Linux CIFS Client Guide

Author: Steve French

01/03/07

Version 0.21

Preface

The most recent version of this document is located at
http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.odt
and equivalently
http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf

This is version 0.21 of the document. Feedback can be sent to
linux-cifs-client@lists.samba.org
and/or to the author directly:
sfrench@samba.org

Thanks to Chris Cowan and Shirish Pargaonkar for valuable suggestions and corrections.

Introduction

A network filesytem provides access to files and directories on another computer as if they were on a locally attached disk. The 2.6 version of the Linux kernel introduced a new file system driver cifs.ko to mount to Windows and Samba servers and run demanding applications from those mounts. Although commonly used to mount to non-Linux servers, such as Windows, the cifs virtual file system client [cifs vfs] is optimized for Samba and servers which implement the CIFS POSIX extensions. CIFS is a great alternative to NFS and other network file systems in certain workloads. In earlier versions of Linux the only alternatives for accessing Linux or Windows servers were:

- using the smbclient tool, a user space application with ftp like semantics
- mounting with smbfs, a more limited smb client kernel filesystem driver which provided only partial POSIX semantics
- Installing nfs on the server, and mounting with Linux's nfs client (which does not have an oplock based cache consistency mechanism, and has a more limited security model). Although NFSv4 addresses some of these NFS version 2 and 3 deficiencies, it is not widely deployed, and is unavailable on some platforms.

This new smb/cifs client can be used to mount to a variety of servers which are listed below:

Target Server	***Recommended version of cifs.ko module***	***Comments***
AIX Fast Connect (TM)	Version 1.31 or later	Limited posix semantics
Network Appliance (TM)	Version 1.31 or later	
Samba 2	Version 1.34a or later	Limited posix semantics
Samba 3 and Samba 4	Version 1.31 or later	Best posix semantics since it implements cifs posix extensions (Samba 4 does not implement the posix extensions yet)
Windows Server 2003 (TM)	Version 1.32 or later	Most posix semantics work
Windows XP, Windows	Version 1.31 or later	Most posix semantics work

Target Server	*Recommended version of cifs.ko module*	*Comments*
2000 Server (TM)		
Windows ME (TM)	Version 1.37 or later	Relatively poor posix semantics, and extending file size does not fill with zero
Windows NT Server 4.0 (TM)	Version 1.31 or later	Some posix semantics of delete/rename in particular on open files can not be performed
OS/2 (TM)	Version 1.46 or later	As with NT, lacks some posix semantic

Additional server types, including various popular network appliances, also may work. Note that some specific network appliances are not listed because they are based on Samba server. For those NAS appliances refer to the corresponding Samba server version. The version of the cifs client module can be viewed by:

```
        cat /proc/fs/cifs/DebugData
```

or

```
        /sbin/modinfo /lib/modules/<kernel-version>/kernel/fs/cifs/cifs.ko
```

The CIFS VFS for Linux supports many advanced network filesystem features such as hardlinks, packet signing, 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 advanced cifs servers. It also provides limited compatibility with older SMB servers.

When to use CIFS?

There are times when CIFS can not be used or another network filesystem choice is better. If kerberos (krb5/SPNEGO) authentication support is needed for added security, then Samba's smbclient or smbfs must be used instead of cifs (NFS version 4 will also provide kerberos enabled authentication but is harder to configure for the popular ActiveDirectory Kerberos KDCs) . The support in cifs.ko for Kerberos is expected to be available by or before cifs.ko version 1.48. NTLMv2 can be used as an alternative to Kerberos for stronger CIFS authentication to Sambaservers, and starting in version 1.47 of cifs (Kernel verison 2.6.20), NTLMv2 can be used for mounting to Windows servers as well. For optimal performance, especially on large file reads from a single process, nfs version 3 client for Linux will often exceed the performance of the cifs client, although the performance gap is closing as cifs is improved to dispatch more read requests asynchronously from a single process at one time. In addition there are a few cases in which nfs semantics (especially for NFS version 4) more closely match posix semantics. Some cluster filesystems also offer excellent POSIX semantics and performance (at least for all Linux environments), but are often harder to configure and lack some of the security features of CIFS and NFSv4.

