CIFSD

User Manual

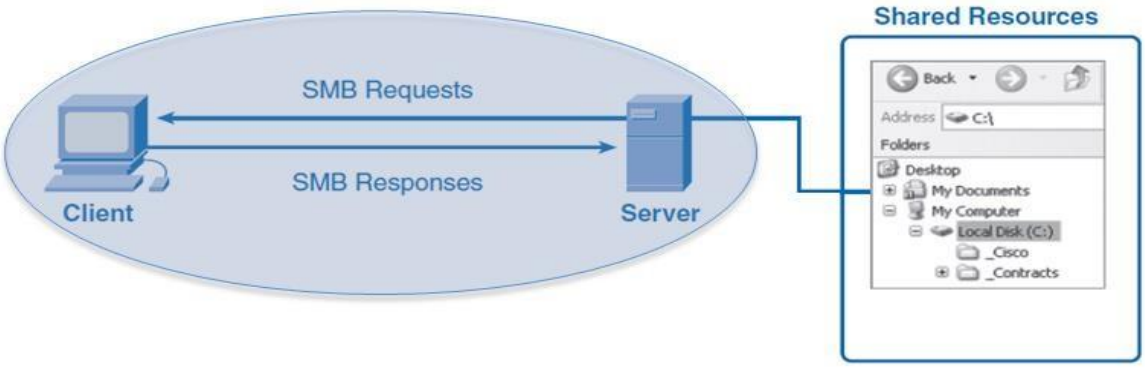
Contents

Table of Contents

Introduction

CIFSD is implementation of SMB/CIFS protocol in kernel space for sharing files over network. SAMBA suite is its user space implementation available freely. Clients (Windows/Linux) can request to accessing the files and folders which is shared by other machines.

The folders or directories which are shared on server side are known as ***"share"***. This term is used at many places throughout whole document.



CIFSD has two main parts for communication

While the File sharing related services are managed by the kernel daemon, the IPC related requests are handled in the user space daemon. It allowed for separating the performance and keeping it in kernel.

When the connection is established in the kernel, the share access part (or say the DCERPC) requests are propagated to the user space daemon via. the netlink sockets.

So, for complete server – Both kernel and user space daemon should be started.

1. Installation of CIFSD on PC

CIFSD is available in two module:-

(1) Kernel and

(2) User

There are 2 methods to obtain the source code, either via. GIT or direct download. Using GIT allows tracking the source changes history. But on certain network there are issues for using GIT, so direct download can be used in those cases.

For Easier code browsing and referencing, the directory structure can be made like this:

```
a.sahrawat@DELL-BUILD10:git$ tree
.
|-- kernel
|   `-- cifsd
|-- user
    `-- cifsd-tools
```

Both the method are provided below.

1.1. Cloning the CIFSD Code repository

Create a Directory for cloning the kernel module and user package repositories:

1.1.1 Cloning Kernel Module source:

```
a.sahrawat@DELL-BUILD10:kernel$ git clone https://github.com/namjaejeon/cifsd.git
Cloning into 'cifsd'...
```

```
remote: Counting objects: 947, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 947 (delta 4), reused 0 (delta 0), pack-reused 934
Receiving objects: 100% (947/947), 685.10 KiB | 81.00 KiB/s, done.
Resolving deltas: 100% (713/713), done.
Checking connectivity... done.
```

1.1.2 Cloning User Package source:

```
a.sahrawat@DELL-BUILD10:user$ git clone https://github.com/namjaejeon/cifsd-tools.git
Cloning into 'cifsd-tools'...
remote: Counting objects: 105, done.
remote: Total 105 (delta 0), reused 0 (delta 0), pack-reused 105
Receiving objects: 100% (105/105), 92.42 KiB | 36.00 KiB/s, done.
Resolving deltas: 100% (44/44), done.
Checking connectivity... done.
```

1.2. Downloading CIFSD Module from github

The above two modules can be downloaded from github by the following links:-
CIFSD Kernel part: https://github.com/namjaejeon/cifsd
CIFSD user space: https://github.com/namjaejeon/cifsd-tools

Download the above two files in appropriate directory and store it in two different folders namely kernel and user.

