Using fstab

A Beginner's Guide

Table of Contents

1 Colophon

2 Introduction

This is a guide to using `/etc/fstab` to mount disc partitions and network shares on Raspberry Pi OS while avoiding most of the common pitfalls that can result in a system that won't boot.

While aimed at Raspberry Pi OS much will apply to any Linux distribution.

It is assumed that the reader has a basic familiarity with the Linux command line and at least one text editor. Desktop users will need to open a terminal to execute many of the commands in this guide.

2.1 What’s Not Covered

- Choosing which file system to use.
- Partitioning drives.
- Formatting drives/partitions.
- Configuring networking on your Pi.
- Choosing which network protocol to use.
- Server configuration.
- Using SAN[1] protocols (iSCSI, ATAoE, et al).
- Migrating your OS from SD card to USB or network storage.
- NOOBS, Pinn, and other boot loaders.
- Loop mounts, bind mounts, using files as block devices.
- Use of desktop tools.
- Hardware issues. For example CM, CM3, CM3+, CM4, A, A+, 3A+, power supply, USB hubs, USB chipset quirks.
- IPv6

[1] Storage Area Network

2.2 Requirements:

- Raspberry Pi (any model) and the normal accessories.
- A user account on your Pi with permission to use `sudo` or the ability to login as root.
- A USB drive or network share to be mounted.
- Optional but highly recommended:
  - A second SD card with Raspberry Pi OS installed or a computer capable of reading and writing your root file system (a Linux PC or Windows PC with the appropriate drivers installed).
  - A USB SD card reader

2.3 Conventions

```
Text like this indicates input to or output from the command line.
```

`Text like this` also refers to full or partial commands but is not generally intended to be entered into the command line as is.

“SD card” refers equally to full size and micro SD cards.

3 Read the Documentation

Before attempting to add to `/etc/fstab` read the relevant documentation.
This can most easily be achieved by reading the manual pages:

```
man mount

man fstab
```

For some files systems additional information is available via

```
man mount.filesystemname
```

e.g.

```
man mount.cifs
```

4 Create A Mount Point

A mount point is just a directory somewhere in your file system hierarcy. It does not have to be empty[2]. It does not need any particular permissions. It does not need to be owned by root.

While there are commonly used locations for mount points (see here) you do not have to stick with those conventions.

There are no restrictions other than those imposed by the file system in what you call your mount points.

Mount points are created with the `mkdir` command:

```
sudo mkdir -p /path/to/mountpoint
```

`sudo` runs `mkdir` as root, "`-p`" instructs `mkdir` to create any directories in “/path/to/mountpoint” that do not already exist.

Replace “/path/to/mountpoint” as required.

Do not use a space, tab, or other special character anywhere in “/path/to/mountpoint”. Doing so will cause `/etc/fstab` to be miss-parsed.

I strongly suggest that the new mount point is made read only to all users. While this will have no impact when something has been mounted there, it will prevent accidental writes to your SD card should nothing be mounted.

Run:

```
sudo chmod a-w /path/to/mountpoint
```

Skip this if mounting onto a system created directory such as /mnt or /srv to avoid potential conflicts with other utilities.

[2] If the mount point is not empty files and directories within it will not be accessible once something is mounted on it. They still take up space and are still counted by tools such as `df`.

5 Identifying A Partition

A drive can have multiple partitions (e.g. the Raspberry Pi OS SD card). It is partitions that get mounted not drives.

Partitions can be identified in a number of ways:

- By device node (A.K.A. their entry in /dev)
- By label
- By UUID
- By PARTUUID

Device node can change across boots where more than one drive is connected.

Label is neither expected nor guaranteed to be unique.

UUID should be unique but maybe long and cannot be used in cmdline.txt to specify the root partition.

PARTUUID should be unique and is shorter than UUID.