Differences between cifs and the older Linux client smbfs

In versions of the Linux kernel prior to 2.6, an older implementation of an SMB/CIFS client, smbfs was used which will be deprecated starting in Linux kernel 2.6.20. The newer cifs virtual filesystem introduced various new features, but also introduced some changes which can be confusing for former smbfs users. Some commonly encountered differences include:

1) configuration

Unlike smbfs and smbclient, configuration for the cifs vfs does not depend on smb.conf.   CIFS configuration is done using /proc (modifying pseudofiles in /proc/fs/cifs) and by specifying module install parameters (module options passed to /sbin/insmod or modprobe). In addition the cifs vfs does not use the Samba utilities smbmnt or smbmount as smbfs did.   Most of the cifs vfs is implemented in kernel, although cifs vfs has a small userspace mount helper (mount.cifs)

2) mount syntax:

mounting to older servers (those prior to1997) may require specifying two additional fields beyond those which smbfs required:

a) The server's netbios (RFC1001) name AND the server's tcp (or ip) address.   CIFS does not assume, as smbfs did, that the tcp name and the netbios name of the server are the same.   The netbios name of the server is specified by passing the mount option “servern=SERVERNAME” and is not assumed to be necessarily the same as the tcp name of the server (also note that the netbios name is usually capitalized). Both tcp/ip name and netbios name have to be specified for mounts for most old lanman servers. For example:

```
                    mount -t cifs //tcp-name-of-server/sharename   /mnt -o
user=username,sec=lanman,servern=SERVERNAME
```

b) A security option allowing weaker password hashes to be used.   Specifying “sec=lanman” in the mount options allows the client to send weaker lanman hashes to the server.   Older servers such as Windows 98 require this.

Mounting to more current servers with stronger authentication (ntlmv2) typically requires an additional mount option (unless the security flags are changed in /proc/fs/cifs) “sec=ntlmv2”

3) Default UID/GID/mode for remote files

For mounts to servers which can report uids and gids (the number representing the user and group names) such as Samba server, it may be necessary to disable the unix extensions if uids and gids do not match between cifs client and Samba server. For servers which do not support Unix modes directly (e.g. Most versions of Windows), the default mode for files differs from that reported by smbfs. The default mode can be overridden by mount options “file_mode” and “dir_mode”for these server types.

Mount procedures

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

```
  mount -t cifs //9.53.216.11/e$ /mnt -o user=myname,pass=mypassword
```

If the mount helper, mount.cifs, is installed (which is usually the case for most Linux distributions), then a tcp host name rather than ip address may be used. For example,

```
  mount -t cifs //my_server/e$ /mnt -o user=myname,pass=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:

```
  user=<username>
  pass=<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. \\server\share):

```
        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. A complete list of mount options is described in detail in fs/cifs/README in the source code for your kernel.

Automounting cifs is possible with the generic automount module. There is also an experimental auto.smb sample in the automount project.

<u>When a user rather than root mounts</u>

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:

```
      gcc samba/source/client/mount.cifs.c -DCIFS_ALLOW_USR_SUID -o mount.cifs
```

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 ummount 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.

<u>Server Considerations</u>
To get the maximum benefit from the CIFS VFS, we recommend using a server that supports the SNIA CIFS Unix Extensions standard (e.g. Samba 2.2.5 or later or Samba 3.0) 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 invisbile to Windows clients and typically will not affect local applications running on the same server as Samba.

Restrictions

Although limited supported is provided for old SMB servers (“LM1.2X002" and LANMAN2.1) servers, for optimal support servers must support the NTLM SMB/CIFS dialect (which is the most recent, supported by Samba and Windows NT version 4, 2000 and XP and many other SMB/CIFS servers) . This module also supports the newer POSIX CIFS dialect.

Servers must support either "pure-TCP" (port 445 TCP/IP CIFS connections by default) or RFC 1001/1002 support for "Netbios-Over-TCP/IP." Neither of these is likely to be a problem as most servers support this. IPv6 support is planned for the future, and is almost complete.

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).

Bug Reporting

Potential bugs may be reported to bugzilla.samba.org (CIFS VFS component) and/or to the bug management system of your Linux distribution. A list of important changes and fixes to the Linux CIFS module are listed in the file fs/cifs/CHANGES. In addition a list (and short description) of all recent changes can be seen by searching for “CIFS” in the Linux kernel's source code control system:

http://kernel.org/git/?p=linux/kernel/git/torvalds/linux-2.6.git;a=summary

Note that the version of the cifs client code can be seen via

“cat /proc/fs/cifs/DebugData”

or

“/sbin/modinfo cifs.ko”

The version of the cifs mount helper can be displayed by:

“/sbin/mount.cifs -V”

The CIFS project page includes an updated version of the cifs source that can build and run on a wide variety of kernel versions. In addition, many distributions provide updates to their kernel and/or cifs module (cifs.ko).

Debugging and RAS Considerations

The CIFS client exports information at runtime to assist in debugging network problems. The pseudo-file /proc/fs/cifs/DebugData shows information on the status of cifs mounts, sessions, and active network requests (“cat /proc/fs/cifs/DebugData”). The dmesg (Linux message log) will show certain serious errors, but enabling additional cifs debugging flags can increase the debug output. Setting /proc/fs/cifs/DebugData to 1 will cause additional cifs informational messages to be logged, and setting it to 3 (“echo 3 > /proc/fs/cifs/cifsFYI”) will also log return codes from most cifs entry points to dmesg.

One of the most powerful tools for analyzing networking or network filesystem (cifs client or SMB/CIFS server) problems is “Wireshark” (was “ethereal”). Most network analyzers, including wireshark, provide support for analyzing SMB/CIFS network traffic. See http://www.wireshark.org for more details.