1.2.1 Unzip files

Unzip these files using the following command:

In kernel folder

```
$ unzip cifsd-master.zip
```

In user folder

```
$ unzip cifsd-tools-master.zip
```

This will create directories namely cifsd-master in kernel folder and cifsd-tools-master in user folder.

1.3. User Package compilation and installation

Go in folder /user/cifsd-tools-master and run the following commands

```
$ ./autogen.sh
```

```
$ ./configure
```

```
$ make
```

```
$ make install
```

Note: - the above commands will be executed in “root” permissions

1.4. Verifying User Package installation

There are 3 executables from the User package:

- cifsd (user space server daemon for communication with kernel)
- cifsadmin (utility to allow for addition/deletion of users/shares)
- cifsstat (utility to get the server stats per client connection)

We can check that whether the files are installed properly and where they are located by the following commands:

```
$ which cifsd
$ /urs/local/sbin/cifsd
```

```
$ which cifsadmin
$ /usr/local/sbin/cifsadmin
```

```
$ which cifsstat
$ /usr/local/sbin/cifsstat
```

1.5. CIFSD Kernel Source Compilation

There are two methods for compilation of the CIFSD kernel package. It can be compiled as independent kernel module or it can be made part of the static kernel image.
Kernel module allows more control from development point of view as the module can be inserted/removed (inmod/rmmod) several times.
While, when compiling as static – each change will need the system to be restarted. Both methods are explained below, depending upon the scenario any method can be selected.

1.5.1 Compiling as Kernel Module

The default **Makefile** is for building the cifsd in a Kernel build system.
However, it can also be compiled as an independent kernel module (*ko - kernel object)*.
To build it as a kernel module, replace the existing Makefile with the snippet as given below:
Note: KDIR should point to the appropriate kernel header files for which the user intends to build the kernel module.

In this example, the KDIR is pointing to the linux kernel which is installed on the system. *"uname"* command gives the version of the current kernel. So, from the default system path – it will use the kernel headers of the installed kernel.
For a system with multiple kernels : /lib/modules has a layout like:

```
[root@localhost cifsd-tools-master]# ls -l /lib/modules/
total 32
drwxr-xr-x. 6 root root 4096 May 25 05:54 3.11.10-301.fc20.i686+PAE
drwxr-xr-x. 3 root root 4096 May 26 01:23 4.1.10-linux-port+
drwxr-xr-x. 3 root root 4096 May 30 06:29 4.4.11
drwxr-xr-x. 3 root root 4096 Aug  9 04:50 4.6.1
```

And current kernel is :

```
[root@localhost cifsd-tools-master]# uname -r
4.6.1
```

So, the proper path for the compilation and make it working for the current kernel will be:

```
[root@localhost cifsd-tools-master]# ls /lib/modules/4.6.1/build/
arch   certs    CREDITS  Documentation  firmware  include  ipc     Kconfig  lib
Makefile  modules.order   net     REPORTING-BUGS  scripts   sound  usr
block  COPYING  crypto   drivers        fs        init     Kbuild  kernel   MAINTAINERS  mm
Module.symvers README samples           security  tools  virt
[root@localhost cifsd-tools-master]#
```

MAKEFILE to be used:

```
KDIR=/lib/modules/$(shell uname -r)/build
ccflags-y        += -DCONFIG_CIFS_SMB2_SERVER=y
ccflags-y        += -DCONFIG_CIFS_SERVER=m
obj-m            += cifsd.o

cifsd-y :=       export.o connect.o srv.o unicode.o encrypt.o auth.o \
                 fh.o vfs.o misc.o smb1pdu.o smb1ops.o dcerpc.o \
                 oplock.o winreg.o netmisc.o netlink.o

cifsd-y += smb2pdu.o smb2ops.o

all:
        make -C ${KDIR} M=$(PWD) modules

clean:
        make -C ${KDIR} M=$(PWD) clean
```

