Multisession Mounts with the Linux CIFS Client

Jeff Layton

jlayton@samba.org
(Red Hat/Samba Team)

Who is this guy?

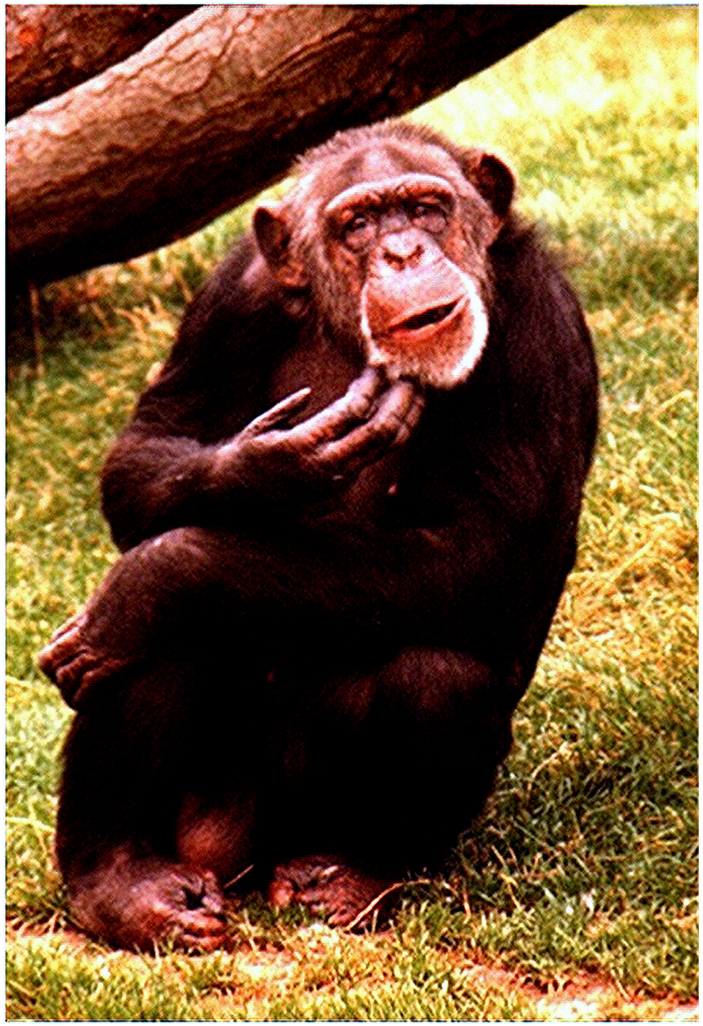
- Years of work as Unix sysadmin
- Member of file system engineering team at Red Hat since 2006
- Joined worldwide Samba team in 2008
- Primarily work on NFS and CIFS, but also dabble in generic VFS layer (and other places)
- Maintain the cifs-utils package

File creation confusion

- First test:
  - Mount cifs share with one user's credentials and with unix extensions enabled
  - Share is world-writable
  - “touch” file in share as another user

```
$ touch testfile1
touch: cannot touch `testfile1': Permission denied
```

What happened?



- File was created on server using mount credentials
- CIFS attempts to enforce permissions on client
- That can't fix ownership
- File is created but later ops fail!

Permissions Enforcement

- Second test:
  - Mount share with one user's credentials and without unix permissions
  - As another user, access a file that should be accessible by only that user.
- You can't enforce permissions correctly if you don't know what they should be
- Even if you do, checking on the client is racy – they can change after you check them but before enforcement action

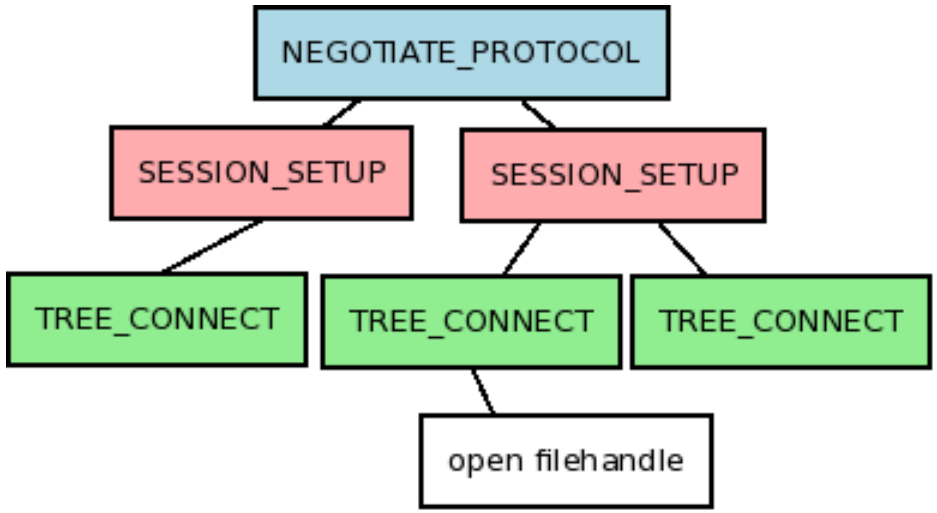
Why is it this way?