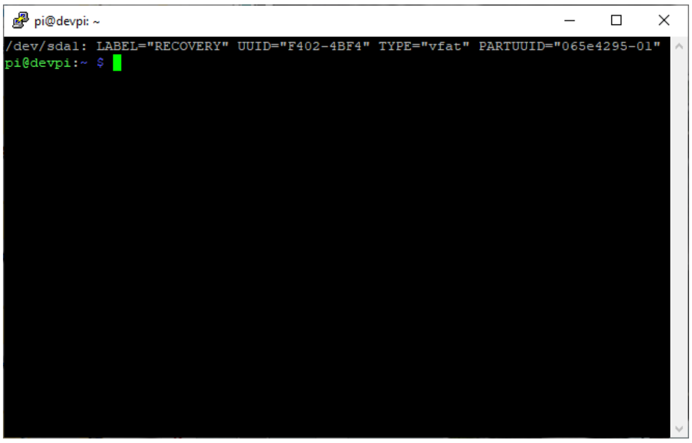
Note: both UUID and PARTUUID will not be unique if multiple drives connected were cloned from the same source drive or written from the same .img file.

We'll be using PARTUUID. This can be obtained by running

```
sudo blkid|grep -v mmcblk0
```

`grep -v mmcblk0` discards any results for the SD card. Omit this if the partition you want to mount is on the SD card in the on board slot.

Output should be similar to this:



```
/dev/sda1: LABEL="RECOVERY" UUID="F402-4BF4" TYPE="vfat" PARTUUID="065e4295-01"
pi@devpi:~ $
```

Make a note of the values for TYPE and PARTUUID. In the above example that's vfat and 065e4295-01

6 Identifying A Network Share

The most likely server types are CIFS[3] (Windows network) and NFS[4] (Linux, Unix, etc). Others exist but are less likely to be encountered so are not covered here. Connection to a CIFS server in a windows domain is also outside the scope of this document, as is server configuration.

CIFS servers include, but are not limited to, Linux computers running Samba, computers running Windows Server OSes, Windows desktop PCs with shared folders, dedicated NAS[5] boxes, and domestic routers.

NFS servers include, but are not limited to, Linux computers and dedicated NAS boxes.

6.1 CIFS

Note down the server name or IP address, the share name, the login credentials if any, and whether guest access is permitted or not. Server and share names are not case sensitive.

For example:

- Server name: foo
- IP address: 10.0.0.1
- Share name: bar
- Credentials: Username=fred, Password=derf
- Guest OK: yes

6.2 NFS

Note down the server name or IP address, and the full path to the exported directory as seen by the server. Server name is not case sensitive but the path to the export is.

For example:

- Server name: foo
- IP address: 10.0.0.1
- Export path: /srv/bar

[3] Common Internet File System. Also know as SMB and, incorrectly, samba.

[4] Network File System

[5] Network Attached Storage

6.3 All Network File Systems

Only use the IP address of a server if it is fixed and your Pi is unable to connect using the hostname.

mDNS/avahi/bonjour style hostnames (`hostname.local`) may be used provided both the Pi and the server have the correct software installed.

7 Test Mount and Determine the Required Options

7.1 Partitions

1. Manually mount the partition with the default options:

   ```
   sudo mount -t type PARTUUID=partuuid /path/to/mountpoint
   ```

   Frome the example in section 5:

   ```
   sudo mount -t vfat PARTUUID=065e4295-01 /path/to/mountpoint
   ```

2. If the mount fails, check that the mount point exists and that the PARTUUID is correct. For some rarer filesystems it may be necessary to install a driver. Searching the web or the package manager should point you in the right direction.

3. Check that your users have the permissions on the mounted drive they need.

   The most common issue here is that the default permissions only allow writing with `sudo` or as the root user. The fix is dependent on the file system in use:

   ○ **ext4** (and most Linux file systems): use `chmod` to set permissions on the mount point and any existing files/directories under it.

   ○ **vfat** (and all other FAT variants): Permissions are set a mount time and cannot be changed. The simplest way is to include `umask=000` in the mount options, e.g.

   ```
   sudo mount -t vfat -o umask=000 PARTUUID=065e4295-01
   /path/to/mountpoint
   ```

   ○ **NTFS**: Like FAT, permissions are set a mount time and cannot be changed.

   Check if ntfs-3g is installed. If not install it via `apt` and try again. If writing is still not possible try including `allow_other` in your mount options, e.g.