After replacing this Makefile run the following command:

```
$ make
```

The above command generates a ***cifsd.ko*** file

After the creation of "**.ko"** file run the following commands on command line

```
$ insmod /kernel/cifsd-master/cifsd.ko
```

This will start the kernel part of the CIFSD Server. It can be verified by checking the modules inserted in kernel via. 'lsmod' command

```
[root@localhost]# lsmod|grep cifsd
cifsd                 237568  0
```

1.5.2 Compiling statically with kernel image

Copy the complete kernel source directory under the Kernel source tree.
"CIFS" is part of the Kernel/filesystem directory. So, copy the "cifsd" code also under FS

```
|--linux-4.7
      `--fs
           `--cifs
           `--cifsd
```

There is Makefile and Kconfig under the "cifsd" code repository. In order to compile with kernel, need to make entries so that these can be picked up in configuration and compilation.

In the "filesystem" fs tree (e.g., linux-4.7/fs) – there are 2 files:
Makefile, Kconfig – which selects all independent filesystem modules.

a) Make entry for cifsd/Kconfig in fs/Kconfig (below CIFS for easier reference and categorization under network filesystems.

```
vi fs/Kconfig
```

```
source "fs/cifs/Kconfig"
source "fs/cifsd/Kconfig"
```

b) Similarly, make entry in fs/Makefile

```
vi fs/Makefile
```

```
obj-$(CONFIG_CIFS)                  += cifs/
obj-$(CONFIG_CIFS_SERVER)           += cifsd/
```

After making these entries, “CIFSD” can be selected via. the kernel configuration for compilation with tree.
Select like below:

```
make menuconfig
      File systems -->
            [*] Network File systems -->
```

```
    <M>   CIFS support (advanced network filesystem, SMBFS successor)
    [*]     CIFS statistics
    [ ]       Extended statistics
    [*]     Support legacy servers which use weaker LANMAN security
    [*]     Kerberos/SPNEGO advanced session setup
    [*]     CIFS extended attributes
    [*]       CIFS POSIX Extensions
    [*]       Provide CIFS ACL support
    [*]     Enable CIFS debugging routines
    [ ]       Enable additional CIFS debugging routines
    [*]     DFS feature support
    [*]     SMB2 and SMB3 network file system support
    [*]       SMB3.1.1 network file system support (Experimental)
    [*]   Provide CIFS client caching support
    <*>   CIFS server support
    [ ]     SMB2 server support (NEW)
```

Proceed for normal kernel compilation, install kernel image and reboot the system.

1.6. Copying smb.conf.example to new location

```
$ cp cifsd-tools-master/smb.conf.example /etc/cifs/smb.conf
```

2. Execution of CIFSD

2.1. Adding username and password

A new user in cifsd can be added by the following commands :-

```
$ cifsadmin –a <username>
```

The above feeds a username and then the we can provide the password for the new user as follows:

```
sh-3.2# cifsadmin -a usr1
New Password:
Retype Password:
sh-3.2#
sh-3.2# cat /sys/fs/cifssrv/user
usr2
usr1
sh-3.2#
```

The above two commands add a user with username and password

2.2. Running cifsd in background

Now we can run cifsd user space server daemon in background

```
$ cifsd &
```

```
sh-3.2# ./cifsd &
[1] 211
```

2.3. Checking registered users list

We can check the registered users by the following command

```
$ cat /sys/fs/cifsd/user
```

This will give the names for all the registered users:

```
[root@localhost cifsd-tools-master]# cat /sys/fs/cifsd/user
guest
user
usr2
usr1
```

