Usage

This module supports the SMB3 family of advanced network protocols (as well as older dialects, originally called "CIFS" or SMB1).

The CIFS VFS module for Linux supports many advanced network filesystem features such as hierarchical DFS like namespace, hardlinks, locking and more. It was designed to comply with the SNIA CIFS Technical Reference (which supersedes the 1992 X/Open SMB Standard) as well as to perform best practice practical interoperability with Windows 2000, Windows XP, Samba and equivalent servers. This code was developed in participation with the Protocol Freedom Information Foundation. CIFS and now SMB3 has now become a defacto standard for interoperating between Macs and Windows and major NAS appliances.

Please see MS-SMB2 (for detailed SMB2/SMB3/SMB3.1.1 protocol specification) or https://samba.org/samba/PFIF/ for more details.

For questions or bug reports please contact:

smfrench@gmail.com

See the project page at: https://wiki.samba.org/index.php/LinuxCIFS_utils

Build instructions

For Linux:

1. Download the kernel (e.g. from https://www.kernel.org) and change directory into the top of the kernel directory tree (e.g. /usr/src/linux-2.5.73)
2. make menuconfig (or make xconfig)
3. select cifs from within the network filesystem choices
4. save and exit
5. make

Installation instructions

If you have built the CIFS vfs as module (successfully) simply type `make modules_install` (or if you prefer, manually copy the file to the modules directory e.g. /lib/modules/2.4.10-4GB/kernel/fs/cifs/cifs.ko).

If you have built the CIFS vfs into the kernel itself, follow the instructions for your distribution on how to install a new kernel (usually you would simply type `make install`).

If you do not have the utility mount.cifs (in the Samba 4.x source tree and on the CIFS VFS web site) copy it to the same directory in which mount helpers reside (usually /sbin). Although the helper software is not required, mount.cifs is recommended. Most distros include a `cifs-utils` package that includes this utility so it is recommended to install this.

Note that running the Winbind pam/nss module (logon service) on all of your Linux clients is useful in mapping Uids and Gids consistently across the domain to the proper network user. The mount.cifs mount helper can be found at cifs-utils.git on git.samba.org

If cifs is built as a module, then the size and number of network buffers and maximum number of simultaneous requests to one server can be configured. Changing these from their defaults is not recommended. By executing modinfo:

```
modinfo kernel/fs/cifs/cifs.ko
```

on kernel/fs/cifs/cifs.ko the list of configuration changes that can be made at module initialization time (by running insmod cifs.ko) can be seen.

Recommendations

To improve security the SMB2.1 dialect or later (usually will get SMB3) is now the new default. To use old dialects (e.g. to mount Windows XP) use "vers=1.0" on mount (or vers=2.0 for Windows Vista). Note that the CIFS (vers=1.0) is much older and less secure than the default dialect SMB3 which includes many advanced security features such as downgrade attack detection and encrypted shares and stronger signing and authentication algorithms. There are additional mount options that may be helpful for SMB3 to get improved POSIX behavior (NB: can use vers=3.0 to force only SMB3, never 2.1):

`mfsymlinks` and either `cifsacl` or `modefromsid` (usually with `idsfromsid`)

Allowing User Mounts

To permit users to mount and unmount over directories they own is possible with the cifs vfs. A way to enable such mounting is to mark the mount.cifs utility as suid (e.g. `chmod +s /sbin/mount.cifs`). To enable users to umount shares they mount requires

1. mount.cifs version 1.4 or later

2. an entry for the share in /etc/fstab indicating that a user may unmount it e.g.:

```
//server/usersharename  /mnt/username cifs user 0 0
```

Note that when the mount.cifs utility is run suid (allowing user mounts), in order to reduce risks, the `nosuid` mount flag is passed in on mount to disallow execution of an suid program mounted on the remote target. When mount is executed as root, nosuid is not passed in by default, and execution of suid programs on the remote target would be enabled by default. This can be changed, as with nfs and other filesystems, by simply specifying `nosuid` among the mount options. For user mounts though to be able to pass the suid flag to mount requires rebuilding mount.cifs with the following flag: CIFS_ALLOW_USR_SUID