   ```
   sudo mount -t ntfs -o allow_other PARTUUID=065e4295-01
   /path/to/mountpoint
   ```

   You will need to `umount` the partition before retrying the mount with new options:

   ```
   sudo umount /path/to/mountpoint
   ```

   Be aware that setting the `umask` as above for FAT and `allow_other` for NTFS will allow any user to write to the partition. There are more secure, but more complex, ways of achieving write access, see the relevant man pages.

4. Make a note of all mount options used.

7.2 Network Shares

7.2.1 Server Connection

Check that your Pi can reach the server:

```
ping -c 5 servername
```

If that fails try:

```
ping -c 5 servername.local
```

If that fails, try:

```
ping -c 5 server’s IP address
```

If that fails, check that the server is on and check the network configuration of your Pi.

For the example server in section 6 the commands would be:

```
ping -c 5 foo

ping -c 5 foo.local

ping -c 5 10.0.0.1
```

7.2.2 CIFS

1. Manually mount the partition with the default options:

   ```
   sudo mount -t cifs -o user=username,password=password
   //servername/sharename /path/to/mountpoint
   ```

   If the server allows guest access, use this instead:

   ```
   sudo mount -t cifs -o guest //servername/sharename /path/to/mountpoint
   ```

   For the example server in section 6 the commands would be:

   ```
   sudo mount -t cifs -o user=fred,password=derf //foo/bar /mnt
   ```

   or

   ```
   sudo mount -t cifs -o guest //foo/bar /mnt
   ```

   The mount may fail due to protocol version being different between your Pi and the server[6]. If so, add `vers=1.0` to your mount options and try again. For example:

   ```
   sudo mount -t cifs -o guest,vers=1.0 //foo/bar /mnt
   ```

   This is an old, insecure, and no longer developed version of the protocol so only do this as a last resort.

2. Check your users have the desired permissions on the mounted share.

[6] Most likely with domestic routers.

The most common issue here is that the default permissions only allow writing with `sudo` or as the root user and, as most CIFS servers do not support them, permissions cannot be changed.

3. The simplest solution is to add `file_mode=0777` and `dir_mode=0777` to your mount options however this allows all users on your Pi to read and write to the share once it is mounted. See the relevant manual pages for more secure options.

4. For example:

```
5. umount /path/to/mountpoint

6. sudo mount -t cifs -o guest,file_mode=0777,dir_mode=0777
   //servername/sharename /path/to/mountpoint
```

7. If you are still unable to write to the share, check the server configuration. Client mount options cannot provide permissions that the server denies.

8. Make a note of all mount options used.

7.2.3 NFS

1. Manually mount the partition with the default options:

   ```
   sudo mount -t nfs servername:/export/path /path/to/mountpoint
   ```

   For the example server in section 6:

   ```
   sudo mount -t nfs foo:/srv/bar /mnt
   ```

2. Check your users have the desired permissions on the mounted share.

   NFS supports Linux permissions[7] so if you’re unable to read from or write to the shared directories and files try changing their permissions via `chmod` or `chown`[8]. If those fail, check the server configuration allows writing.

3. Make a note of all mount options used.

[7] Assuming the underlying file system on the server supports them.

[8] This may need to be done as root or with sudo.

8 Updating /etc/fstab

By now you should have all the information needed.

- PARTUUID:
- Share Address:
- File System Type:
- Mount Point:
- Mount Options:

While optional, I strongly recommend using the following additional mount options in your `/etc/fstab`:

• For partitions:	`defaults,noatime,nofail`
• For NFS exports:	`defaults,noatime,nofail,_netdev`
• For CIFS/SMB shares:	`defaults,nofail,_netdev`

`defaults` ensure the default options for the filesystem are used. It must be the first option given[9].

`noatime` prevents unnecessary writes.

`nofail` forces boot to continue should the mount fail.

`_netdev` flags the mount as needing the network to be available before it is attempted.

See the documentation for `mount` for more details.

[9] This is not strictly true but as later options override earlier ones it is best if defaults is first.