3. Accessing CIFSD from Windows

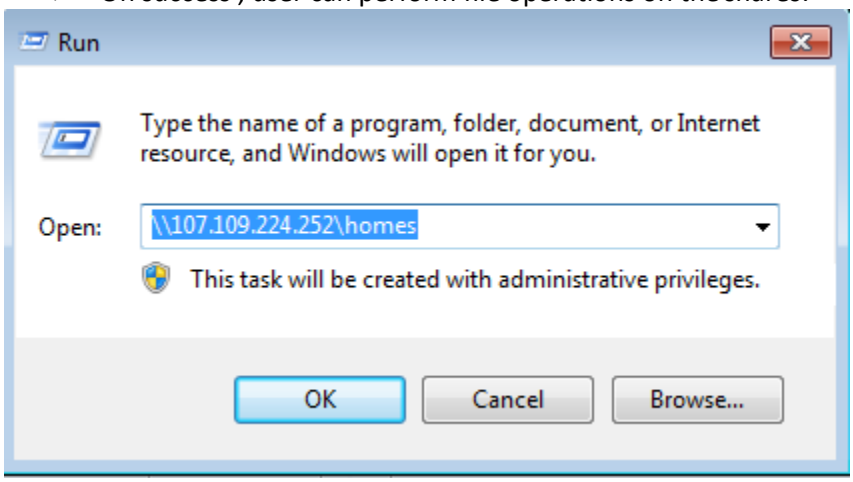
3.1. Prerequisite:

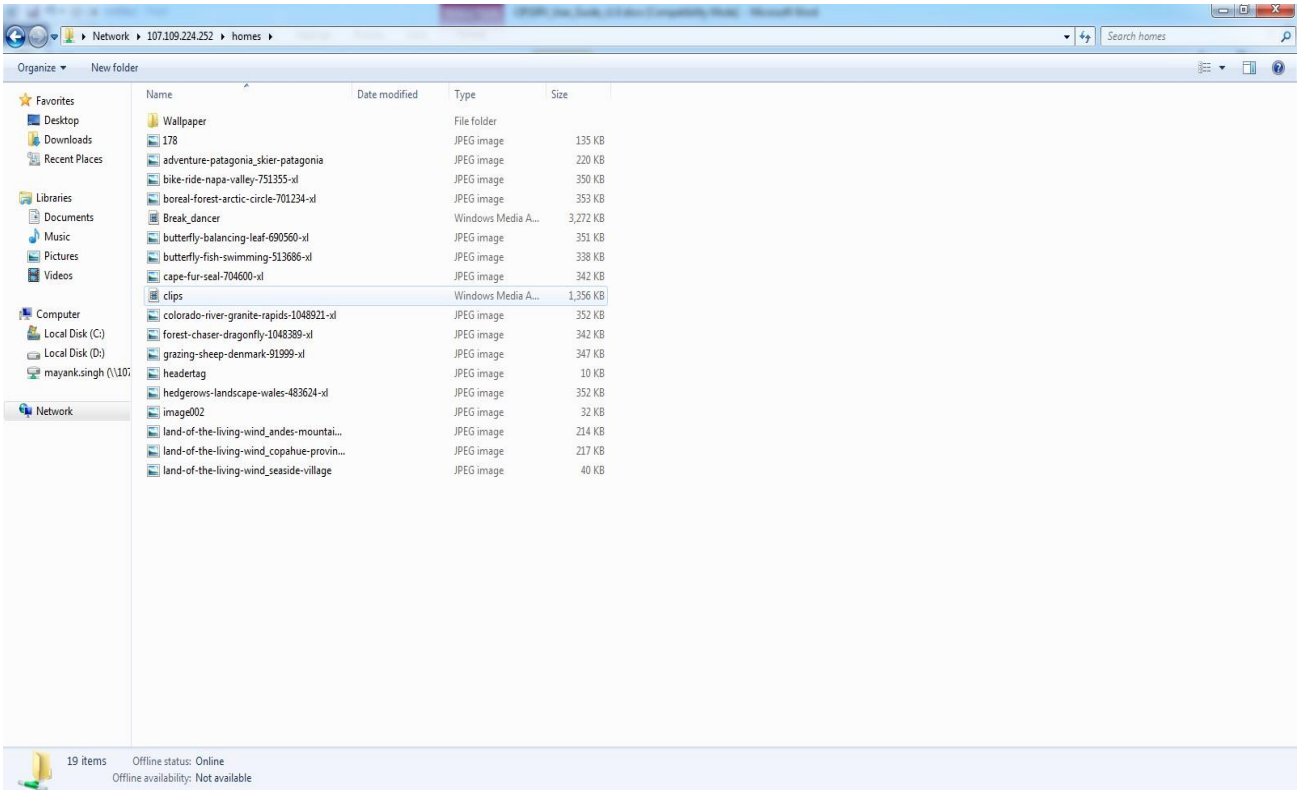
1. Proper network connection should exist between connection client and the Server (client can be linux/Windows machine/…) and CIFSD running as server on linux PC.
2. Lan Manager should be in proper working condition.
   - "ping" command can be used for verification of network connection from Windows machine/linux to check whether it is able to ping correctly the CIFSD Server.

```
C:\Users\vaibhav.k94>ping 107.109.224.252

Pinging 107.109.224.252 with 32 bytes of data:
Reply from 107.109.224.252: bytes=32 time<1ms TTL=64
Reply from 107.109.224.252: bytes=32 time<1ms TTL=64
Reply from 107.109.224.252: bytes=32 time<1ms TTL=64

Ping statistics for 107.109.224.252:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\vaibhav.k94>
```

