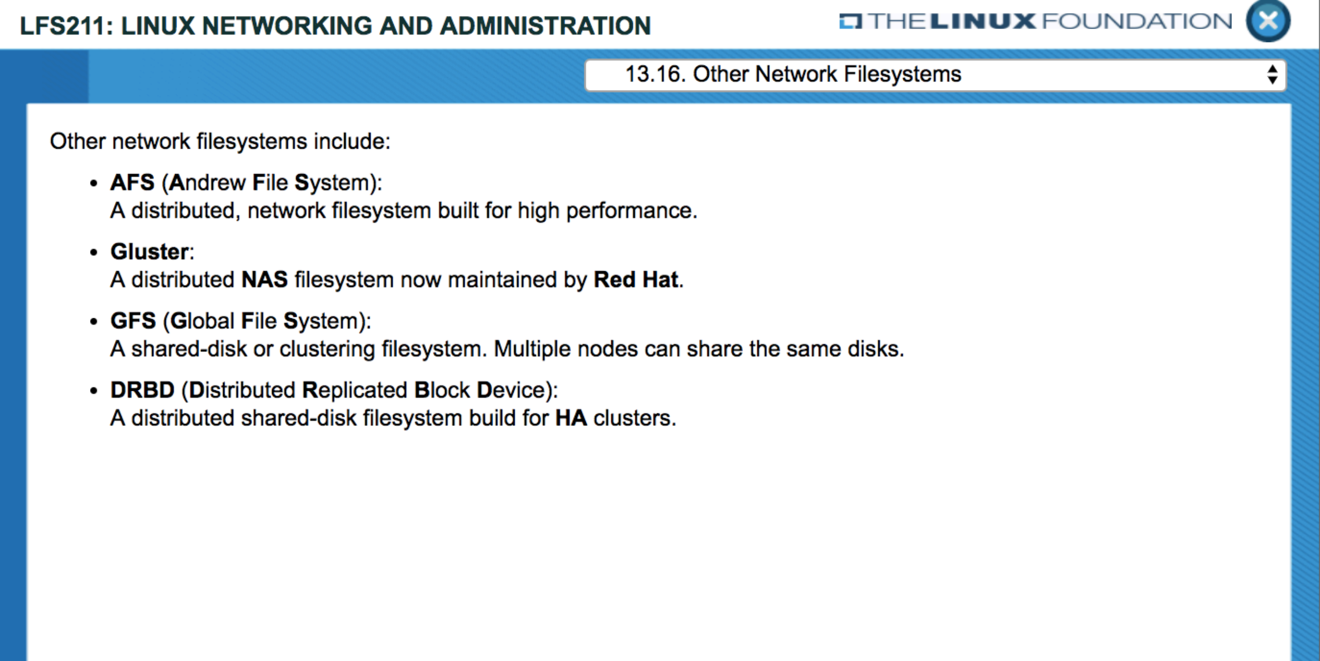
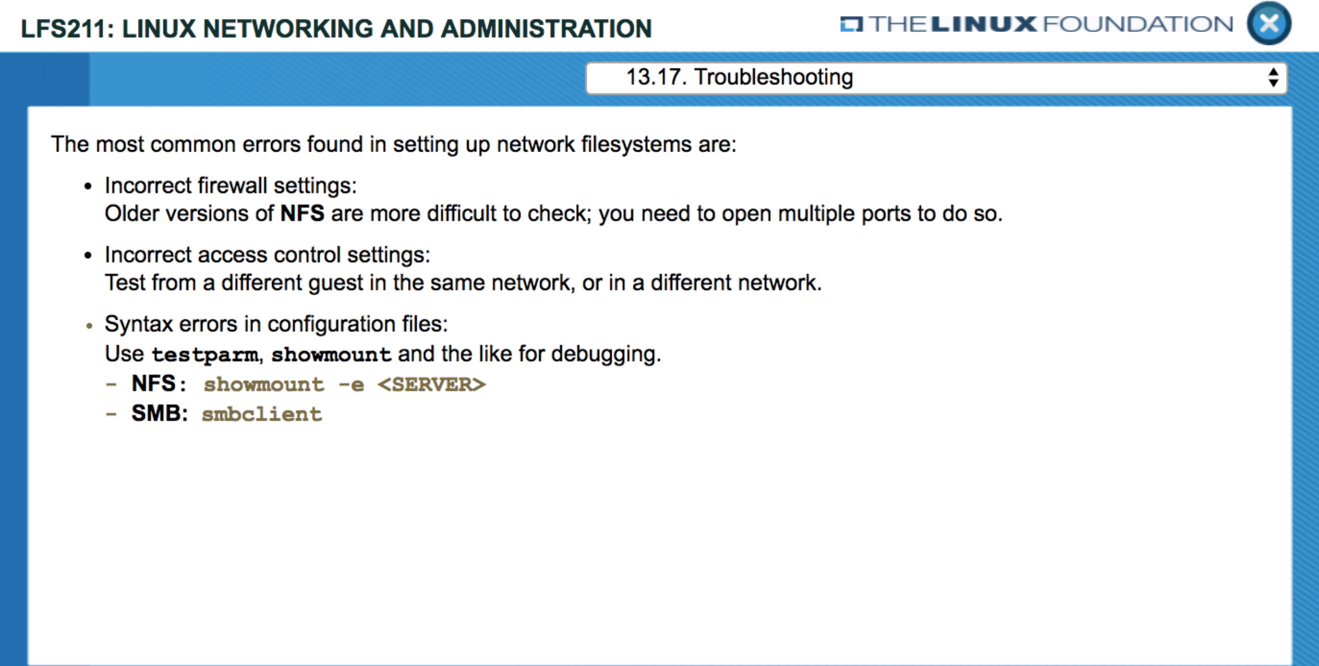


13.16. Other Network Filesystems

Other network filesystems include:

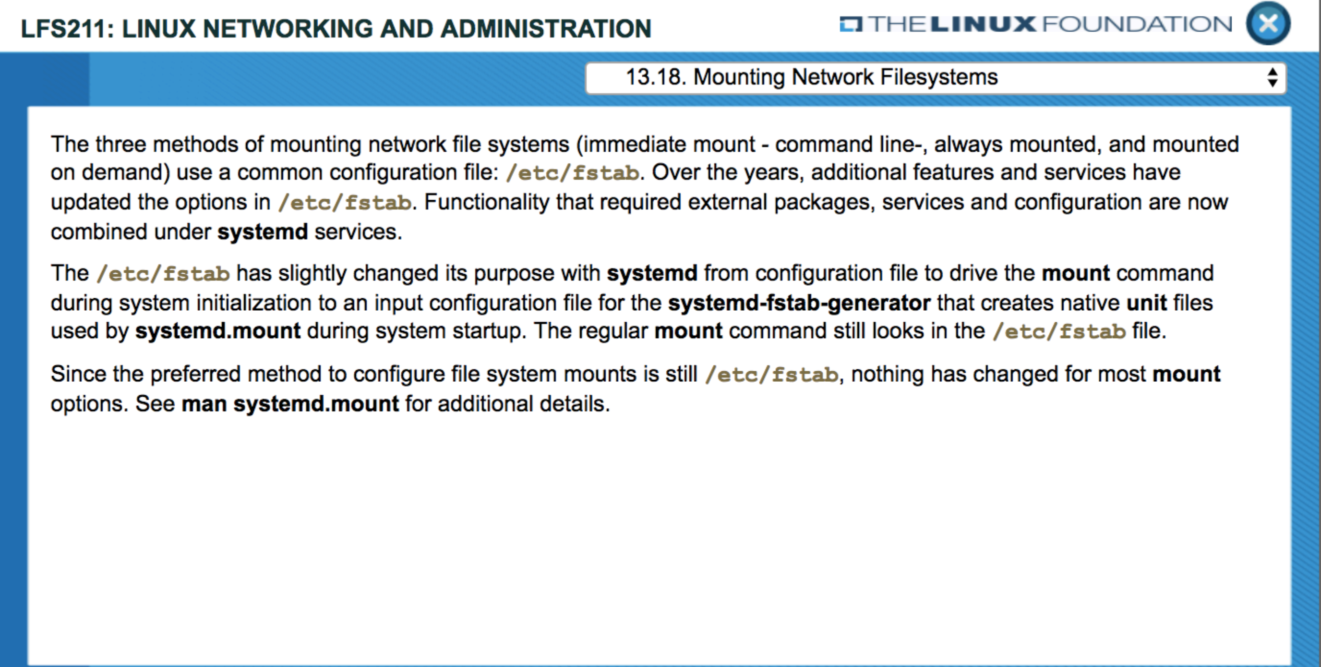
- **AFS** (**A**ndrew **F**ile **S**ystem):
  A distributed, network filesystem built for high performance.
- **Gluster**:
  A distributed **NAS** filesystem now maintained by **Red Hat**.
- **GFS** (**G**lobal **F**ile **S**ystem):
  A shared-disk or clustering filesystem. Multiple nodes can share the same disks.
- **DRBD** (**D**istributed **R**eplicated **B**lock **D**evice):
  A distributed shared-disk filesystem build for **HA** clusters.