- <TBD: Add a demonstration of how you would detect the need for an adjustment on both the server and client side. In other words, example of how to interpret the statistics in /proc/fs/cifs and how they relate to various monitors and/or log entries. A decoder ring for the diligent sys admin.>

Performance Considerations

Server speed, server disk speed and network speed can constrain the overall performance of a cifs mount, but in some cases, client client configuration settings can be changed to increase performance:

1) size of file write (wsize). The Linux CIFS client usually sends 56K writes (14 pages) and is limited to 56K maximum unless mounted forcedirectio.
2) size of file read (rsize). The Linux CIFS client usually sends 16K reads (4 pages).   Since CIFS large network buffers are about 16K in size by default, increasing the rsize would have little effect unless the setting of module load parameter CIFSMaxBufSize (via insmod) also is increased.
3) maximum number of simultaneous requests to a particular server. Default is 50. It is configured via module load parameter (via insmod) "cifs_max_pending"
4) minumum number of small and large network buffers in the cifs buffer pool. These are configured via module load parameters cifs_min_small and cifs_min_rcv respectively. Although increasing these will reduce available memory, they can increase performance for some workloads in which large numbers of simultaneous requests are made to the same server from different processes (by reducing the number of memory allocations).
5) caching on client (forcedirectio). The default is to attempt to cache ie try to request oplock on files opened by the client (forcedirectio is off). Foredirectio also can indirectly alter the network read and write size, since i/o will now match what was requested by the application, as readahead and writebehind is not being performed by the page cache when forcedirectio is enabled for a mount.
6) Sending byte range lock requests. Sending byte range lock requests can be disabled (only enforced locally on the client) by the mount parameter "nobrl"

In addition, whether or not cifs packet signing is negotiated, and whether or not the Linux Extensions to CIFS (POSIX CIFS Extensions) are negotiated, can affect performance.

The Linux CIFS client exports various statistics to assist in tuning. See   /proc/fs/cifs/Stats and /proc/fs/cifs/DebugData.   Additional timing information can be logged to dmesg (the kernel debug message log) by setting the "CIFS_TIMER" flag ("echo 4 > /proc/fs/cifs/cifsFYI").

Global Name Space

Creating a global namespace is possible through a feature of CIFS called "DFS" (not to be confused with DCE/DFS). Use of DFS can provide additional redundancy (high availability) as well as simple load balancing when a replicated resource is available from more than one Samba or Windows server and exported via DFS. Samba servers and Windows Servers and some NAS appliances support DFS (giving DFS referrals to clients, when a subdirectory is located on one or more external servers). Samba's smbclient tool and the Windows clients (since Windows 2000) support following DFS referrals, but the Linux CIFS client does not have sufficient support for DFS yet (it is missing one necessary subfunction which is expected before cifs version 1.50). Some NFS version 4 clients and servers have a similar feature called “fs_locations” (the Linux NFS server code for fs_locations was added into Linux Kernel 2.6.19). As the number of ip addresses continues to grow, the interest in ipv6 (instead of the current ipv4) is increasing Support for Ipv6 in the Linux cifs client is expected by cifs version 1.48.