3.2. Connecting to CIFSD Server from Windows:

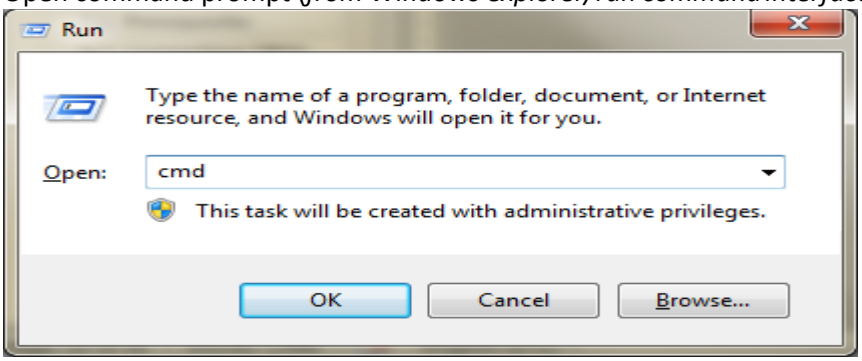
- In the Windows Explorer Bar (or Run command interface)
- Enter the CIFSD Server IP address. (IP address = **107.109.224.252)**
- Enter username and password on prompting.
- On success , user can perform file operations on the shares:





3.3. Changing SMB2 to SMB1 protocol version on Windows PC:

- Open command prompt (*from Windows explorer/run command interface*).



- Enter below set of commands.

```
sc.exe config lanmanworkstation depend= bowser/mrxsmb10/nsi   sc.exe
config mrxsmb20 start= disabled
sc.exe config mrxsmb10 start= auto
```

- Reboot the system
- Now client machine will use SMB1 protocol for making requests to CIFSD.

4. Accessing CIFSD from Linux Machine

4.1. Prerequisite:

- CIFS file system should be available on Linux client side.
  Verify this by checking the registered filesystem list like:

```
[root@localhost cifsd-tools-master]# cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   ramfs
nodev   bdev
nodev   proc
nodev   cpuset
nodev   cgroup
nodev   cgroup2
nodev   tmpfs
nodev   devtmpfs
nodev   debugfs
nodev   tracefs
nodev   securityfs
nodev   sockfs
nodev   pipefs
nodev   hugetlbfs
nodev   devpts
        ext3
        ext2
        ext4
        vfat
nodev   ecryptfs
nodev   autofs
        fuseblk
nodev   fuse
nodev   fusectl
nodev   pstore
nodev   mqueue
nodev   rpc_pipefs
nodev   nfsd
        xfs
nodev   cifs
```

