	Print job is printing, bits 2-11 are valid

Tab. 2

Bit	Code	Value	Description
2	PRJ_COMPLETE	0x0004	Print job is complete
3	PRJ_INTERV	0x0008	Internal error occurred
4	PRJ_ERROR	0x0010	Print job is spooling
5	PRJ_DESTOFFLINE	0x0020	The print destination is offline
6	PRJ_DESTPAUSED	0x0040	The print destination is paused
7	PRJ_NOTIFY	0x0080	An alert is raised
8	PRJ_DESTNOPAPER	0x0100	The print destination is out of paper
9	PRJ_DESTFORMCHG	0x0200	The printer is waiting for a form change
10	PRJ_DESTCRTCHG	0x0400	Waiting for a cartridge change
11	PRJ_DESTENCHG	0x0800	Waiting for a pen change
15	PRJ_ PRINTING	0x8000	An alert indicates the job was deleted

The smbmount utility

Because the SMB protocol requires some authentication information, the standard OS-9/NFS `mount` program cannot be used. Instead the special `smbmount` utility is supplied.

smbmount	Mount/unmount remote Samba file system

Syntax: smbmount [</mount-point>] [<resource>] {[opts]}

Function: mount/unmount remote Samba file system, or display current mount information.

`mount-point` is the device name you want to mount the filesystem over. It is the same as in the normal OS-9/NFS `mount` command.

`resource` is the name of the service you want to use on the server. It takes the form `//server/service%user` - where server is the name of the Samba server offering the desired service, service is the name of the service offered, and user means user name as in `-U` option. Thus to connect to the root directory until the service " Disk_C" on the Samba server "NT", you would use the following resource name:

```
//NT//Disk_C
```

To make `smbmount` more friendly, only one leading '/' is required. The `%user` part can be also omitted.

Currently `smbmount` uses *gethostbyname()* to find the IP address of the desired host. It is thus not really compatible with Samba conventions, where the Netbios name of the server is not necessarily the same as the hostname. In environments which enforce a Netbios name that is different than the hostname, you should use `-I` and `-c` to simulate appropriate behavior.

Option	Description
-c=<clientname>	Netbios name of client should be used when Netbios names and hostnames differ and the server refuses login with the default value for the Netbios name, which is your hostname.
-I=<hostname>	Hostname/IP address of the server, should be used when Netbios names and hostnames differ on the server. This sets the hostname to contact on the network, the Netbios name of the server still precedes the service name.
-U=<username>	If the user name your server administrator gave to you differs from your OS-9 based user name, you should use -U to tell the server about your server's user name. Instead default value from environment variable USER is used.
-P=<password>	You may want to give the password required by the server on the command line. You should be careful to use passwords in scripts. If neither -n nor -P are given, smbmount prompts for a password. This makes it difficult to use smbmount in scripts running in batch mode.
-C	By default, passwords are converted to uppercase before they are sent to the server, because most servers require this. You can turn off this conversion by using this option.
-n	Should be given to mount resourses which do not require a password to log in.
-u	Unmount previously mounted resource. Only super-user (with uid 0.0) or user who mounted the resource can unmount it (only mount-point argument required).

Option	Description
`-i`	Display information about mounted device - device name (mount-point), IP address of the server, resource type (Disk, Printer, Interprocess Communication Link), and the full name of the resource (only mount-point argument required).
`-a=<cache-size>`	Set directory cache size (320 by default). If you got *E$NoTask* error code when working with mounted Samba volume, try to increase this value. Do not use here a very large number to minimize memory usage.
`-m=<max-files>`	Set maximum number of files per a directory (128 by default). If the actual number of files in a directory exceeds this value, the rest of the directory will be skipped without any error. This is a good practice to set it to about 40% of the directory cache size.
`-x=<max-requests>`	Set maximal count of simultaneous requests which can be sent to the server at one time (default is 2). This value is server dependent. Windows NT supports up to 50 request other versions of Windows - 2 at one time. Anyhow, in the current version you cannot set this value above 8 to save memory.
`-t`	Synchronize clock with Samba server
`-$`	Connect to IPC$ share
`-p`	Connect to printer share

Option	Description

`-O=<compat-flags>`	Set compatibility options
	
	▫ `0x02` Windows 95/98 has well known bug returning invalid date/time values for directory listings. In many cases OS-9 Samba File Manager automatically determines when this trick have to be used.
	
	▫ `0x04` Do not fix Microware's `free` utility bug for large volumes
	
	▫ `0x08` Do not use large block transfers •
	
	▫ `0x20` Try to handle Windows 95 bug on directory reading (use only if OS-9 `dir` command sporadically returns *E$NoChld* error)
	
	These options may be reiterated in the command line, or concatenated together. For example, the following lines
	
	`smbmount /pc /server/share -O=2 -O=10`
	`smbmount /pc /server/share -O=12`
	
	are identical.

`-f=<filemode>`	Not implemented yet
`-d=<dirmode>`	Not implemented yet
`-e=<eol-character>`	Not implemented yet

• Valid for Version I only

The example session can look like this:

```
** ------------------------------------
** Mount remote resource to device pc0
** Display directory for that tree
** Display a C source file
** Copy remote file onto local device
** Change the current working directory
**       to be inside the mounted volume
** Unmount device
** Mount remote printer (empty password)
** Print file
** ------------------------------------

smbmount /pc0 /server/resource -P=pword

dir /pc0 -e

list /pc0/file.c

copy /pc0/file.c /h0/file2.c

chd /pc0/some_dir

smbmount /pc0 -u

smbmount /lp0 /server/laser -p -n

list sometext.c>/lp0
```

Useful scripts

Samba servers support such concept as "browsing" available resources. Any client can view resource names server supplies to all outside world. OS-9 Samba File Manager installation archive includes a small Microware's `mshell` script `smbview` implementing this feature. It accepts only one command line argument - server name (except if the server is running Windows NT and the server administrator has closed guest account with a password - you also must supply your password as a second argument) and displays all available resources on given server.

`smbstat` script walks through all mounted Samba devices and displays mount information as `smbmount -i` does.

`smbsetime` script synchronizes your computer's clock with the shared clock on given Samba server. Server name is required as command line argument.

`smbjobs` script displays print jobs list and its status on given network printer. Server name and printer name (its share name) are required as command line arguments.