Security Considerations

Various aspects can be configured:

1) Where ACL checks are performed (at server or at both client and server)
2) Which credentials to use on the server (the userid/password specified on the mounted, or the best match we can find among active mounts for the userid/password of the uid of the current process). This is controlled via /proc/fs/cifs/MultiuserMount
3) How to authenticate (whether to allow or require NTLM, NTLMv2, LANMAN, plaintext, null-user authentication). This is controlled via various flags on /proc/fs/cifs/SecurityFlags or alternatively can be overridden via the mount option “sec=” which specifies the particular authentication method to use to this server.
4) Whether cifs packet signing is required (/proc/fs/cifs/SecurityFlags)

Additional details about security options can be found in the cifs man page or in fs/cifs/README.

Additional Information

The man page for cifs (“man mount.cifs”) includes additional detail about some of the available mount options. The file fs/cifs/README in the kernel source directory may include more recent information on the cifs module. The file fs/cifs/CHANGES includes a summary of the more visible changes to the cifs module by cifs version number.

Appendix A – An overview of CIFS Architecture

Overview

The cifs.ko module is a Linux virtual file system module. It exports a set of entry points to the kernel. The kernel communicates with userspace via libc (the operating system runtime library). The cifs module is tsimilar to a dynamically loadable device driver, and typically is loaded implicitly by any attempt to mount with type cifs (“mount -t cifs”) although it can be loaded explicitly via modprobe or insmod. The cifs module can be unloaded via rmmod (or shutdown of the system).

Mount

When the mount utility is invoked it searches for a mount helper with a matching name to the filesystem type (mount.fstype) which in our case is /sbin/mount.cifs (similarly on umount /sbin/umount.cifs would be invoked, if found, but umount.cifs is not necessary for most purposes). The small helper mount.cifs lightly parses the cifs specific mount options (most importantly translating the host name for the target server into an ip address) then invokes do_mount in libc (the main operating system runtime library) which crosses into the kernel address space and into the virtual filesystem mapping layer of the kernel (vfs).   The vfs layer of the kernel mediates between libc and a filesystem (such as the cifs module).   The filesystem sets up function pointers in key objects in order to facilitate this mapping.

Filesystem objects

Filesystems manage the following objects, most of which export function pointers (which cifs sets up when the object is instantiated in order to point to cifs helper functions)