- CIFS protocol is **session-based**
- Credentials are handled per-session
- Linux CIFS only has single session per mount
- **Shared Credentials!**

The solution...

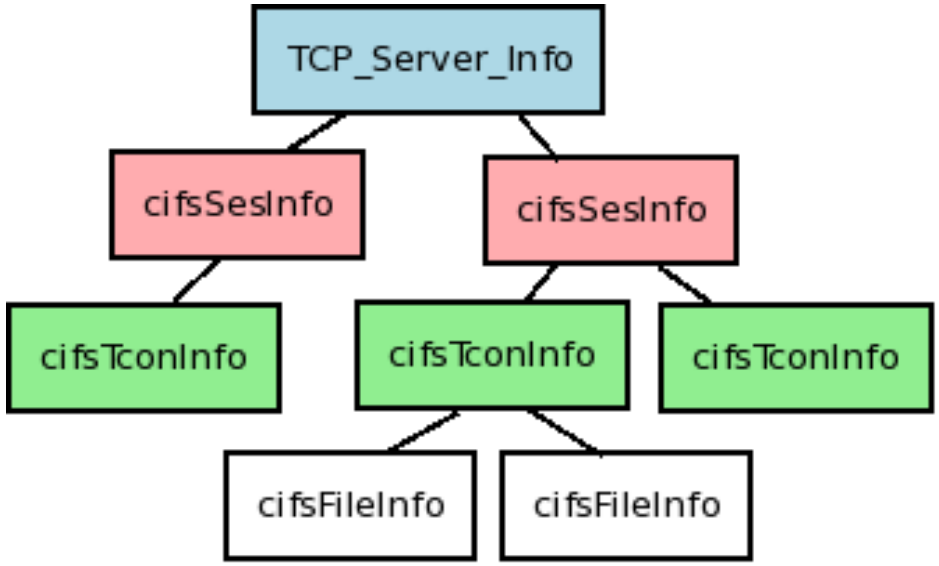
- Each user should use their own credentials
- Have multiple sessions per mount
- Establish sessions on an as-needed basis
- Let the server handle permissions
- **Goal:** Easy as NFS

Protocol hierarchy



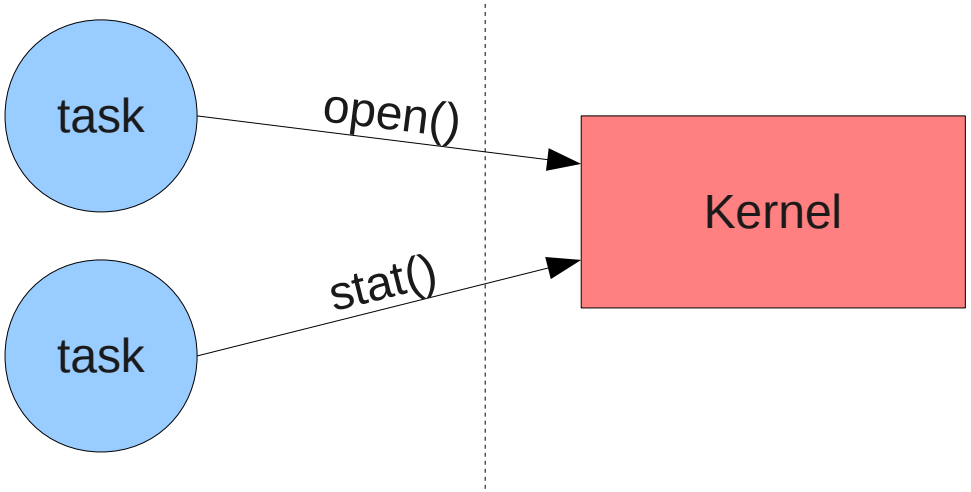
- The CIFS protocol has a hierarchy of sorts
- NEGOTIATE
- SESSION_SETUP
- TREE_CONNECT
- Open filehandles
- Other path-based ops