  If it not available by default, try doing a “modprobe” command for “CIFS” If the CIFS filesystem modules are available. It will register it to be used. Verify again after doing modprobe if the “CIFS” is available or not.

```
[root@localhost cifsd-tools-master]# modprobe cifs
[root@localhost cifsd-tools-master]# lsmod |grep cifs
cifs                  557056  0
fscache                61440  1 cifs
```

- It should be connected to network properly and must have reachability to CIFSD target.

```
[root@localhost mayanksingh]# ping 107.109.224.252
PING 107.109.224.252 (107.109.224.252) 56(84) bytes of data.
64 bytes from 107.109.224.252: icmp_seq=1 ttl=64 time=0.302 ms
64 bytes from 107.109.224.252: icmp_seq=2 ttl=64 time=0.297 ms
64 bytes from 107.109.224.252: icmp_seq=3 ttl=64 time=0.301 ms
^C
--- 107.109.224.252 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.297/0.300/0.302/0.002 ms
```

4.2. Connecting with CIFSD Server:

We require root permissions to access the CIFSD running machine
We will mount the drive in /mnt1 folder by using the following command

```
$ mount -t cifs //107.109.224.252/homes/ /mnt -o
user=usr1,pass=123,sec=ntlm,vers=2.1,actimeo=0
```

```
root@VDFS:rachit$ mount -t cifs //107.109.224.252/homes/ /mnt1 -o user=usr1,pass=123,vers=2.1
root@VDFS:rachit$ cd /mnt1/
root@VDFS:mnt1$ ls
178.jpg                                butterfly-fish-swimming-513686-xl.jpg        headertag.jpg
adventure-patagonia_skier-patagonia.jpg  cape-fur-seal-704600-xl.jpg                hedgerows-landscape-wales-483624-xl.jpg
bike-ride-napa-valley-751355-xl.jpg      clips.wmv                                  image002.jpg
boreal-forest-arctic-circle-701234-xl.jpg  colorado-river-granite-rapids-1048921-xl.jpg  land-of-the-living-wind_andes-mountains-waterfall.jpg
Break_dancer.wmv                         forest-chaser-dragonfly-1048389-xl.jpg     land-of-the-living-wind_copahue-provincial-park.jpg
butterfly-balancing-leaf-690560-xl.jpg   grazing-sheep-denmark-91999-xl.jpg         land-of-the-living-wind_seaside-village.jpg
root@VDFS:mnt1$
```

Here,

- **homes** named share exist on network location 107.109.224.145
- Client is trying to mount that folder locally at /mnt directory by providing by providing correct authentication (username/password).
- **ntlm** represents security model in CIFS protocol (SMB1)

On success it does not return anything and in case of any failure it gives error logs. After mount success "/mnt" will point to the contents of the shared directory. And all file operations can be performed on this mount point.

4.3. Know-Hows about some known issues in CIFSD setup

During CIFSD setup, certain issues can occur. There can be issues when connecting from Client side. These can be easily overcome if we have some prior information about them.
Below a list of situations/behaviors are given and their possible explanation/remedy is given.

```
root@sri-123:~# mount -t cifs //107.109.224.252/homes /mnt -o
user=usr1,pass=123,sec=ntlm
```

```
mount: block device //107.109.224.252/homes is write-protected, mounting read-only
mount: cannot mount block device //107.109.224.252/homes read-only
```

- Here user **vaibhav** with password **vaibhavk** is trying to particular network share named **homes.**
  But client side is receiving ***red colored error message***.

Possible Reason/s:

- User does not exist in CIFSD registered user list
- CIFSD user list itself not configured while CIFSD share exists
- Given username or password may be wrong.