13.17. Troubleshooting

The most common errors found in setting up network filesystems are:

- Incorrect firewall settings:
  Older versions of **NFS** are more difficult to check; you need to open multiple ports to do so.
- Incorrect access control settings:
  Test from a different guest in the same network, or in a different network.
- Syntax errors in configuration files:
  Use **`testparm`**, **`showmount`** and the like for debugging.
  - **NFS**: `showmount -e <SERVER>`
  - **SMB**: `smbclient`

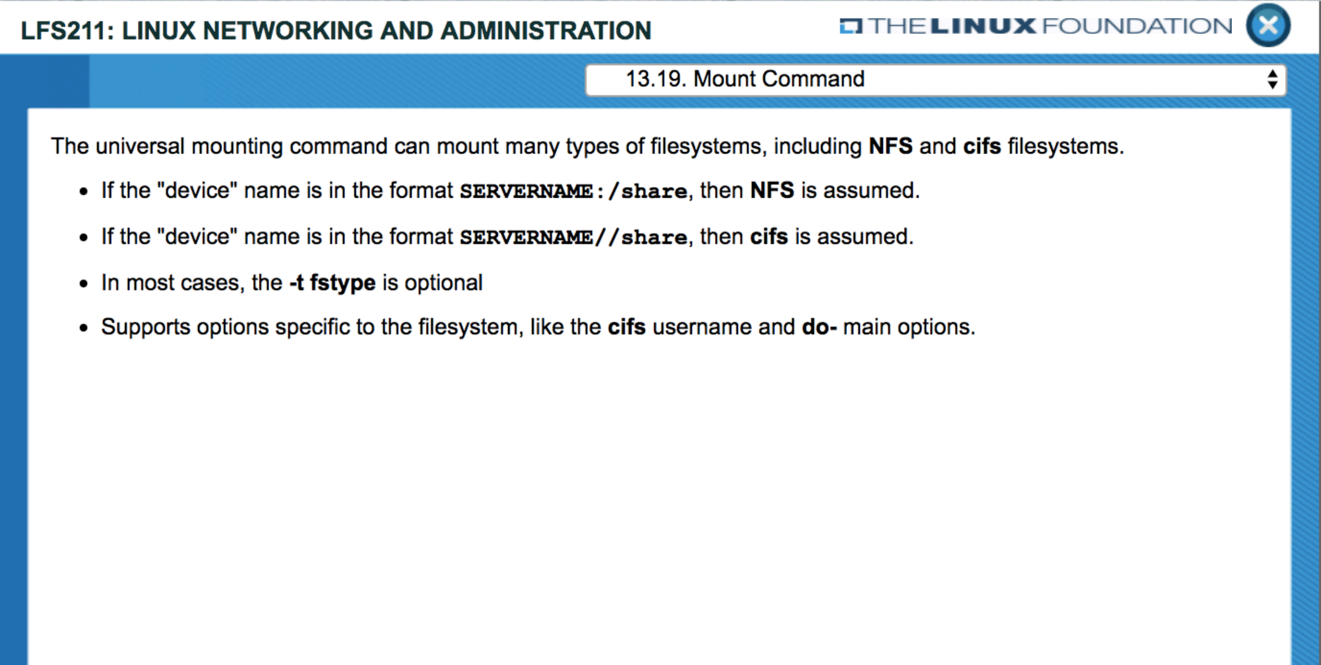


13.18. Mounting Network Filesystems

The three methods of mounting network file systems (immediate mount - command line-, always mounted, and mounted on demand) use a common configuration file: `/etc/fstab`. Over the years, additional features and services have updated the options in `/etc/fstab`. Functionality that required external packages, services and configuration are now combined under **systemd** services.

The `/etc/fstab` has slightly changed its purpose with **systemd** from configuration file to drive the **mount** command during system initialization to an input configuration file for the **systemd-fstab-generator** that creates native **unit** files used by **systemd.mount** during system startup. The regular **mount** command still looks in the `/etc/fstab` file.

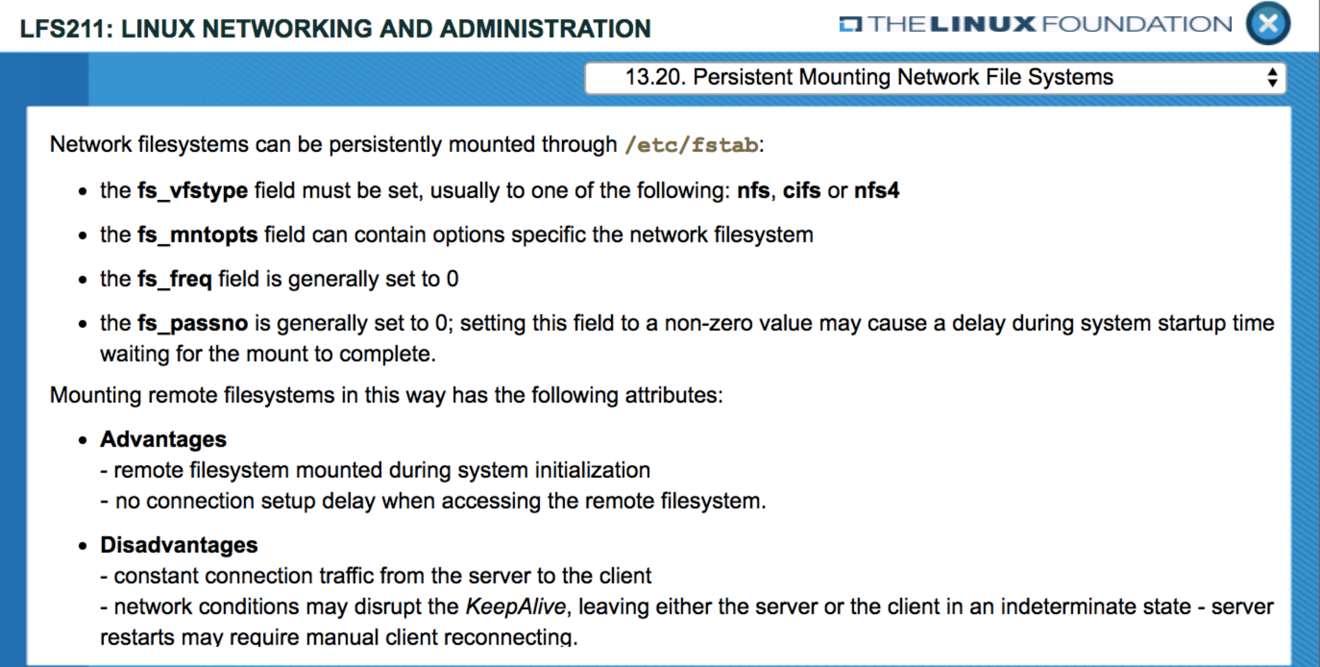
Since the preferred method to configure file system mounts is still `/etc/fstab`, nothing has changed for most **mount** options. See **man systemd.mount** for additional details.



13.19. Mount Command

The universal mounting command can mount many types of filesystems, including **NFS** and **cifs** filesystems.

- If the "device" name is in the format `SERVERNAME:/share`, then **NFS** is assumed.
- If the "device" name is in the format `SERVERNAME//share`, then **cifs** is assumed.
- In most cases, the **-t fstype** is optional
- Supports options specific to the filesystem, like the **cifs** username and **do- main** options.