8.1 Adding An Entry To /etc/fstab

1. Backup your existing fstab:

   ```
   sudo cp /etc/fstab /etc/fstab.bak
   ```

2. Open `/etc/fstab` in your preferred text editor. You will need to use `sudo` or be logged in as root.

3. Find the end of the file.

4. Start a new line.

   An entry in `/etc/fstab` is on one line and contains multiple fields separated by white space (space or tab characters). Any lines starting with # are comments and are ignored.

   > The first field is your partition identifier or your share address.
   >
   > The second is the full path to your mount point.
   >
   > The third is the file system type.
   >
   > The fourth is the mount options. Commas in this field must not be followed by spaces.
   >
   > The fifth should be 0.
   >
   > The sixth should be either 0 (if you don’t want it checked at boot time and for network shares) or 2 (if you do).

5. Enter the details for your device:

   ```
   PARTUUID=partuuid /path/to/mountpoint filesystem_type mount_options 0
   0
   ```

   or, for CIFS

   ```
   //server/share /path/to/mountpoint cifs mount_options 0 0
   ```

   or, for NFS

   ```
   server:/export/path /path/to/mountpoint nfs mount_options 0 0
   ```

   For the examples in sections 5 and 6:

   ```
   PARTUUID=065e4295-01 /mnt/partition vfat
   defaults,noatime,nofail,guest,umask=000 0 0

   //foo/bar /mnt/cifs cifs
   defaults,noatime,nofail,_netdev,guest,file_mode=0777,dir_mode=0777 0 0

   foo:/srv/bar /mnt/nfs nfs defaults,noatime,nofail,_netdev 0 0
   ```

6. Save and close /etc/fstab.

7. Reboot.

8. Login and check that the mount has succeeded:

```
mountpoint /path/to/mountpoint
```

9 Troubleshooting

9.1 Boot Fails

1. Power down your Pi.
2. Remove the SD card[10].
3. Insert your second SD card with a clean OS into your Pi.
4. Reboot.
5. Insert the original SD card into your card reader, your card reader into your PI, and manually mount the second partition[11]:

   ```
   sudo mount /dev/sda2 /mnt
   ```

   The partition may not be `/dev/sda2`. Refer to section 5 to find the correct partition.
6. Open `/mnt/etc/fstab` in your preferred text editor. You will need to use `sudo` or be logged in as root.
7. Find the new entry.
8. Add `,noauto` to the end of the existing mount options. This will prevent any attempt to mount the device at boot.
9. Save and close the file.
10. Shutdown your PI.
11. Remove the card reader and reinsert the original SD card.
12. Reboot.
13. Attempt to mount the device:

    ```
       sudo mount /path/to/mountpoint
    ```

    `mount` should return an error. If no error is reported the mount has succeeded, this is most likely when mounting network shares.

    If this is the case, use `raspi-config` to check that your Pi is configured so that boot waits until a network connection is established (under “System Options”).
14. Using the error message as a guide correct the entry in `/etc/fstab`.
15. Save your `/etc/fstab` and retry the mount. If it succeeds, remove `,noauto` from the entry in `/etc/fstab`.

[10] A safe shutdown may not be possible, you may have no option but to pull the plug.

[11] If you have booted to the desktop the SD card may have been mounted for you. Adjust paths as necessary.

9.2 Boot Completes But The Device Is Not Mounted

1. Attempt to mount the device:

   ```
   sudo mount /path/to/mountpoint
   ```

   `mount` should return an error. If no error is reported the mount has succeeded, this is most likely when mounting network shares.

   If this is the case, use `raspi-config` to check that your Pi is configured so that boot waits until a network connection is established (under “System Options”).

2. Using the error message as a guide correct the entry in `/etc/fstab`.

3. Save your `/etc/fstab` and retry the mount.

9.3 Boot Fails If The Device Is Not Connected

1. Reconnect the device or boot from your second SD card (see 9.1, above).

2. Ensure that `nofail` and/or `noauto` are included in your mount options.

9.4 Device Is Not Mounted If Connected After Booting

If booting to the command line (or to the desktop without automatic login) this is the expected behaviour. Auto mounts in `/etc/fstab` are only processed during boot and when `sudo mount -a` is run.