a) dentries: file names and directory names in the filesystem namespace. A file on disk may have multiple names.
b) inodes: the metadata such as timestamps, mode and attributes which describe a file on disk
c) pages in the page cache: the data in a file
d) file structs: the information about an open file instance, e.g. whether it was opened for read or write or both
e) superblock: the information about a unique server resource which is mounted by this client (the vfs mount includes the information about which local path it is mounted).   A superblock may be mounted over more than one local directory.

CIFS specific objects

Most of the important CIFS structures are defined in fs/cifs/cifsglob.h which includes more detailed structure definitions.   During the first mount to a server (target with a unique ip address), the cifs module creates a tcp socket and a correspoinding cifs TCP Server Info structure.   For each unique user name (mounts can be made more than one time to the same server/share with different credentials) mounted to that server, a cifsSesInfo structure is created.   For each unique share (\\servername\sharename) which is mounted (each unique exported resource on some server that is mounted by this client) a cifsTconInfo structure is created (“Tcon “ stands for the SMB/CIFS term “tree connection”).   Each inode which is accessed (looked up e.g. via stat) causes a cifInodeInfo struct to be created and linked in with the inode structure, and similarly opening a file causes a cifsFileInfo struct to be created an linked in with the filesystem specific area of the file object.

The following diagram (source: http://www.geocities.com/ravikiran_uvs/articles/rkfs.html) illustrates the relationships among the filesystem objects:

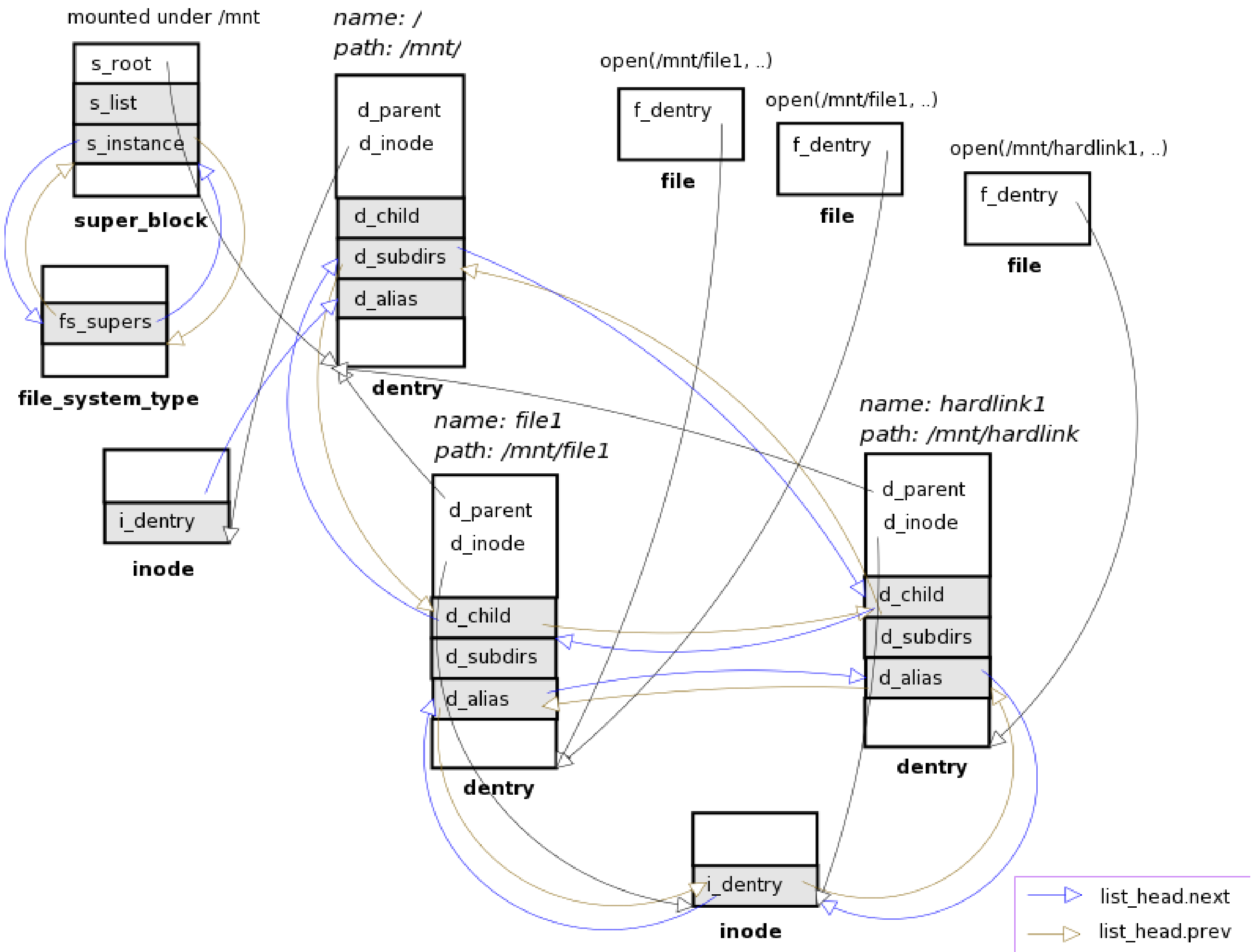

Fig: Relationships between the VFS objects

CIFS runtime resource usage

CIFS creates a kernel thread named “cifsd” at mount time for each unique server (unique target ip address) to manage the corresponding tcp socket and demultiplex responses safely. In addition at module load time (when cifs is first started) cifs creates a thread to manage caching (writeback of “opportunistically locked” oplocked cached file data) called “cifsoplockd” and another one (rarely used) for handling directory change notification responses (“cifsdnotifyd”).

In addition to the memory dynamically allocated for inodes, file structs and dentries, cifs preallocates three pools of buffers to handle requests: cifs small request buffers (cifs_small_rq), cifs large request buffers (cifs_request) and the cifs demultiplex request queue (cifs_mpx_ids). The size of these pools can be configured at module install time but rarely needs to be changed. If many applications are accessing cifs mounts at the same time and therefore more buffers are needed than are in the pool they are dynamically allocated and freed (which can be somewhat slower). The size and usage of the three pools, as well as the two small cifs structures allocated from the slab (cifs_oplock_structs and cifs_inode_cache) can be seen by examining /proc/slabinfo:

```
<name>          <active_objs> <num_objs> <objsize>
cifs_small_rq        31        36          448
cifs_request         6         6           16512
cifs_mpx_ids         19        59          64

cifs_oplock_structs  0         0           32
cifs_inode_cache     1         8           464
```

As can be seen above, the cifs_request buffers are larger than a page for most architectures which can create memory contention inefficiency when many processes (more than five) are simultaneously reading large files during low memory conditions on the client. Large buffers (cifs_request objects) are used most heavily during file read and readdir (listing large directories) but are not typically used during file write (unless SMB signing is negotiated).

The cifs client will attempt to cache file data on the client in the Linux page cache for improved read/write performance if it can be done safely. The Linux page cache automatically balances memory used for file caching with that needed for applications and so the amount of memory used for cifs readahead and writebehind file data can vary widely. Local file caching of cifs files can be disabled by mounting using the “directio” mount option.

CIFS source code

The CIFS kernel source code and header files are located in the fs/cifs directory of the kernel. The files can be categorized as follows::
a) Network protocol definition (mostly SMB/CIFS on-the-wire data formats):
cifspdu.h, ntlmssp.h, rfc1002pdu.h
b) the modules internal data struture defintions, cifsglob.h
c) Worker funtions which handle SMB/CIFS protocol implementation:
cifssmb.c
d) Linux VFS specific mappings to the worker functions: inode.c, file.c, link.c, readdir.c

e) mount (and new session) handling: cifsfs.c, connect.c and sess.c
f) encryption routines: cifsencrypt.c, smbencrypt.c, md5.c, md4.c, smbdes.c
g) misc helper funtions: misc.c, netmisc.c, cifs_debug.c
h) functions which manage sending SMB as tcp data (transport.c)

The cifs mount helper, mount.cifs, is located in Samba version 3 source code in source/client/mount.cifs.c

Appendix B – CIFS versions and Changes

See the file fs/cifs/CHANGES for current list.

Version 1.47

Fix oops in list_del during mount caused by unaligned string.
Fix file corruption caused by writepages page i/o completion bug.

Version 1.46 (Linux kernel 2.6.19)

Support deep tree mounts. Better support OS/2, Win9x (DOS) time stamps.
Allow null user to be specified on mount ("username="). Do not return
EINVAL on readdir when filldir fails due to overwritten blocksize
(fixes FC problem). Return error in rename 2nd attempt retry (ie report
if rename by handle also fails, after rename by path fails, we were
not reporting whether the retry worked or not). Fix NTLMv2 to
work to Windows servers (mount with option "sec=ntlmv2").

Version 1.45 (Linux kernel 2.6.18)

Do not time out lockw calls when using posix extensions. Do not
time out requests if server still responding reasonably fast
on requests on other threads. Improve POSIX locking emulation,
(lock cancel now works, and unlock of merged range works even
to Windows servers now). Fix oops on mount to lanman servers
(win9x, os/2 etc.) when null password. Do not send listxattr
(SMB to query all EAs) if nouser_xattr specified. Fix SE Linux
problem (instantiate inodes/dentries in right order for readdir).

Version 1.44

Rewritten sessionsetup support, including support for legacy SMB
session setup needed for OS/2 and older servers such as Windows 95 and 98.
Fix oops on ls to OS/2 servers. Add support for level 1 FindFirst
so we can do search (ls etc.) to OS/2. Do not send NTCreateX
or recent levels of FindFirst unless server says it supports NT SMBs
(instead use legacy equivalents from LANMAN dialect). Fix to allow

NTLMv2 authentication support (now can use stronger password hashing
on mount if corresponding /proc/fs/cifs/SecurityFlags is set (0x4004).
Allow override of global cifs security flags on mount via "sec=" option(s).

Version 1.43 (Linux kernel 2.6.17)

POSIX locking to servers which support CIFS POSIX Extensions
(disabled by default controlled by proc/fs/cifs/Experimental).
Handle conversion of long share names (especially Asian languages)
to Unicode during mount. Fix memory leak in sess struct on reconnect.
Fix rare oops after acpi suspend. Fix O_TRUNC opens to overwrite on
cifs open which helps rare case when setpathinfo fails or server does
not support it.

Version 1.42

Fix slow oplock break when mounted to different servers at the same time and
the tids match and we try to find matching fid on wrong server. Fix read
looping when signing required by server (2.6.16 kernel only). Fix readdir
vs. rename race which could cause each to hang. Return . and .. even
if server does not. Allow searches to skip first three entries and
begin at any location. Fix oops in find_writeable_file.

Version 1.41

Fix NTLMv2 security (can be enabled in /proc/fs/cifs) so customers can
configure stronger authentication. Fix sfu symlinks so they can