13.20. Persistent Mounting Network File Systems

Network filesystems can be persistently mounted through `/etc/fstab`:

- the **fs_vfstype** field must be set, usually to one of the following: **nfs**, **cifs** or **nfs4**
- the **fs_mntopts** field can contain options specific the network filesystem
- the **fs_freq** field is generally set to 0
- the **fs_passno** is generally set to 0; setting this field to a non-zero value may cause a delay during system startup time waiting for the mount to complete.

Mounting remote filesystems in this way has the following attributes:

- **Advantages**
  - remote filesystem mounted during system initialization
  - no connection setup delay when accessing the remote filesystem.
- **Disadvantages**
  - constant connection traffic from the server to the client
  - network conditions may disrupt the *KeepAlive*, leaving either the server or the client in an indeterminate state - server restarts may require manual client reconnecting.

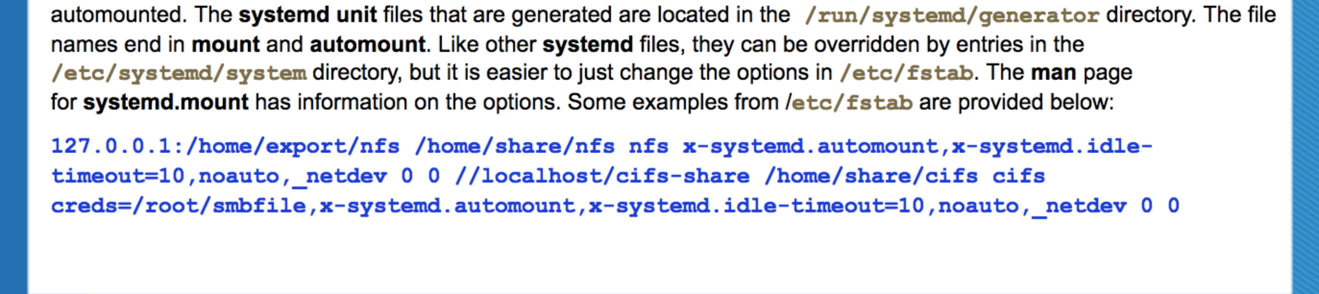
13.21. Automount Network File Systems

The automount will monitor a configured mount point and, if the information in the filesystem is accessed, the mount will be executed.

There may be an option to disconnect an idle connection. There are a couple of options for automounting:

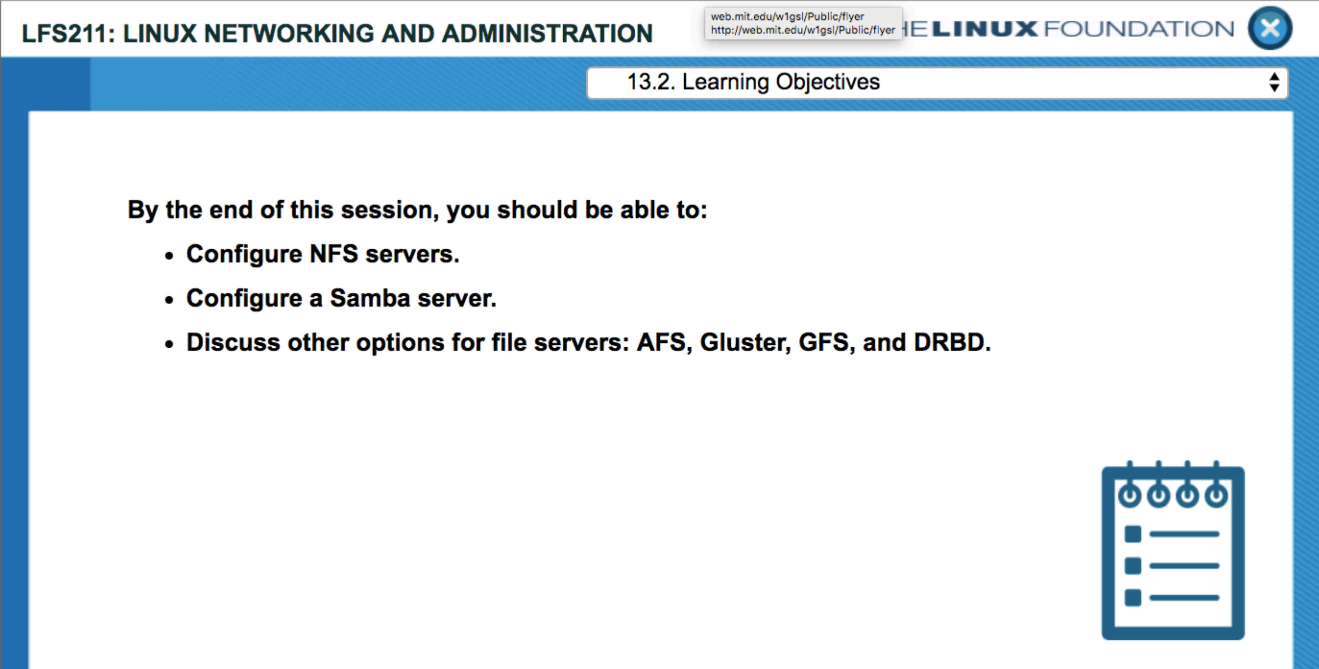
- **autofs**
  - requires additional packages to be installed
  - multi-part configuration files
  - may require scripts as helpers for mounting some filesystem
- **systemd.automount**
  - integrated into **systemd**
  - only available on **systemd** distros
  - unified configuration files, uses `/etc/fstab`
  - uses common **systemd.mount** facilities, no scripts, but allows unit file overrides.

With the **systemd.automount** implementation, the actual **unit** configuration files are generated from the entries in `/etc/fstab`. There are options to add to the `/etc/fstab` to indicate to the **systemd-fstab-generator** the entries are to be automounted. The **systemd unit** files that are generated are located in the `/run/systemd/generator` directory. The file



automounted. The **systemd unit** files that are generated are located in the `/run/systemd/generator` directory. The file names end in **mount** and **automount**. Like other **systemd** files, they can be overridden by entries in the `/etc/systemd/system` directory, but it is easier to just change the options in `/etc/fstab`. The **man** page for **systemd.mount** has information on the options. Some examples from /`etc/fstab` are provided below:

```
127.0.0.1:/home/export/nfs /home/share/nfs nfs x-systemd.automount,x-systemd.idle-
timeout=10,noauto,_netdev 0 0 //localhost/cifs-share /home/share/cifs cifs
creds=/root/smbfile,x-systemd.automount,x-systemd.idle-timeout=10,noauto,_netdev 0 0
```

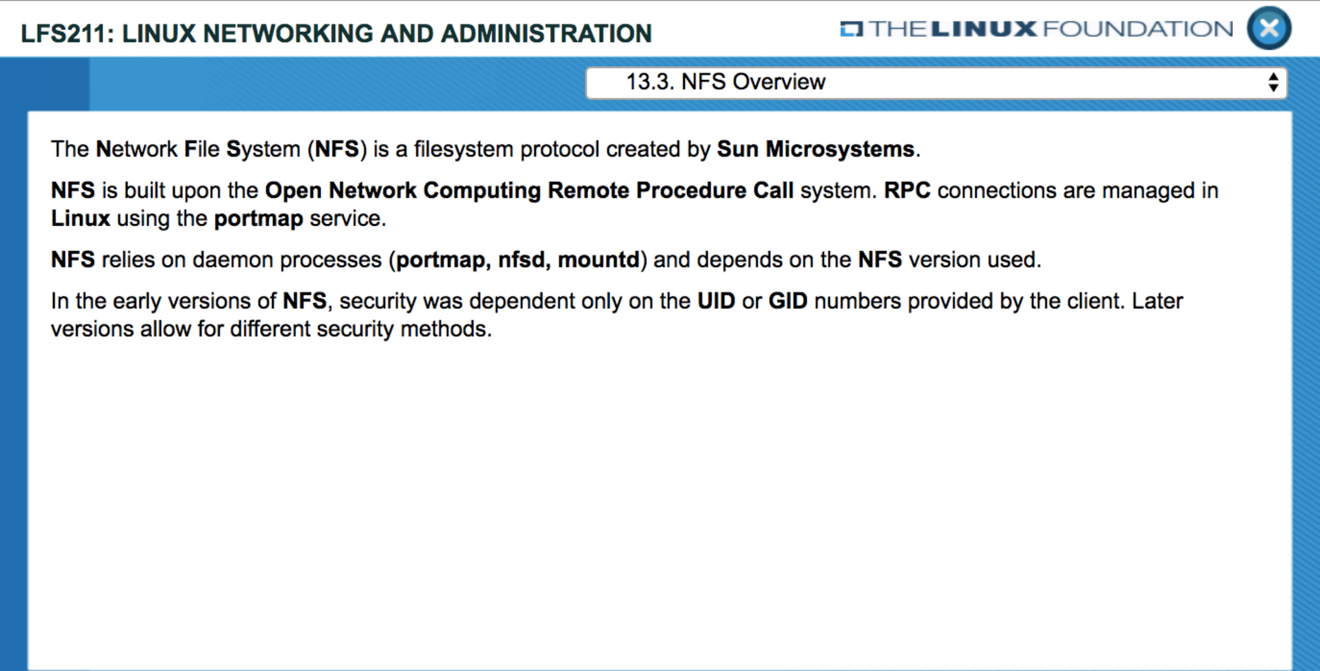


web.mit.edu/w1gsl/Public/flyer
http://web.mit.edu/w1gsl/Public/flyer

13.2. Learning Objectives

By the end of this session, you should be able to:

- **Configure NFS servers.**
- **Configure a Samba server.**
- **Discuss other options for file servers: AFS, Gluster, GFS, and DRBD.**



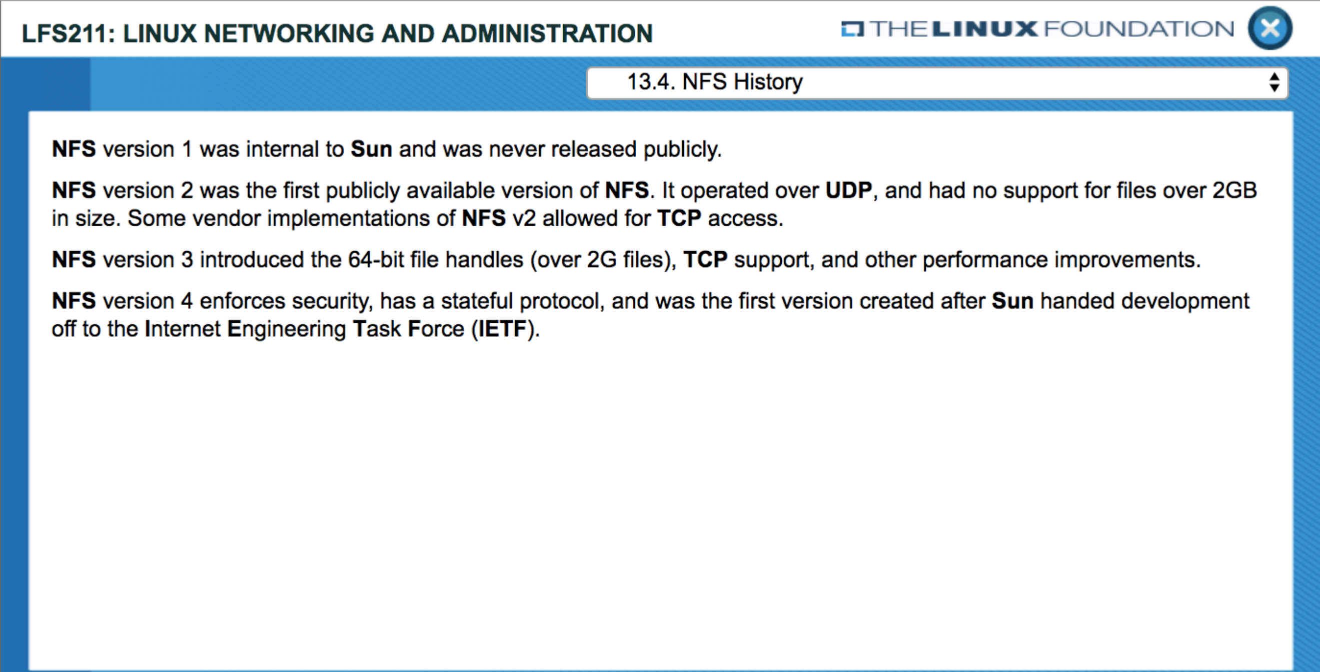
13.3. NFS Overview

The **N**etwork **F**ile **S**ystem (**NFS**) is a filesystem protocol created by **Sun Microsystems**.

NFS is built upon the **Open Network Computing Remote Procedure Call** system. **RPC** connections are managed in **Linux** using the **portmap** service.

NFS relies on daemon processes (**portmap, nfsd, mountd**) and depends on the **NFS** version used.

In the early versions of **NFS**, security was dependent only on the **UID** or **GID** numbers provided by the client. Later versions allow for different security methods.



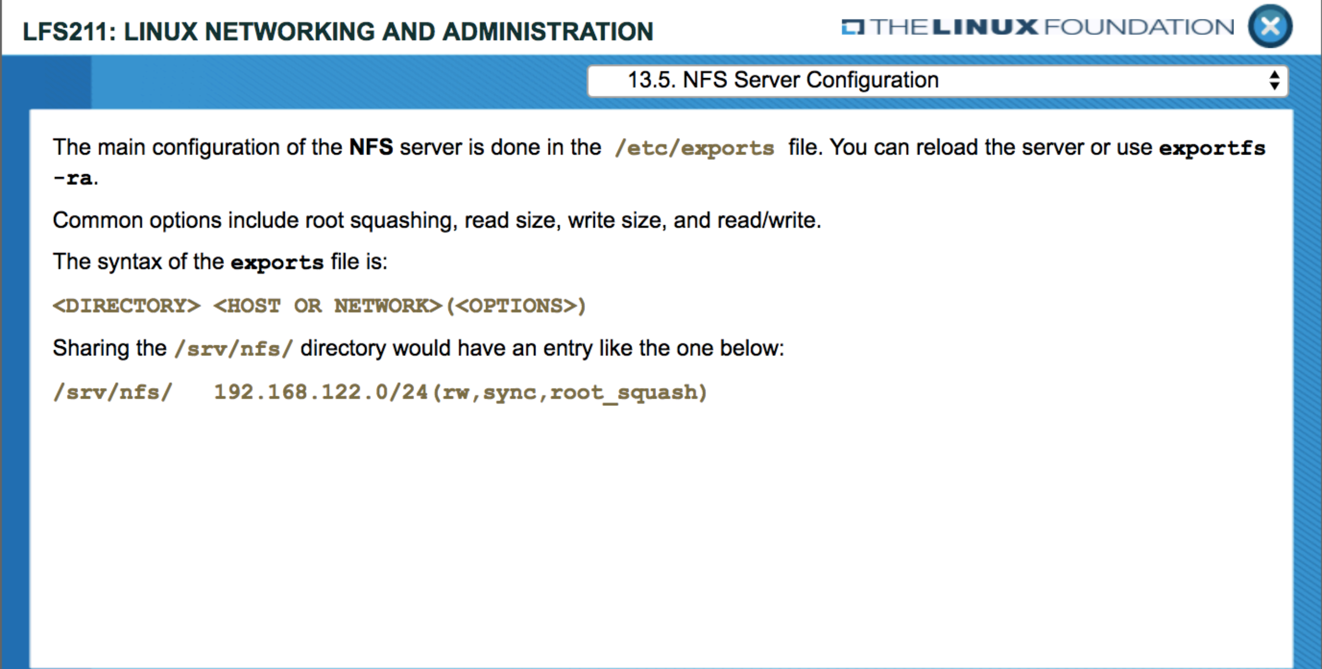
13.4. NFS History

NFS version 1 was internal to **Sun** and was never released publicly.

NFS version 2 was the first publicly available version of **NFS**. It operated over **UDP**, and had no support for files over 2GB in size. Some vendor implementations of **NFS** v2 allowed for **TCP** access.

NFS version 3 introduced the 64-bit file handles (over 2G files), **TCP** support, and other performance improvements.

NFS version 4 enforces security, has a stateful protocol, and was the first version created after **Sun** handed development off to the **I**nternet **E**ngineering **T**ask **F**orce (**IETF**).



13.5. NFS Server Configuration

The main configuration of the **NFS** server is done in the `/etc/exports` file. You can reload the server or use **`exportfs -ra`**.