Mounting within the desktop is outside the scope of this document.

9.5 Cannot Login To CIFS Share

9.5.1 With Username and Password

1. Ensure you have included the username and password. When using /etc/fstab they will not be prompted for.
2. Ensure the username and password are correct.
3. Ensure the username and password in use are those of the server not those of the local user.
4. Ensure the user exists on the server. Both as an OS user and, if it’s a samba server, a samba user.

9.5.2 With Guest Access

1. Ensure you have included the `guest` mount option. Without it a password is required which cannot be prompted for.
2. Ensure the server supports guest access on the share[12].

9.6 Cannot Connect To NFS Export

1. Check you are using the correct path and that its letter case is correct. NFS paths are case sensitive[13].
2. Check the server isn’t restricting access.

[12] It’s a per share option not a per server one.

[13] Usually. Though it depend on which file system the server is using.

10 Making Things More Secure

If you’re the only user on the system, are happy with the default options, or want access for all user skip this section.

10.1 All Mounts

The following additional mount options may be useful:

`ro`: Read only

`noexec`: Do not allow direct execution of any binaries on the mounted filesystem.

10.2 Linux Native File Systems & NFS

In most cases the standard mechanism should suffice.

10.3 FAT (all variants)

FAT is inherently insecure[14]. It does not directly support Linux permissions instead they are provided by the driver and fixed at mount time. The default permissions allow reading by all users and writing only by the user who mounted the device.

10.3.1Allow Access By Only One User

Replace the existing mount options with

```
defaults,noatime,nofail,uid=user,gid=group,umask=077
```

Omit `,umask=077` to allow other users read only access

For the pi user that would be:

```
defaults,noatime,nofail,uid=pi,gid=pi,umask=077
```

or

```
defaults,noatime,nofail,uid=pi,gid=pi
```

[14] FAT has no concept of file ownership, the only security “features” are the ability to mark things as read only and/or hidden. Linux permissions are set a mount time and cannot be changed without remounting the partition.

10.3.2Allow Access Only By A Group Of Users

1. Create a new group:

   ```
   sudo groupadd groupname
   ```

   e.g.

   ```
   sudo groupadd fatwrite
   ```

2. Add users to the group:

   ```
   sudo usermod -a -G groupname username
   ```

   e.g.

   ```
   sudo usermod -a -G fatwrite pi
   ```

3. Replace the existing mount options with

   ```
   defaults,noatime,nofail,gid=group,umask=707
   ```

   Change the `umask` to 202 to allow read only access to users who are not members of the group.

   For the pi user that would be:

   ```
   defaults,noatime,nofail,gid=fatwrite,umask=707
   ```

   or

   ```
   defaults,noatime,nofail,gid=fatwrite,umask=202
   ```

10.3.3Notes

All masks are the permissions you want to disallow. 000 gives all permissions to everyone. 777 gives no permissions to anyone.

`fmask` and `dmask` can be used instead of `umask` if different permissions are required for files (`fmask`) and directories (`dmask`).

10.4 NTFS

The in kernel NTFS driver does not allow write access regardless of mount options and permissions. The ntfs-3g driver does. This driver is installed on Raspberry Pi OS by default.

Access is controlled in the same way as for FAT. See 10.3, above. Like FAT, NTFS does not support Linux permissions.

10.5 CIFS

10.5.1Using A Credentials File

`/etc/fstab` is readable by all users. Passwords included here will not stay secret for long. For shares mounted automatically at boot it is safer to use a credentials file.

1. Open a new file in your preferred text editor.
2. Insert the following:

   ```
   username=name
   password=password
   ```

   e.g.

   ```
   username=fred
   password=derf
   ```

3. Save the file. Name and location is up to you. e.g.

   ```
   /root/credentials-foo-bar
   ```

4. Ensure root owns the file:

   ```
   sudo chown root:root /path/to/file
   ```

5. Change permissions so that only root can read and write to it:

   ```
   sudo chmod 600 /path/to/file
   ```

6. Update your /etc/fstab replacing `user=username,password=password` with

   ```
   credentials=/path/to/file
   ```

   e.g.