There is a corresponding manual page for cifs mounting in the Samba 3.0 and later source tree in docs/manpages/mount.cifs.8

Allowing User Unmounts

To permit users to unmount directories that they have user mounted (see above), the utility umount.cifs may be used. It may be invoked directly, or if umount.cifs is placed in /sbin, umount can invoke the cifs umount helper (at least for most versions of the umount utility) for umount of cifs mounts, unless umount is invoked with -i (which will avoid invoking a umount helper). As with mount.cifs, to enable user unmounts umount.cifs must be marked as suid (e.g. `chmod +s /sbin/umount.cifs`) or equivalent (some distributions allow adding entries to a file to the /etc/permissions file to achieve the equivalent suid effect). For this utility to succeed the target path must be a cifs mount, and the uid of the current user must match the uid of the user who mounted the resource.

Also note that the customary way of allowing user mounts and unmounts is (instead of using mount.cifs and unmount.cifs as suid) to add a line to the file /etc/fstab for each //server/share you wish to mount, but this can become unwieldy when potential mount targets include many or unpredictable UNC names.

Samba Considerations

Most current servers support SMB2.1 and SMB3 which are more secure, but there are useful protocol extensions for the older less secure CIFS dialect, so to get the maximum benefit if mounting using the older dialect (CIFS/SMB1), we recommend using a server that supports the SNIA CIFS Unix Extensions standard (e.g. almost any version of Samba ie version 2.2.5 or later) but the CIFS vfs works fine with a wide variety of CIFS servers. Note that uid, gid and file permissions will display default values if you do not have a server that supports the Unix extensions for CIFS (such as Samba 2.2.5 or later). To enable the Unix CIFS Extensions in the Samba server, add the line:

```
unix extensions = yes
```

to your smb.conf file on the server. Note that the following smb.conf settings are also useful (on the Samba server) when the majority of clients are Unix or Linux:

```
case sensitive = yes
delete readonly = yes
ea support = yes
```

Note that server ea support is required for supporting xattrs from the Linux cifs client, and that EA support is present in later versions of Samba (e.g. 3.0.6 and later (also EA support works in all versions of Windows, at least to shares on NTFS filesystems). Extended Attribute (xattr) support is an optional feature of most Linux filesystems which may require enabling via make menuconfig. Client support for extended attributes (user xattr) can be disabled on a per-mount basis by specifying `nouser_xattr` on mount.

The CIFS client can get and set POSIX ACLs (getfacl, setfacl) to Samba servers version 3.10 and later. Setting POSIX ACLs requires enabling both XATTR and then POSIX support in the CIFS configuration options when building the cifs module. POSIX ACL support can be disabled on a per mount basic by specifying `noacl` on mount.

Some administrators may want to change Samba's smb.conf `map archive` and `create mask` parameters from the default. Unless the create mask is changed newly created files can end up with an unnecessarily restrictive default mode, which may not be what you want, although if the CIFS Unix extensions are enabled on the server and client, subsequent setattr calls (e.g. chmod) can fix the mode. Note that creating special devices (mknod) remotely may require specifying a mkdev function to Samba if you are not using Samba 3.0.6 or later. For more information on these see the manual pages (`man smb.conf`) on the Samba server system. Note that the cifs vfs, unlike the smbfs vfs, does not read the smb.conf on the client system (the few optional settings are passed in on mount via -o parameters instead). Note that Samba 2.2.7 or later includes a fix that allows the CIFS VFS to delete open files (required for strict POSIX compliance). Windows Servers already supported this feature. Samba server does not allow symlinks that refer to files outside of the share, so in Samba versions prior to 3.0.6, most symlinks to files with absolute paths (ie beginning with slash) such as:

```
ln -s /mnt/foo bar
```

would be forbidden. Samba 3.0.6 server or later includes the ability to create such symlinks safely by converting unsafe symlinks (ie symlinks to server files that are outside of the share) to a samba specific format on the server that is ignored by local server applications and non-cifs clients and that will not be traversed by the Samba server). This is opaque to the Linux client application using the cifs vfs. Absolute symlinks will work to Samba 3.0.5 or later, but only for remote clients using the CIFS Unix extensions, and will be invisible to Windows clients and typically will not affect local applications running on the same server as Samba.