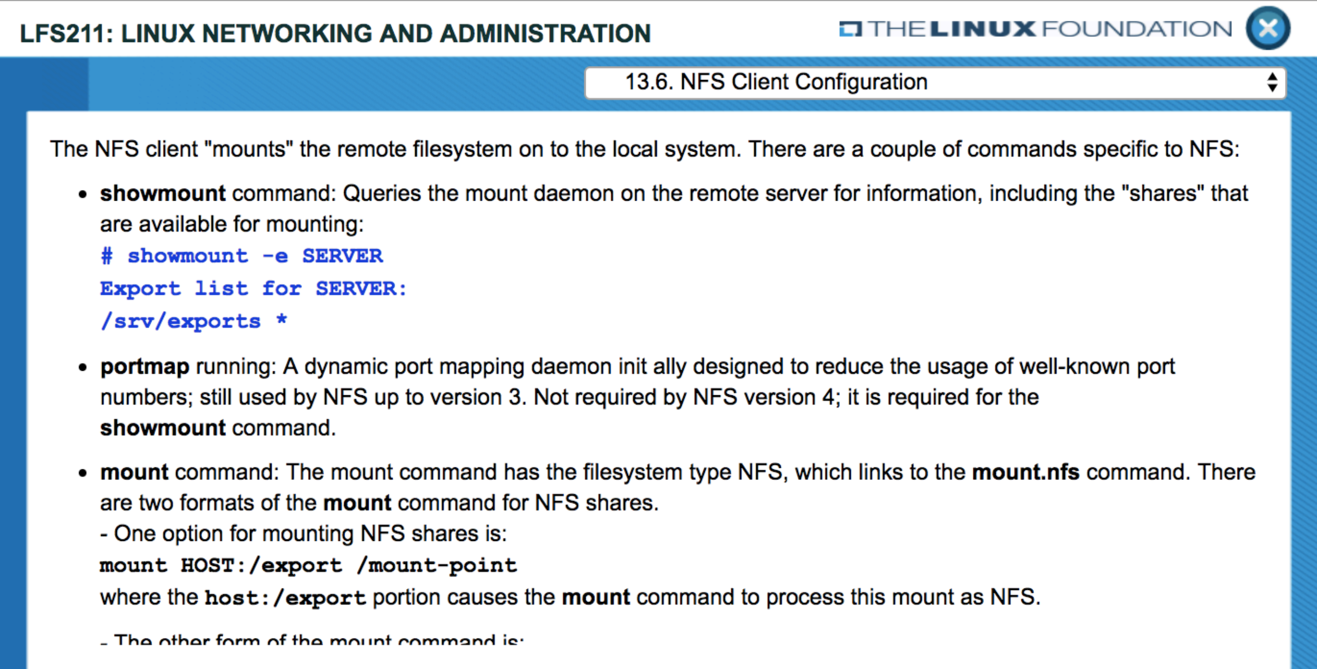
Common options include root squashing, read size, write size, and read/write.

The syntax of the **`exports`** file is:

```
<DIRECTORY> <HOST OR NETWORK>(<OPTIONS>)
```

Sharing the `/srv/nfs/` directory would have an entry like the one below:

```
/srv/nfs/   192.168.122.0/24(rw,sync,root_squash)
```



13.6. NFS Client Configuration

The NFS client "mounts" the remote filesystem on to the local system. There are a couple of commands specific to NFS:

- **showmount** command: Queries the mount daemon on the remote server for information, including the "shares" that are available for mounting:
  ```
  # showmount -e SERVER
  Export list for SERVER:
  /srv/exports *
  ```
- **portmap** running: A dynamic port mapping daemon init ally designed to reduce the usage of well-known port numbers; still used by NFS up to version 3. Not required by NFS version 4; it is required for the **showmount** command.
- **mount** command: The mount command has the filesystem type NFS, which links to the **mount.nfs** command. There are two formats of the **mount** command for NFS shares.
  - One option for mounting NFS shares is:
  ```
  mount HOST:/export /mount-point
  ```
  where the `host:/export` portion causes the **mount** command to process this mount as NFS.

  - The other form of the mount command is:

- **portmap** running: A dynamic port mapping daemon init ally designed to reduce the usage of well-known port numbers; still used by NFS up to version 3. Not required by NFS version 4; it is required for the **showmount** command.

- **mount** command: The mount command has the filesystem type NFS, which links to the **mount.nfs** command. There are two formats of the **mount** command for NFS shares.
  - One option for mounting NFS shares is:

  ```
  mount HOST:/export /mount-point
  ```

  where the **`host:/export`** portion causes the **mount** command to process this mount as NFS.

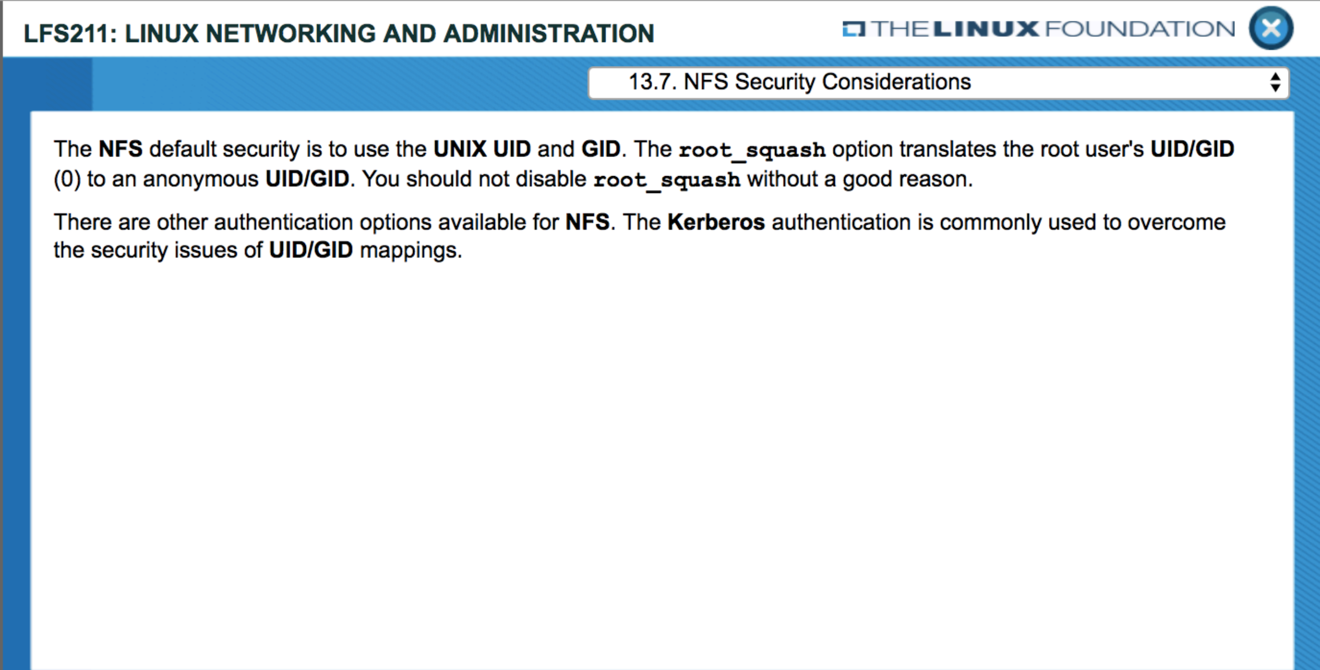
  - The other form of the mount command is:

  ```
  mount -t NFS HOST:/export /mountpoint
  ```

  which specifies NFS is being used.

  ```
  # mount SERVER:/share /mnt/share
  ```