Basic Object Hierarchy



- TCP_Server_Info (socket)
  - per socket
  - NEGOTIATE
- cifsSesInfo (credential)
  - SESSION_SETUP
- cifsTconInfo (share)
  - TREE_CONNECT
- cifsFileInfo (file)
  - open filehandles

Userspace to Kernel



- To do privileged ops, userland processes have to ask kernel to do it for them
- Typically this is done via **system calls** (**syscalls**)

Linux VFS Anatomy

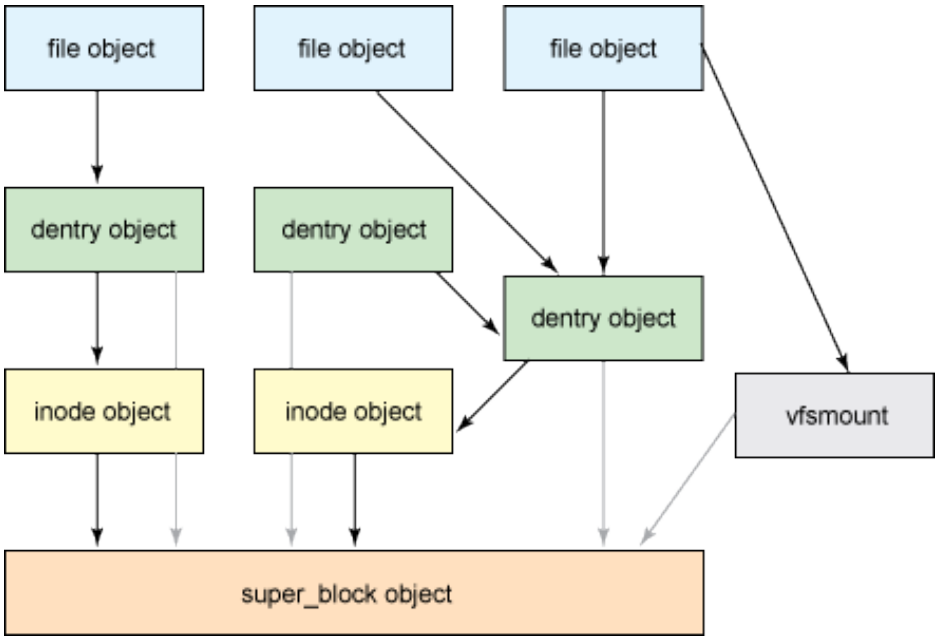


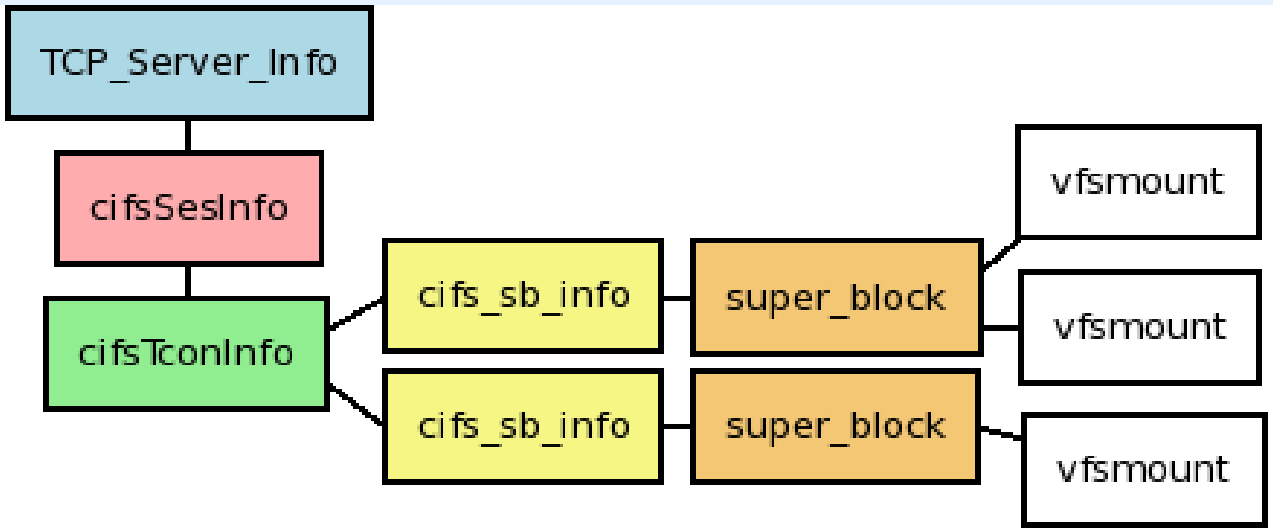
Image copyright M Tim Jones and IBM

- **inode** is an actual file or directory
- **dentry** is a path component
- **super_block** is connection to backing store
- **vfsmount** is connection to mount tree
- **file** is an open file descriptor