Use instructions

Use Instructions

Once the CIFS VFS support is built into the kernel or installed as a module (cifs.ko), you can use mount syntax like the following to access Samba or Mac or Windows servers:

```
mount -t cifs //9.53.216.11/e$ /mnt -o username=myname,password=mypassword
```

Before -o the option -v may be specified to make the mount.cifs mount helper display the mount steps more verbosely. After -o the following commonly used cifs vfs specific options are supported:

```
username=<username>
password=<password>
domain=<domain name>
```

Other cifs mount options are described below. Use of TCP names (in addition to ip addresses) is available if the mount helper (mount.cifs) is installed. If you do not trust the server to which are mounted, or if you do not have cifs signing enabled (and the physical network is insecure), consider use of the standard mount options `noexec` and `nosuid` to reduce the risk of running an altered binary on your local system (downloaded from a hostile server or altered by a hostile router).

Although mounting using format corresponding to the CIFS URL specification is not possible in mount.cifs yet, it is possible to use an alternate format for the server and sharename (which is somewhat similar to NFS style mount syntax) instead of the more widely used UNC format (i.e. \servershare):

```
mount -t cifs tcp_name_of_server:share_name /mnt -o user=myname,pass=mypasswd
```

When using the mount helper mount.cifs, passwords may be specified via alternate mechanisms, instead of specifying it after -o using the normal `pass=` syntax on the command line: 1) By including it in a credential file. Specify credentials=filename as one of the mount options. Credential files contain two lines:

```
username=someuser
password=your_password
```

2. By specifying the password in the PASSWD environment variable (similarly the user name can be taken from the USER environment variable).
3. By specifying the password in a file by name via PASSWD_FILE
4. By specifying the password in a file by file descriptor via PASSWD_FD

If no password is provided, mount.cifs will prompt for password entry

Restrictions

Servers must support either "pure-TCP" (port 445 TCP/IP CIFS connections) or RFC 1001/1002 support for "Netbios-Over-TCP/IP." This is not likely to be a problem as most servers support this.

Valid filenames differ between Windows and Linux. Windows typically restricts filenames which contain certain reserved characters (e.g.the character : which is used to delimit the beginning of a stream name by Windows), while Linux allows a slightly wider set of valid characters in filenames. Windows servers can remap such characters when an explicit mapping is specified in the Server's registry. Samba starting with version 3.10 will allow such filenames (ie those which contain valid Linux characters, which normally would be forbidden for Windows/CIFS semantics) as long as the server is configured for Unix Extensions (and the client has not disabled /proc/fs/cifs/LinuxExtensionsEnabled). In addition the mount option `mapposix` can be used on CIFS (vers=1.0) to force the mapping of illegal Windows/NTFS/SMB characters to a remap range (this mount parameter is the default for SMB3). This remap (`mapposix`) range is also compatible with Mac (and "Services for Mac" on some older Windows).

CIFS VFS Mount Options

A partial list of the supported mount options follows:

username
: The user name to use when trying to establish the CIFS session.

password
: The user password. If the mount helper is installed, the user will be prompted for password if not supplied.

ip
: The ip address of the target server

unc
: The target server Universal Network Name (export) to mount.

domain
: Set the SMB/CIFS workgroup name prepended to the username during CIFS session establishment

forceuid

Set the default uid for inodes to the uid passed in on mount. For mounts to servers which do support the CIFS Unix extensions, such as a properly configured Samba server, the server provides the uid, gid and mode so this parameter should not be specified unless the server and clients uid and gid numbering differ. If the server and client are in the same domain (e.g. running winbind or nss_ldap) and the server supports the Unix Extensions then the uid and gid can be retrieved from the server (and uid and gid would not have to be specified on the mount. For servers which do not support the CIFS Unix extensions, the default uid (and gid) returned on lookup of existing files will be the uid (gid) of the person who executed the mount (root, except when mount.cifs is configured setuid for user mounts) unless the `uid=` (gid) mount option is specified. Also note that permission checks (authorization checks) on accesses to a file occur at the server, but there are cases in which an administrator may want to restrict at the client as well. For those servers which do not report a uid/gid owner (such as Windows), permissions can also be checked at the client, and a crude form of client side permission checking can be enabled by specifying file_mode and dir_mode on the client. (default)