Possible Remedy:

- Check username/ password.
- Check whether CISD has been properly configured.

```
root@sri-123:~# mount -t cifs //107.109.224.252/homes /mnt -o
user=usr1,pass=123,sec=ntlm
```

```
mount: Operation now in progress
      OR
mount: No route to host
```

- If you receive any of two above error message there can be two possible reasons

Possible Reason/s:

- Given IP address machine might not exist on Network
- CIFSD might not have been configured

Possible Remedy:

- Give correct machine IP while making request to CIFSD.
- Configure CIFSD correctly.

```
root@sri-123:~# mount -t cifs //107.109.223.19/homes /mnt -o
user=vaibhav,pass=vaibhavk,sec=ntlm
```

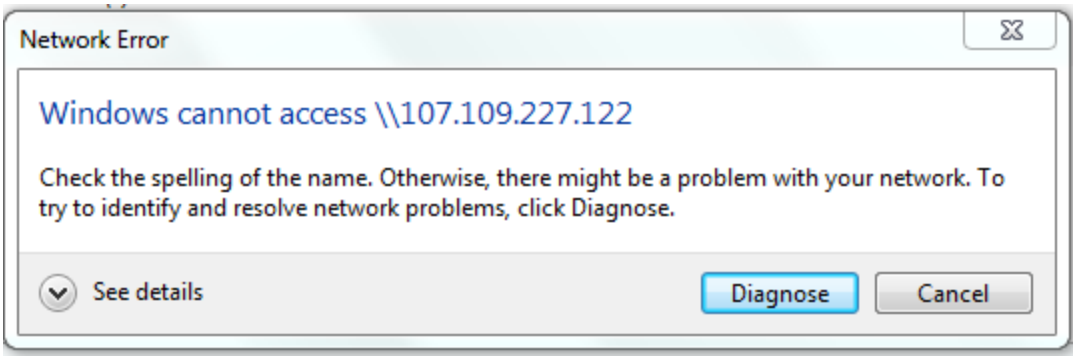
```
mount: //107.109.223.19/homes is not a valid block device
```

Possible Reason/s :

- When user is configured but share does not exist.

Possible Remedy:

- Configure the share properly.



Possible Reasons:

- Lan Manager may not be working properly.
- Destination host do not exist on network
- CIFSD may not be configured properly on server side.

5. CIFSD related tools

5.1. CIFSD

This tools aims at managing the share part effectively. It controls what to keep under share section and also controls various attributes of share directory and its sub directory. This utility mainly deal with *smb.conf* CIFSD configuration file for storing information about shares and their attributes.

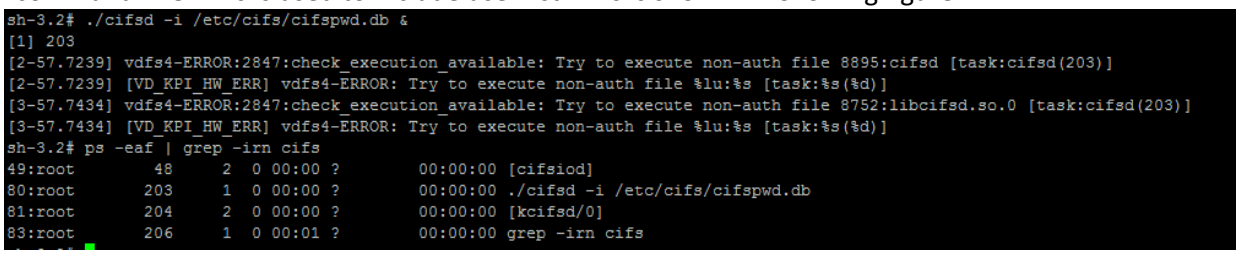
5.1.1. CIFSD utility command line options

```
-sh-3.2# ./cifsd Usage:
cifsd [option] option:
      -h help
      -v version
      -d debug
      -c smb.conf
      -i usrs-db
```