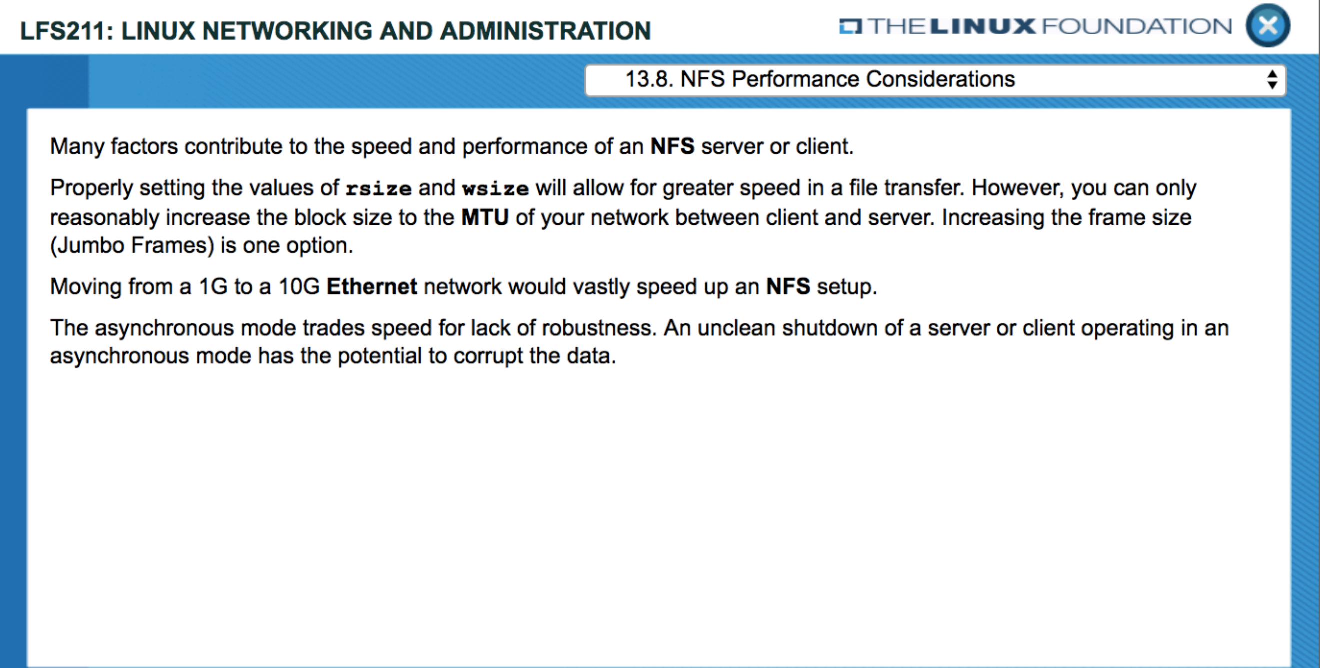
  Note that both examples above use the construct **`server:/sharename`** for mounting. The *sharename* is discovered with the **showmount -e servername** command.



13.7. NFS Security Considerations

The **NFS** default security is to use the **UNIX UID** and **GID**. The `root_squash` option translates the root user's **UID/GID** (0) to an anonymous **UID/GID**. You should not disable `root_squash` without a good reason.

There are other authentication options available for **NFS**. The **Kerberos** authentication is commonly used to overcome the security issues of **UID/GID** mappings.



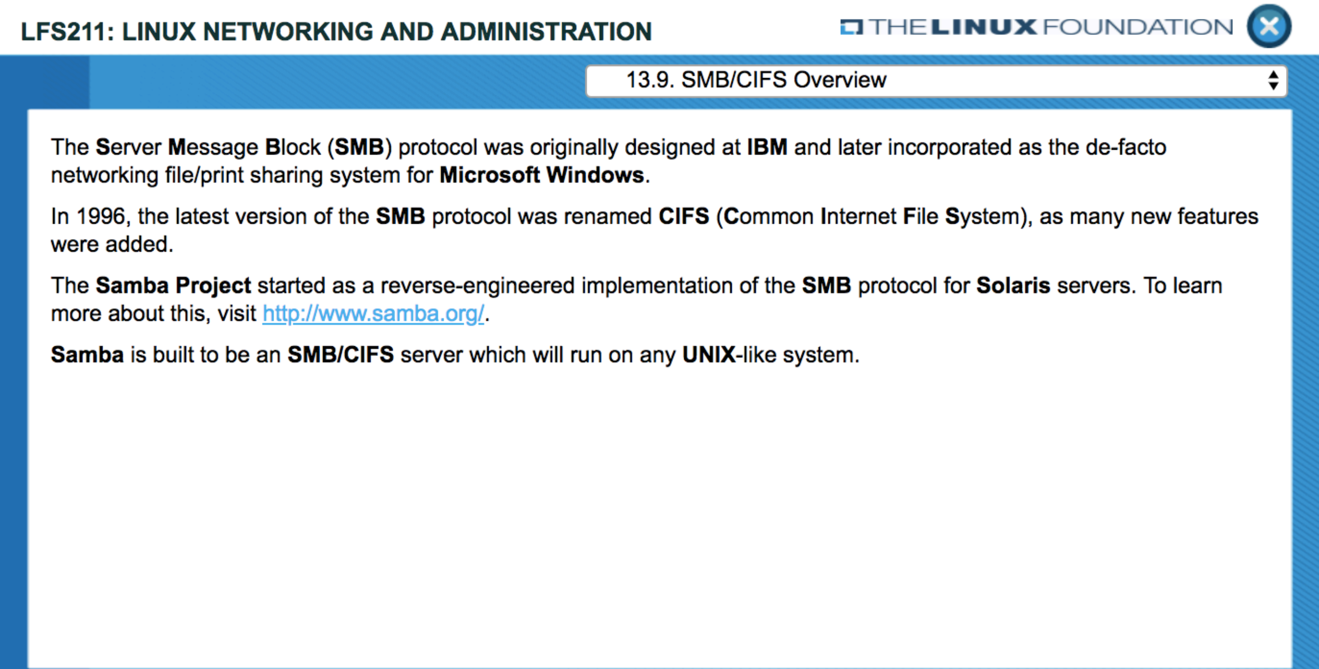
13.8. NFS Performance Considerations

Many factors contribute to the speed and performance of an **NFS** server or client.

Properly setting the values of **`rsize`** and **`wsize`** will allow for greater speed in a file transfer. However, you can only reasonably increase the block size to the **MTU** of your network between client and server. Increasing the frame size (Jumbo Frames) is one option.

Moving from a 1G to a 10G **Ethernet** network would vastly speed up an **NFS** setup.

The asynchronous mode trades speed for lack of robustness. An unclean shutdown of a server or client operating in an asynchronous mode has the potential to corrupt the data.



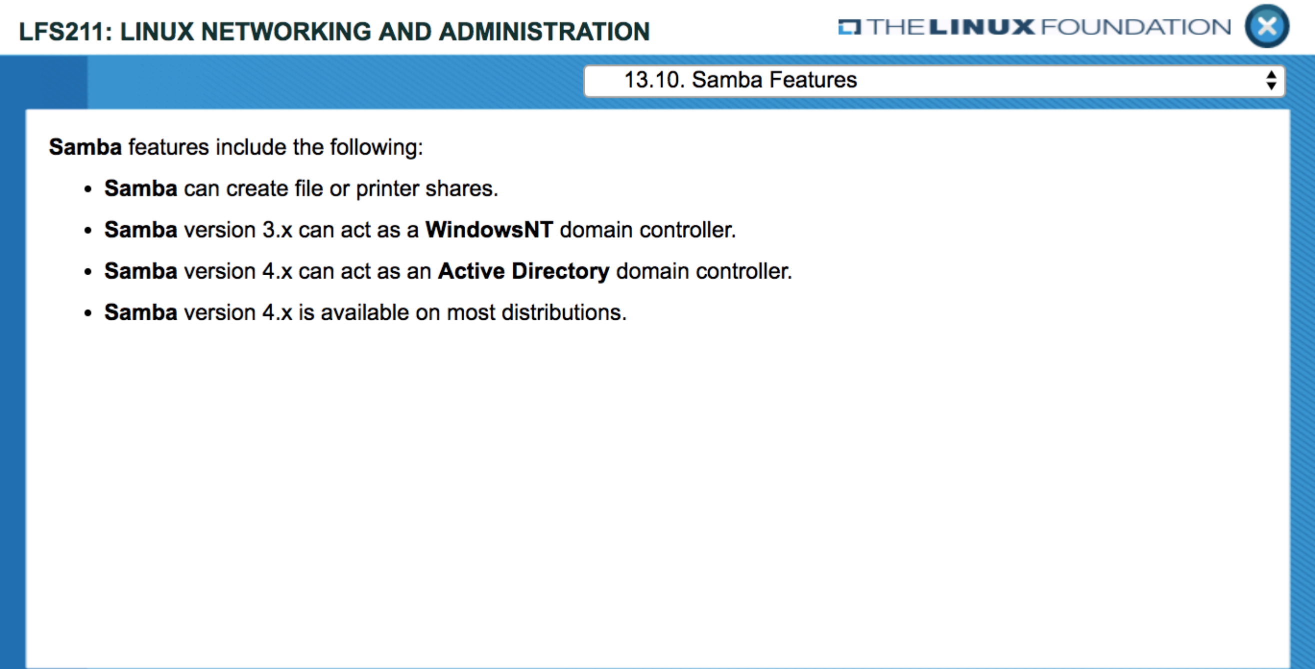
13.9. SMB/CIFS Overview

The **S**erver **M**essage **B**lock (**SMB**) protocol was originally designed at **IBM** and later incorporated as the de-facto networking file/print sharing system for **Microsoft Windows**.