forcegid

(similar to above but for the groupid instead of uid) (default)

noforceuid

Fill in file owner information (uid) by requesting it from the server if possible. With this option, the value given in the uid= option (on mount) will only be used if the server can not support returning uids on inodes.

noforcegid

(similar to above but for the group owner, gid, instead of uid)

uid

Set the default uid for inodes, and indicate to the cifs kernel driver which local user mounted. If the server supports the unix extensions the default uid is not used to fill in the owner fields of inodes (files) unless the `forceuid` parameter is specified.

gid

Set the default gid for inodes (similar to above).

file_mode

If CIFS Unix extensions are not supported by the server this overrides the default mode for file inodes.

fsc

Enable local disk caching using FS-Cache (off by default). This option could be useful to improve performance on a slow link, heavily loaded server and/or network where reading from the disk is faster than reading from the server (over the network). This could also impact scalability positively as the number of calls to the server are reduced. However, local caching is not suitable for all workloads for e.g. read-once type workloads. So, you need to consider carefully your workload/scenario before using this option. Currently, local disk caching is functional for CIFS files opened as read-only.

dir_mode

If CIFS Unix extensions are not supported by the server this overrides the default mode for directory inodes.

port

attempt to contact the server on this tcp port, before trying the usual ports (port 445, then 139).

iocharset

Codepage used to convert local path names to and from Unicode. Unicode is used by default for network path names if the server supports it. If iocharset is not specified then the nls_default specified during the local client kernel build will be used. If server does not support Unicode, this parameter is unused.

rsize

default read size (usually 16K). The client currently can not use rsize larger than CIFSMaxBufSize. CIFSMaxBufSize defaults to 16K and may be changed (from 8K to the maximum kmalloc size allowed by your kernel) at module install time for cifs.ko. Setting CIFSMaxBufSize to a very large value will cause cifs to use more memory and may reduce performance in some cases. To use rsize greater than 127K (the original cifs protocol maximum) also requires that the server support a new Unix Capability flag (for very large read) which some newer servers (e.g. Samba 3.0.26 or later) do. rsize can be set from a minimum of 2048 to a maximum of 130048 (127K or CIFSMaxBufSize, whichever is smaller)

wsize

default write size (default 57344) maximum wsize currently allowed by CIFS is 57344 (fourteen 4096 byte pages)

actimeo=n

attribute cache timeout in seconds (default 1 second). After this timeout, the cifs client requests fresh attribute information from the server. This option allows to tune the attribute cache timeout to suit the workload needs. Shorter timeouts mean better the cache coherency, but increased number of calls to the server. Longer timeouts mean reduced number of calls to the server at the expense of less stricter cache coherency checks (i.e. incorrect attribute cache for a short period of time).

rw

mount the network share read-write (note that the server may still consider the share read-only)

ro

mount network share read-only

version

used to distinguish different versions of the mount helper utility (not typically needed)

sep

if first mount option (after the -o), overrides the comma as the separator between the mount parms. e.g.:

```
-o user=myname,password=mypassword,domain=mydom
```

could be passed instead with period as the separator by:

```
-o sep=.user=myname.password=mypassword.domain=mydom
```

this might be useful when comma is contained within username or password or domain. This option is less important when the cifs mount helper cifs.mount (version 1.1 or later) is used.

nosuid

Do not allow remote executables with the suid bit program to be executed. This is only meaningful for mounts to servers such as Samba which support the CIFS Unix Extensions. If you do not trust the servers in your network (your mount targets) it is recommended that you specify this option for greater security.

exec

Permit execution of binaries on the mount.

noexec

Do not permit execution of binaries on the mount.