Task Credentials

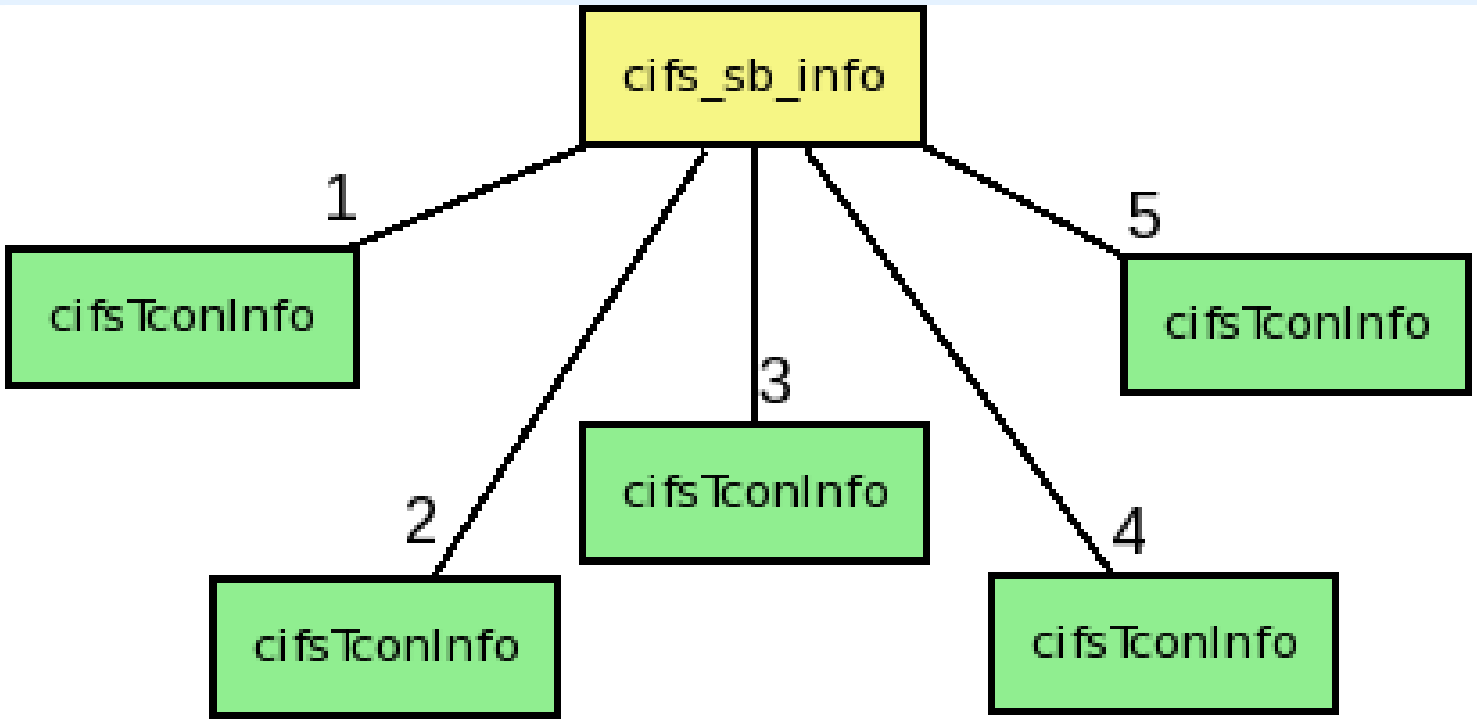
- “task” in kernel lingo is thread of execution (aka a process or thread)
- Each task has several uid's associated with it
  - **real:** who owns the process?
  - **effective:** determines permissions when accessing shared resources (shmem, etc.)
  - **saved:** allows task to switch effective uids
  - **filesystem:** permissions for accessing files
- **Goal is to match up task creds to sessions**