be followed (not just recognized). Fix wraparound of bcc on
read responses when buffer size over 64K and also fix wrap of
max smb buffer size when CIFSMaxBufSize over 64K. Fix oops in
cifs_user_read and cifs_readpages (when EAGAIN on send of smb
on socket is returned over and over). Add POSIX (advisory) byte range
locking support (requires server with newest CIFS UNIX Extensions
to the protocol implemented). Slow down negprot slightly in port 139
RFC1001 case to give session_init time on buggy servers.

Version 1.40 (Linux kernel 2.6.16)

Use fsuid (fsgid) more consistently instead of uid (gid). Improve performance
of readpages by eliminating one extra memcpy. Allow update of file size
from remote server even if file is open for write as long as mount is
directio. Recognize share mode security and send NTLM encrypted password
on tree connect if share mode negotiated.

Version 1.39 (Linux kernel 2.6.15)

Defer close of a file handle slightly if pending writes depend on that handle
(this reduces the EBADF bad file handle errors that can be logged under heavy
stress on writes). Modify cifs Kconfig options to expose CONFIG_CIFS_STATS2
Fix SFU style symlinks and mknod needed for servers which do not support the
CIFS Unix Extensions. Fix setfacl/getfacl on bigendian. Timeout negative
dentries so files that the client sees as deleted but that later get created
on the server will be recognized. Add client side permission check on setattr.
Timeout stuck requests better (where server has never responded or sent corrupt
responses)

Version 1.38

Fix tcp socket retransmission timeouts (e.g. on ENOSPACE from the socket)
to be smaller at first (but increasing) so large write performance performance
over GigE is better. Do not hang thread on illegal byte range lock response
from Windows (Windows can send an RFC1001 size which does not match smb size) by
allowing an SMBs TCP length to be up to a few bytes longer than it should be.
wsize and rsize can now be larger than negotiated buffer size if server
supports large readx/writex, even when directio mount flag not specified.
Write size will in many cases now be 16K instead of 4K which greatly helps
file copy performance on lightly loaded networks. Fix oops in dnotify
when experimental config flag enabled. Make cifsFYI more granular.

Version 1.37

Fix readdir caching when unlink removes file in current search buffer,
and this is followed by a rewind search to just before the deleted entry.
Do not attempt to set ctime unless atime and/or mtime change requested
(most servers throw it away anyway). Fix length check of received smbs
to be more accurate. Fix big endian problem with mapchars mount option,
and with a field returned by statfs.

Version 1.36

Add support for mounting to older pre-CIFS servers such as Windows9x and ME.
For these older servers, add option for passing netbios name of server in
on mount (servernetbiosname). Add suspend support for power management, to
avoid cifsd thread preventing software suspend from working.
Add mount option for disabling the default behavior of sending byte range lock
requests to the server (necessary for certain applications which break with
mandatory lock behavior such as Evolution), and also mount option for
requesting case insensitive matching for path based requests (requesting
case sensitive is the default).

Version 1.35 (Linux kernel 2.6.12)

Add writepage performance improvements. Fix path name conversions
for long filenames on mounts which were done with "mapchars" mount option
specified. Ensure multiplex ids do not collide. Fix case in which
rmmod can oops if done soon after last unmount. Fix truncated
search (readdir) output when resume filename was a long filename.
Fix filename conversion when mapchars mount option was specified and
filename was a long filename.

Version 1.34

Fix error mapping of the TOO_MANY_LINKS (hardlinks) case.
Do not oops if root user kills cifs oplock kernel thread or
kills the cifsd thread (NB: killing the cifs kernel threads is not
recommended, unmount and rmmod cifs will kill them when they are
no longer needed). Fix readdir to ASCII servers (ie older servers
which do not support Unicode) and also require asterisk.
Fix out of memory case in which data could be written one page
off in the page cache.

Version 1.33

Fix caching problem, in which readdir of directory containing a file
which was cached could cause the file's time stamp to be updated
without invalidating the readahead data (so we could get stale
file data on the client for that file even as the server copy changed).
Cleanup response processing so cifsd can not loop when abnormally
terminated.

Appendix C – For additional reading

CIFS, the protocol: http://en.wikipedia.org/wiki/Cifs

Samba server: http://en.wikipedia.org/wiki/Samba_software

Linux Filesystems tutorial: http://us1.samba.org/samba/ftp/cifs-cvs/ols2006-fs-tutorial-smf.pdf

Linux Filesystems: http://www.geocities.com/ravikiran_uvs/articles/rkfs.html

Linux Kernel: fs/cifs/README, fs/cifs/TODO and fs/cifs/CHANGES

man page for mount.cifs