dev

Recognize block devices on the remote mount.

nodev

Do not recognize devices on the remote mount.

suid

Allow remote files on this mountpoint with suid enabled to be executed (default for mounts when executed as root, nosuid is default for user mounts).

credentials

Although ignored by the cifs kernel component, it is used by the mount helper, mount.cifs. When mount.cifs is installed it opens and reads the credential file specified in order to obtain the userid and password arguments which are passed to the cifs vfs.

guest

Although ignored by the kernel component, the mount.cifs mount helper will not prompt the user for a password if guest is specified on the mount options. If no password is specified a null password will be used.

perm

Client does permission checks (vfs_permission check of uid and gid of the file against the mode and desired operation), Note that this is in addition to the normal ACL check on the target machine done by the server software. Client permission checking is enabled by default.

noperm

Client does not do permission checks. This can expose files on this mount to access by other users on the local client system. It is typically only needed when the server supports the CIFS Unix Extensions but the UIDs/GIDs on the client and server system do not match closely enough to allow access by the user doing the mount, but it may be useful with non CIFS Unix Extension mounts for cases in which the default mode is specified on the mount but is not to be enforced on the client (e.g. perhaps when MultiUserMount is enabled) Note that this does not affect the normal ACL check on the target machine done by the server software (of the server ACL against the user name provided at mount time).

serverino

Use server's inode numbers instead of generating automatically incrementing inode numbers on the client. Although this will make it easier to spot hardlinked files (as they will have the same inode numbers) and inode numbers may be persistent, note that the server does not guarantee that the inode numbers are unique if multiple server side mounts are exported under a single share (since inode numbers on the servers might not be unique if multiple filesystems are mounted under the same shared higher level directory). Note that some older (e.g. pre-Windows 2000) do not support returning UniqueIDs or the CIFS Unix Extensions equivalent and for those this mount option will have no effect. Exporting cifs mounts under nfsd requires this mount option on the cifs mount. This is now the default if server supports the required network operation.

noserverino

Client generates inode numbers (rather than using the actual one from the server). These inode numbers will vary after unmount or reboot which can confuse some applications, but not all server filesystems support unique inode numbers.

setuids

If the CIFS Unix extensions are negotiated with the server the client will attempt to set the effective uid and gid of the local process on newly created files, directories, and devices (create, mkdir, mknod). If the CIFS Unix Extensions are not negotiated, for newly created files and directories instead of using the default uid and gid specified on the mount, cache the new file's uid and gid locally which means that the uid for the file can change when the inode is reloaded (or the user remounts the share).

nosetuids

The client will not attempt to set the uid and gid on on newly created files, directories, and devices (create, mkdir, mknod) which will result in the server setting the uid and gid to the default (usually the server uid of the user who mounted the share). Letting the server (rather than the client) set the uid and gid is the default. If the CIFS Unix Extensions are not negotiated then the uid and gid for new files will appear to be the uid (gid) of the mounter or the uid (gid) parameter specified on the mount.

netbiosname

When mounting to servers via port 139, specifies the RFC1001 source name to use to represent the client netbios machine name when doing the RFC1001 netbios session initialize.

direct

Do not do inode data caching on files opened on this mount. This precludes mmapping files on this mount. In some cases with fast networks and little or no caching benefits on the client (e.g. when the application is doing large sequential reads bigger than page size without rereading the same data) this can provide better performance than the default behavior which caches reads (readahead) and writes (writebehind) through the local Linux client pagecache if oplock (caching token) is granted and held. Note that direct allows write operations larger than page size to be sent to the server.

strictcache

Use for switching on strict cache mode. In this mode the client read from the cache all the time it has Oplock Level II, otherwise - read from the server. All written data are stored in the cache, but if the client doesn't have Exclusive Oplock, it writes the data to the server.

rwpidforward

Forward pid of a process who opened a file to any read or write operation on that file. This prevent applications like WINE from failing on read and write if we use mandatory brlock style.