In 1996, the latest version of the **SMB** protocol was renamed **CIFS** (**C**ommon **I**nternet **F**ile **S**ystem), as many new features were added.

The **Samba Project** started as a reverse-engineered implementation of the **SMB** protocol for **Solaris** servers. To learn more about this, visit http://www.samba.org/.

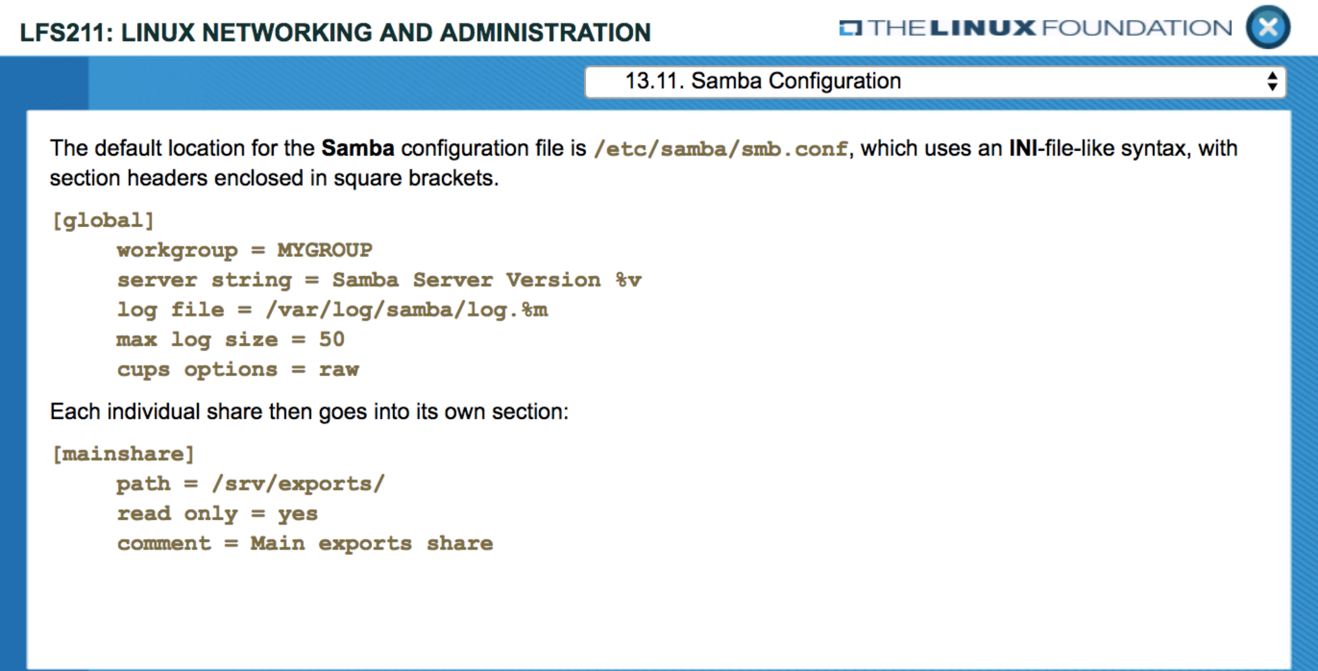
Samba is built to be an **SMB/CIFS** server which will run on any **UNIX**-like system.



13.10. Samba Features

Samba features include the following:

- **Samba** can create file or printer shares.
- **Samba** version 3.x can act as a **WindowsNT** domain controller.
- **Samba** version 4.x can act as an **Active Directory** domain controller.
- **Samba** version 4.x is available on most distributions.



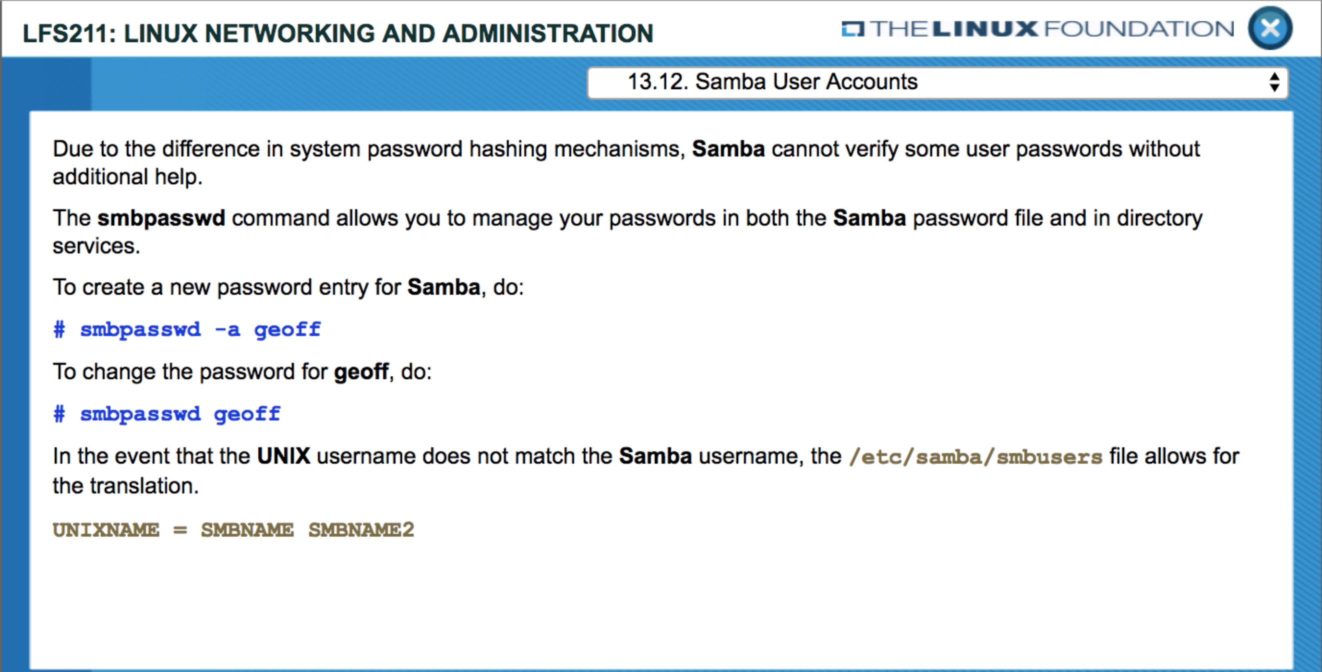
13.11. Samba Configuration

The default location for the **Samba** configuration file is `/etc/samba/smb.conf`, which uses an **INI**-file-like syntax, with section headers enclosed in square brackets.

```
[global]
     workgroup = MYGROUP
     server string = Samba Server Version %v
     log file = /var/log/samba/log.%m
     max log size = 50
     cups options = raw
```

Each individual share then goes into its own section:

```
[mainshare]
     path = /srv/exports/
     read only = yes
     comment = Main exports share
```



13.12. Samba User Accounts

Due to the difference in system password hashing mechanisms, **Samba** cannot verify some user passwords without additional help.

The **smbpasswd** command allows you to manage your passwords in both the **Samba** password file and in directory services.