We can run *cifsd* with *smb.conf* file located at another place by using –c option as follows

```
sh-3.2# ./cifsd -c /etc/cifs/smb.conf &
[1] 204
[2-46.5401] vdfs4-ERROR:2847:check_execution_available: Try to execute non-auth file 8895:cifsd [task:cifsd(204)]
[2-46.5401] [VD_KPI_HW_ERR] vdfs4-ERROR: Try to execute non-auth file %lu:%s [task:%s(%d)]
[2-46.5440] vdfs4-ERROR:2847:check_execution_available: Try to execute non-auth file 8752:libcifsd.so.0 [task:cifsd(204)]
[2-46.5440] [VD_KPI_HW_ERR] vdfs4-ERROR: Try to execute non-auth file %lu:%s [task:%s(%d)]
sh-3.2# ps -eaf | grep -irn cifs
49:root        48     2  0 00:00 ?        00:00:00 [cifsiod]
79:root       204     1  0 00:00 ?        00:00:00 ./cifsd -c /etc/cifs/smb.conf
80:root       205     2  0 00:00 ?        00:00:00 [kcifsd/0]
82:root       207     1  0 00:00 ?        00:00:00 grep -irn cifs
sh-3.2#
```

We can also run *cifsd* with *cifspwd.db* file located at another location by –i option located at command line. This is used to include user list. This is shown in following figure:-



5.2. CIFSADMIN

This tool helps in managing the users who wants to access shared folders/directories over network. It maintains their username and password in specific file named as *usrpwd.db.*

```
-sh-3.2# ./cifsadmin
Usage: cifsadmin [options]
options:
        -h help
        -v verbose
        -a <username> add single user to usrpwd.db
        -d <username> delete user account
        -q <username> query single user configured status in cifsd
```

User must be registered with CIFSD before making request for accessing any service provided by CIFSd.

5.2.1. Adding a user:

- Run

```
$ ./cifsadmin –a <user_name>
$ ./cifsadmin –c
```

- See newly configured user in **/etc/cifs/cifspwd.db** file.

5.2.2. Querying a user:

- Checking added user by below command.

```
$ ./cifsadmin –q  <user_name>
```

5.2.3. Deleting a user:

- Run

```
$  ./cifsadmin  -r <user_name>
```

- to update the user configuration after deletion.

```
$ ./cifsadmin –c
```

6. Parameters Description:

Important: At present only the below attributes can be set/unset for any share. You can set /unset below attributes for any share.

- **allow hosts: -** This parameter tells about the list of machines or IP's which are allowed to access the service given by CIFSD.
- **available** :- This parameter lets you "turn off" a service. If *available = no*, then *ALL* attempts to connect to the service will fail. Such failures are logged.
  Default: *available = yes*
- **browsable** :- This controls whether this share is seen in the list of available shares in a net view and in the browse list.
  Default: *browseable = yes*
- **comment** :- This comment string is get associated with new share . It's a way to giving little elaborate explanation about share to make it more understandable.
- **Deny host** :- This parameter tells about the list of machines or IP's which are not allowed to access the service given by CIFSD.
- **guest ok** :- If this parameter is set yes then it means NO password is required to access the service provided by CIFSD.
- **guest only** :- If this parameter is set yes then only guest connections to users are allowed.
- **invalid users** :- This list out the set of users which would be treated as invalid ones while trying to access CIFSD service.
- **max connections** :- This option allows the number of simultaneous connections to a service to be limited. If *max connections* is greater than 0 then connections will be refused if this number of connections to the service are already open. A value of zero mean an unlimited number of connections may be made.
- **oplocks** :- This parameter is actually boolean. yes value tells smbd whether to issue oplocks (opportunistic locks) to file open requests on this share
- **path** :- This contains full path of share on CIFSD server side.
- **Readlist** :- this list contains the names of user which are given read only access to a service
- **Valid user** :- this list contains the name if the user who should be allowed to login to the service.
- **Writeable** :- it is inverted synonym of read only. The user can edit and write into the file.

- **Invalid user**: -this contains the names of user which should not be allowed to login to the service. This absolutely ensures that some improper setting does not breach the security

- **read only** :- An inverted synonym is writeable.
  If this parameter is yes, then users of a service may not create or modify files in the service's directory.

- **write ok** :- This parameter is a synonym for read only.

- **Write list** :- This is a list of users that are given read-write access to a service. If the connecting user is in this list then they will be given write access, no matter what the read only option is set to. The list can include group names using the @group syntax.