acl

Allow setfacl and getfacl to manage posix ACLs if server supports them. (default)

noacl

Do not allow setfacl and getfacl calls on this mount

user_xattr

Allow getting and setting user xattrs (those attributes whose name begins with `user.` or `os2.`) as OS/2 EAs (extended attributes) to the server. This allows support of the setfattr and getfattr utilities. (default)

nouser_xattr

Do not allow getfattr/setfattr to get/set/list xattrs

mapchars

Translate six of the seven reserved characters (not backslash):

```
*?<>|:
```

to the remap range (above 0xF000), which also allows the CIFS client to recognize files created with such characters by Windows's POSIX emulation. This can also be useful when mounting to most versions of Samba (which also forbids creating and opening files whose names contain any of these seven characters). This has no effect if the server does not support Unicode on the wire.

nomapchars

Do not translate any of these seven characters (default).

nocase

Request case insensitive path name matching (case sensitive is the default if the server supports it). (mount option `ignorecase` is identical to `nocase`)

posixpaths

If CIFS Unix extensions are supported, attempt to negotiate posix path name support which allows certain characters forbidden in typical CIFS filenames, without requiring remapping. (default)

noposixpaths

If CIFS Unix extensions are supported, do not request posix path name support (this may cause servers to reject creatingfile with certain reserved characters).

nounix

Disable the CIFS Unix Extensions for this mount (tree connection). This is rarely needed, but it may be useful in order to turn off multiple settings all at once (ie posix acls, posix locks, posix paths, symlink support and retrieving uids/gids/mode from the server) or to work around a bug in server which implement the Unix Extensions.

nobrl

Do not send byte range lock requests to the server. This is necessary for certain applications that break with cifs style mandatory byte range locks (and most cifs servers do not yet support requesting advisory byte range locks).

forcemandatorylock

Even if the server supports posix (advisory) byte range locking, send only mandatory lock requests. For some (presumably rare) applications, originally coded for DOS/Windows, which require Windows style mandatory byte range locking, they may be able to take advantage of this option, forcing the cifs client to only send mandatory locks even if the cifs server would support posix advisory locks. `forcemand` is accepted as a shorter form of this mount option.

nostrictsync

If this mount option is set, when an application does an fsync call then the cifs client does not send an SMB Flush to the server (to force the server to write all dirty data for this file immediately to disk), although cifs still sends all dirty (cached) file data to the server and waits for the server to respond to the write. Since SMB Flush can be very slow, and some servers may be reliable enough (to risk delaying slightly flushing the data to disk on the server), turning on this option may be useful to improve performance for applications that fsync too much, at a small risk of server crash. If this mount option is not set, by default cifs will send an SMB flush request (and wait for a response) on every fsync call.

nodfs

Disable DFS (global name space support) even if the server claims to support it. This can help work around a problem with parsing of DFS paths with Samba server versions 3.0.24 and 3.0.25.

remount

remount the share (often used to change from ro to rw mounts or vice versa)

cifsacl

Report mode bits (e.g. on stat) based on the Windows ACL for the file. (EXPERIMENTAL)

servern

Specify the server 's netbios name (RFC1001 name) to use when attempting to setup a session to the server. This is needed for mounting to some older servers (such as OS/2 or Windows 98 and Windows ME) since they do not support a default server name. A server name can be up to 15 characters long and is usually uppercased.

sfu

When the CIFS Unix Extensions are not negotiated, attempt to create device files and fifos in a format compatible with Services for Unix (SFU). In addition retrieve bits 10-12 of the mode via the SETFILEBITS extended attribute (as SFU does). In the future the bottom 9 bits of the mode also will be emulated using queries of the security descriptor (ACL).

mfsymlinks

Enable support for Minshall+French symlinks (see http://wiki.samba.org/index.php/UNIX_Extensions#Minshall.2BFrench_symlinks) This option is ignored when specified together with the 'sfu' option. Minshall+French symlinks are used even if the server supports the CIFS Unix Extensions.

sign

Must use packet signing (helps avoid unwanted data modification by intermediate systems in the route). Note that signing does not work with lanman or plaintext authentication.