CIFS-VFS Connection



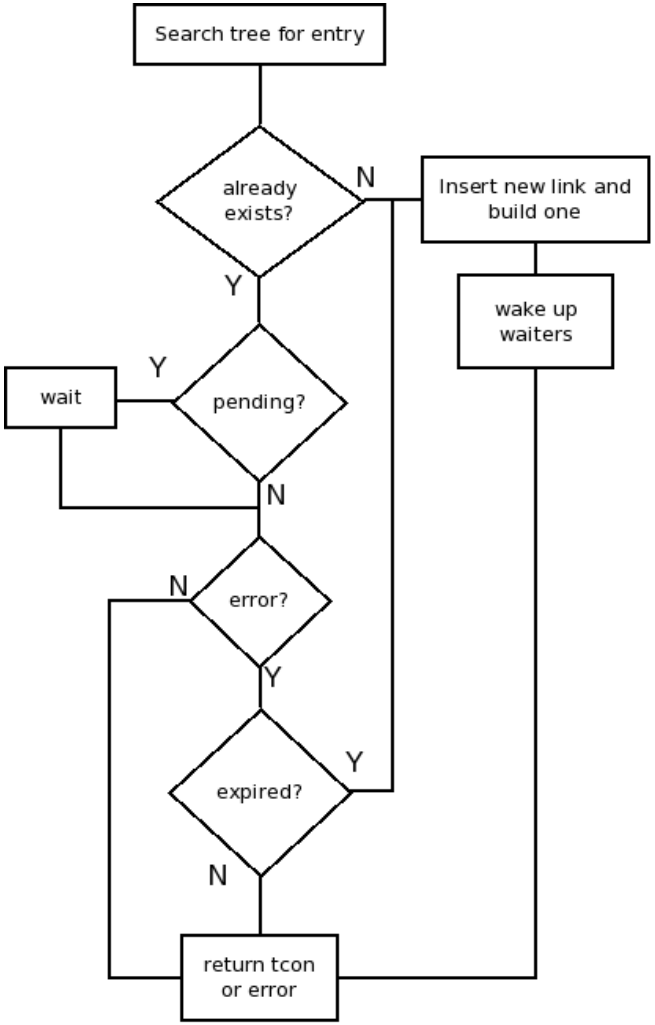
- a mount in the VFS points to a super_block
- super_block points to a cifs_sb_info
- cifs_sb_info has single pointer to cifsTconInfo
- so...each super_block refers to one set of creds

More TCon's, Please...



- cifs_sb_info should point to more than one tcon
- convert tcon pointer into a radix tree (tcon_tree)
- use the fsuid as the key

Finding/Bulding TCons



- Task needs to do a SMB call, so it requests a tcon
- Search for one that matches fsuid
- If not found, then try to build one
- If that fails, error
- map to default?

Building New Tcon's

- TCon established at mount is **master**
- To build new one, use **master** as template
- Build/find new tcon/session for current fsuid
- **Cannot prompt for passwords!**
  - For now, effectively limits this code to krb5 auth
- File creation and permissions enforcement work as expected!

Display of ownership

- How to handle presentation of file ownership on client? (primarily from stat() calls)
- With Unix extensions enabled, assume that client and server have uids/gids mapped the same way (similar to NFSv2/3)
- Without Unix extensions, server doesn't send any ownership info. Current code usually sets owner to value of uid= option or root.
- For multisession mounts, always present current fsuid as owner of inodes.

What about readdir?

- Windows has a feature that's used to present different directory contents to different users (Access Based Enumeration -- ABE)
- Is this potentially at risk of exposing that info to users that shouldn't have it?
- Linux CIFS does not cache readdir info. Any readdir() syscall will cause FIND_* to be sent.
- Potentially we could cache this info, but need to understand better when to invalidate cache.

Future Enhancements

- Need a way to prune off old, unused TCons.
  - Current patchset keeps them around forever.
- Allow for other authtypes besides krb5.
  - Need a way to stash host/username/password
  - http://wiki.samba.org/index.php/LinuxCIFS_CredentialStashing
- Map to “default” session with no creds
- ID mapping
  - don't require unity-mapped uids/gids with Unix ext.
  - allow real ownership info from non-samba servers?

Questions, Comments, Concerns?

My Questions...

- What krb5 principal should we use for root?
  - NFS uses **nfs/host.example.com** and that generally is mapped to “nobody”
  - Is there something equivalent in CIFS?
- What about MaxVCs?
  - Servers send this value but rumor has it that it's not to be trusted.
  - Are there practical limits to number of sessions on a socket?