To create a new password entry for **Samba**, do:

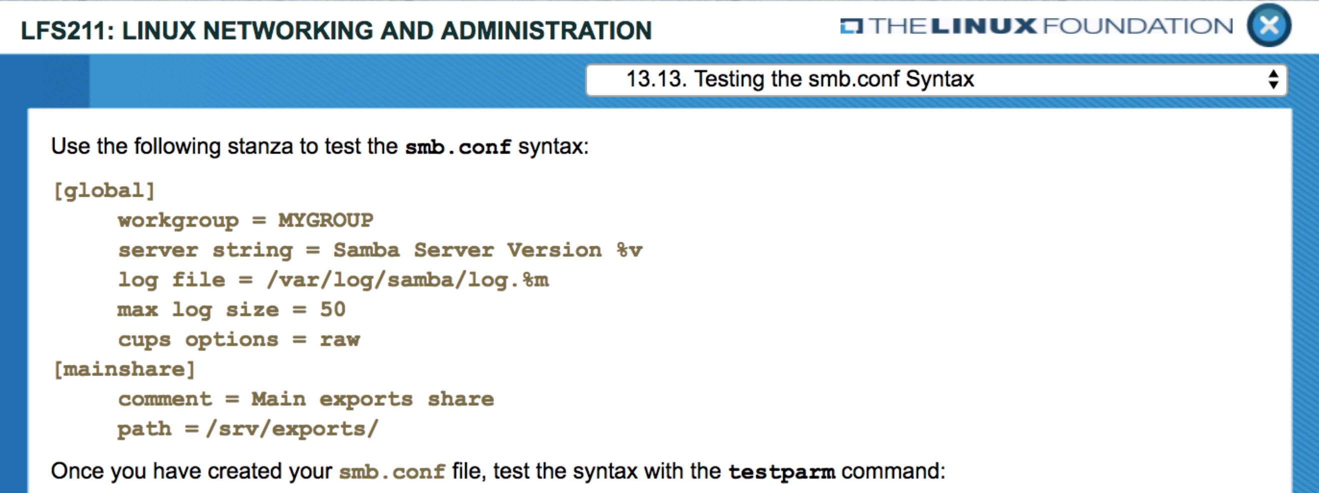
```
# smbpasswd -a geoff
```

To change the password for **geoff**, do:

```
# smbpasswd geoff
```

In the event that the **UNIX** username does not match the **Samba** username, the `/etc/samba/smbusers` file allows for the translation.

```
UNIXNAME = SMBNAME SMBNAME2
```



13.13. Testing the smb.conf Syntax

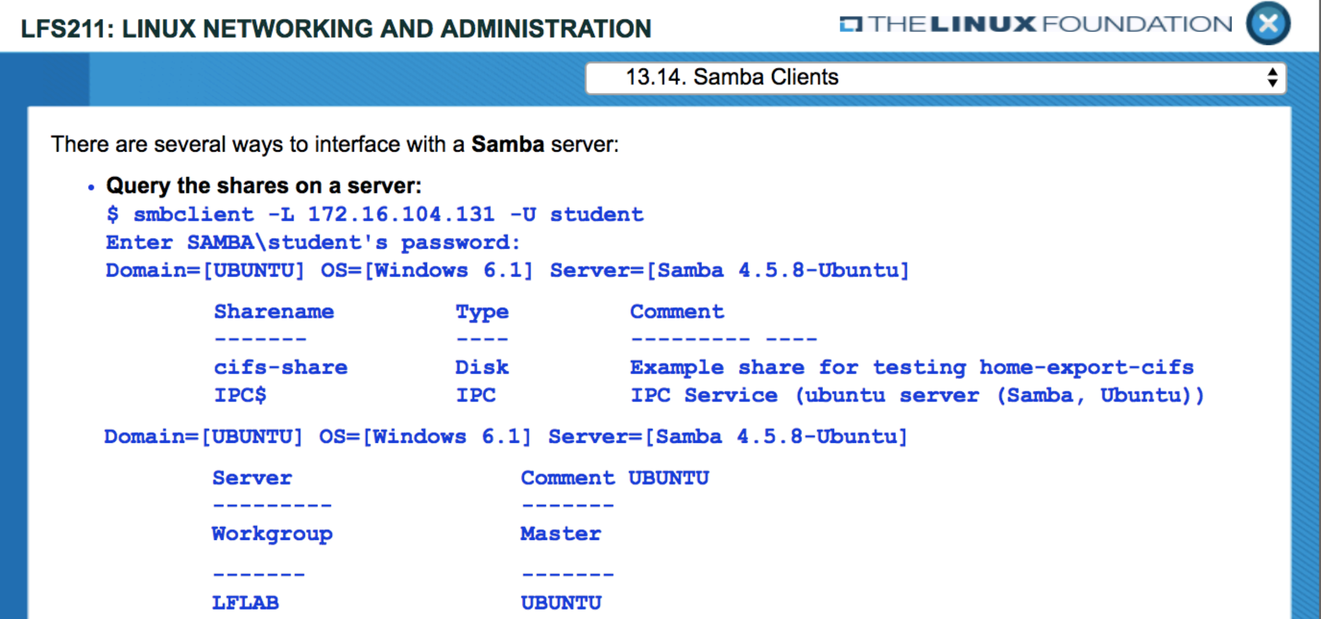
Use the following stanza to test the **smb.conf** syntax:

```
[global]
     workgroup = MYGROUP
     server string = Samba Server Version %v
     log file = /var/log/samba/log.%m
     max log size = 50
     cups options = raw
[mainshare]
     comment = Main exports share
     path =/srv/exports/
```

Once you have created your `smb.conf` file, test the syntax with the **testparm** command:

```
# testparm
Load smb configuration files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit
(16384)
Processing section "[mainshare]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
```

When you press enter, you will get a parsed copy of the configuration file.



LFS211: LINUX NETWORKING AND ADMINISTRATION

13.14. Samba Clients

There are several ways to interface with a **Samba** server:

- **Query the shares on a server:**

```
$ smbclient -L 172.16.104.131 -U student
Enter SAMBA\student's password:
Domain=[UBUNTU] OS=[Windows 6.1] Server=[Samba 4.5.8-Ubuntu]

        Sharename         Type         Comment
        -------           ----         --------- ----
        cifs-share        Disk         Example share for testing home-export-cifs
        IPC$              IPC          IPC Service (ubuntu server (Samba, Ubuntu))

Domain=[UBUNTU] OS=[Windows 6.1] Server=[Samba 4.5.8-Ubuntu]

        Server                 Comment UBUNTU
        ---------              -------
        Workgroup              Master

        -------                -------
        LFLAB                  UBUNTU
```

- Using the **FTP**-like **smbclient** command:

```
$ smbclient //SERVER/mainshare
smb: \> get /foo
```

- Mounting the **SMB/CIFS** share into your name space:

```
# mount -t cifs -o username,password //SERVER/share/ /mnt/point/
```

To avoid putting usernames and passwords into a world-readable file (`/etc/fstab`), the **CIFS mount** command takes the **`credentials=filename`** option.

```
# mount -t cifs -o credentials=filename //SERVER/share/ /mnt/point/
```

The credentials file has the following syntax:

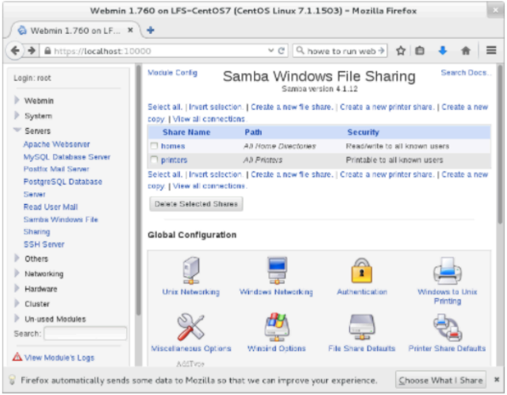
```
username=value
password=value
domain=value
```

13.15. Samba Tools

The distribution-specific tool for **CentOS, system-config-samba**, has been deprecated and removed. The **Samba** organization has also deprecated and removed the included web administration tool **SWAT**.

Several other graphical interfaces for **Samba** are available. For more information, see http://www.samba.org/samba/GUI.

One popular tool is the **webmin** tool, available from http://www.webmin.com.



Webmin 1.760 on LFS-CentOS7 (CentOS Linux 7.1.1503) - Mozilla Firefox

Samba Windows File Sharing

Samba version 4.1.12

Share Name	Path	Security
□ homes	All Home Directories	Read/write to all known users
□ printers	All Printers	Printable to all known users

Global Configuration

Firefox automatically sends some data to Mozilla so that we can improve your experience.

Figure 13.1: Webmin Samba