seal

Must seal (encrypt) all data on this mounted share before sending on the network. Requires support for Unix Extensions. Note that this differs from the sign mount option in that it causes encryption of data sent over this mounted share but other shares mounted to the same server are unaffected.

locallease

This option is rarely needed. Fcntl F_SETLEASE is used by some applications such as Samba and NFSv4 server to check to see whether a file is cacheable. CIFS has no way to explicitly request a lease, but can check whether a file is cacheable (oplocked). Unfortunately, even if a file is not oplocked, it could still be cacheable (ie cifs client could grant fcntl leases if no other local processes are using the file) for cases for example such as when the server does not support oplocks and the user is sure that the only updates to the file will be from this client. Specifying this mount option will allow the cifs client to check for leases (only) locally for files which are not oplocked instead of denying leases in that case. (EXPERIMENTAL)

sec

Security mode. Allowed values are:

none
: attempt to connection as a null user (no name)

krb5
: Use Kerberos version 5 authentication

krb5i
: Use Kerberos authentication and packet signing

ntlm
: Use NTLM password hashing (default)

ntlmi
: Use NTLM password hashing with signing (if /proc/fs/cifs/PacketSigningEnabled on or if server requires signing also can be the default)

ntlmv2
: Use NTLMv2 password hashing

ntlmv2i
: Use NTLMv2 password hashing with packet signing

lanman
: (if configured in kernel config) use older lanman hash

hard

Retry file operations if server is not responding

soft

Limit retries to unresponsive servers (usually only one retry) before returning an error. (default)

The mount.cifs mount helper also accepts a few mount options before -o including:

-S	take password from stdin (equivalent to setting the environment variable `PASSWD_FD=0`
-V	print mount.cifs version
-?	display simple usage information

With most 2.6 kernel versions of modutils, the version of the cifs kernel module can be displayed via modinfo.

Misc /proc/fs/cifs Flags and Debug Info

Informational pseudo-files:

DebugData	Displays information about active CIFS sessions and shares, features enabled as well as the cifs.ko version.
Stats	Lists summary resource usage information as well as per share statistics.
open_files	List all the open file handles on all active SMB sessions.

Configuration pseudo-files:

SecurityFlags	Flags which control security negotiation and also packet signing. Authentication (may/must) flags (e.g. for NTLM and/or NTLMv2) may be combined with the signing flags. Specifying two different password hashing mechanisms (as "must use") on the other hand does not make much sense. Default flags are:
	
	`0x07007`
	
	(NTLM, NTLMv2 and packet signing allowed). The maximum allowable flags if you want to allow mounts to servers using weaker password hashes is 0x37037 (lanman, plaintext, ntlm, ntlmv2, signing allowed). Some SecurityFlags require the corresponding menuconfig options to be enabled. Enabling plaintext authentication currently requires also enabling lanman authentication in the security flags because the cifs module only supports sending laintext passwords using the older lanman dialect form of the session setup SMB. (e.g. for authentication using plain text passwords, set the SecurityFlags to 0x30030):
	
	`may use packet signing 0x00001`
	`must use packet signing 0x01001`
	`may use NTLM (most common password hash) 0x00002`
	`must use NTLM 0x02002`
	`may use NTLMv2 0x00004`
	`must use NTLMv2 0x04004`
	`may use Kerberos security 0x00008`
	`must use Kerberos 0x08008`
	`may use lanman (weak) password hash 0x00010`
	`must use lanman password hash 0x10010`
	`may use plaintext passwords 0x00020`
	`must use plaintext passwords 0x20020`
	`(reserved for future packet encryption) 0x00040`

cifsFYI	If set to non-zero value, additional debug information will be logged to the system error log. This field contains three flags controlling different classes of debugging entries. The maximum value it can be set to is 7 which enables all debugging points (default 0). Some debugging statements are not compiled into the cifs kernel unless CONFIG_CIFS_DEBUG2 is enabled in the kernel configuration. cifsFYI may be set to one or nore of the following flags (7 sets them all):
	
	`+-----------------------------------------------+------+`
	`| log cifs informational messages | 0x01 |`
	`+-----------------------------------------------+------+`
	`| log return codes from cifs entry points | 0x02 |`
	`+-----------------------------------------------+------+`
	`| log slow responses | 0x04 |`
	`| (ie which take longer than 1 second) | |`
	`| | |`
	`| CONFIG_CIFS_STATS2 must be enabled in .config | |`
	`+-----------------------------------------------+------+`

traceSMB	If set to one, debug information is logged to the system error log with the start of smb requests and responses (default 0)
LookupCacheEnable	If set to one, inode information is kept cached for one second improving performance of lookups (default 1)
LinuxExtensionsEnabled	If set to one then the client will attempt to use the CIFS "UNIX" extensions which are optional protocol enhancements that allow CIFS servers to return accurate UID/GID information as well as support symbolic links. If you use servers such as Samba that support the CIFS Unix extensions but do not want to use symbolic link support and want to map the uid and gid fields to values supplied at mount (rather than the actual values, then set this to zero. (default 1)
dfscache	List the content of the DFS cache. If set to 0, the client will clear the cache.

These experimental features and tracing can be enabled by changing flags in /proc/fs/cifs (after the cifs module has been installed or built into the kernel, e.g. insmod cifs). To enable a feature set it to 1 e.g. to enable tracing to the kernel message log type:

```
echo 7 > /proc/fs/cifs/cifsFYI
```

cifsFYI functions as a bit mask. Setting it to 1 enables additional kernel logging of various informational messages. 2 enables logging of non-zero SMB return codes while 4 enables logging of requests that take longer than one second to complete (except for byte range lock requests). Setting it to 4 requires CONFIG_CIFS_STATS2 to be set in kernel configuration (.config). Setting it to seven enables all three. Finally, tracing the start of smb requests and responses can be enabled via:

```
echo 1 > /proc/fs/cifs/traceSMB
```

Per share (per client mount) statistics are available in /proc/fs/cifs/Stats. Additional information is available if CONFIG_CIFS_STATS2 is enabled in the kernel configuration (.config). The statistics returned include counters which represent the number of attempted and failed (ie non-zero return code from the server) SMB3 (or cifs) requests grouped by request type (read,

write, close etc.). Also recorded is the total bytes read and bytes written to the server for that share. Note that due to client caching effects this can be less than the number of bytes read and written by the application running on the client. Statistics can be reset to zero by `echo 0 > /proc/fs/cifs/Stats` which may be useful if comparing performance of two different scenarios.

Also note that `cat /proc/fs/cifs/DebugData` will display information about the active sessions and the shares that are mounted.

Enabling Kerberos (extended security) works but requires version 1.2 or later of the helper program cifs.upcall to be present and to be configured in the /etc/request-key.conf file. The cifs.upcall helper program is from the Samba project(https://www.samba.org). NTLM and NTLMv2 and LANMAN support do not require this helper. Note that NTLMv2 security (which does not require the cifs.upcall helper program), instead of using Kerberos, is sufficient for some use cases.

DFS support allows transparent redirection to shares in an MS-DFS name space. In addition, DFS support for target shares which are specified as UNC names which begin with host names (rather than IP addresses) requires a user space helper (such as cifs.upcall) to be present in order to translate host names to ip address, and the user space helper must also be configured in the file /etc/request-key.conf. Samba, Windows servers and many NAS appliances support DFS as a way of constructing a global name space to ease network configuration and improve reliability.

To use cifs Kerberos and DFS support, the Linux keyutils package should be installed and something like the following lines should be added to the /etc/request-key.conf file:

```
create cifs.spnego * * /usr/local/sbin/cifs.upcall %k
create dns_resolver * * /usr/local/sbin/cifs.upcall %k
```

CIFS kernel module parameters

These module parameters can be specified or modified either during the time of module loading or during the runtime by using the interface:

```
/proc/module/cifs/parameters/<param>
```

i.e.:

```
echo "value" > /sys/module/cifs/parameters/<param>
```

1. enable_oplocks	Enable or disable oplocks. Oplocks are enabled by default. [Y/y/1]. To disable use any of [N/n/0].