   ```
   credentials=/root/credentials-foo-bar
   ```

A credentials file must contain only one set of login information but it can be used by multiple mounts or `/etc/fstab` entries.

10.5.2Allow Access By Only One User

Replace the existing mount options with

```
defaults,noatime,nofail,user=username,password=password,uid=user,gid=group,file_mode=700,dir_mode=700
```

Omit `,file_mode=700,dir_mode=700` to allow other users read only access

For the pi user that would be:

```
defaults,noatime,nofail,user=username,password=password,uid=pi,gid=pi,gid=group,file_mode=700,dir_mode=700
```

or

```
defaults,noatime,nofail,user=username,password=password,uid=pi,gid=pi
```

Adjust the mount options if using a credentials file or guest access.

In the unlikely event that the server supports the CIFS Unix Extensions, add `,nounix` to the mount options.

10.5.3Allow Access Only By A Group Of Users

1. reate a new group:

   ```
   sudo groupadd groupname
   ```

   e.g.

   ```
   sudo groupadd cifswrite
   ```

2. Add users to the group:

   ```
   sudo usermod -a -G groupname username
   ```

   e.g.

   ```
   sudo usermod -a -G cifswrite pi
   ```

3. Replace the existing mount options with

   ```
   defaults,noatime,nofail,user=username,password=password,gid=cifswrite,file_mode=070,dir_mode=070
   ```

   e.g.

   ```
   defaults,noatime,nofail,user=fred,password=derf,gid=cifswrite,file_mode=070,dir_mode=070
   ```

   Change `file_mode` to 575 and `dir_mode` to 575 to allow read only access to users who are not members of the group.

   Adjust the mount options if using a credentials file or guest access.

In the unlikely event that the server supports the CIFS Unix Extensions, add `,nounix` to the mount options.

10.5.4Notes

`file_mode` and `dir_mode` are the permissions you want files and directories to have.

11 Additional Mount Options

The following options may be useful should you wish to go beyond the basics. Refer to the documentation for `mount` for more details and more options.

Including a mount option that does not apply to the file system in use may cause the mount to fail.

11.1 For All File Systems

ro: read only

rw: read/write. The default for many file systems.

auto: Automatically perform the mount at boot or when `sudo mount -a` is used.

noauto: Do not automatically perform the mount at boot or when `sudo mount -a` is used.

sync: Perform all I/O synchronously (i.e. without read and write caching) instead of asynchronously. May reduce performance and SD card lifetime.

user: Allow a normal user to mount and unmount this device if they mounted it. Not very useful with most automatic mounts as these are performed by root.

users: Allow a normal user to mount and to unmount this device even if they did not mount it.

remount: Remount a device. Not useful in `/etc/fstab`.

The following options apply only when systemd is in use. This is the case on Raspberry Pi OS and many modern linux. None are useful in a command line mount.

x-systemd.automount: Mount on first access via systemd.

x-systemd.idle-timeout=N: Time in seconds after which an idle device is unmounted. Should only be used with x-systemd.automount.[15]

x-systemd.device-timeout=N: Time in seconds to wait for a device to show up.

x-systemd.mount-timeout=N: Time in seconds to wait for the mount to complete.

11.2 For ext4 And Earlier

In the unlikely event you need file system specific mount options please refer to the manual page for `mount`.

[15] Otherwise the device will never be automatically remounted.

11.3 For FAT (All variants including exFAT)

uid=: Set the owner of all files and directories e.g. uid=pi

gid=: Set the owner of all files and directories e.g. gid=pi

umask=: Set which linux permissions are not present on all files and directories e.g. umask=000

fmask=: As umask but applies only to files.

dmask=: As umask but applies only to directories.

quiet: Allow `chmod` and `chown` to fail silently, i.e. without raising an error message. `chmod` and `chown` always fail on FAT file systems.

FAT has no concept of file ownership and a limited set of permissions (read only and hidden). Set them at mount time with the above options.

11.4 For NTFS

11.4.1 When using the In Kernel Driver

uid=: Set the owner of all files and directories e.g. uid=pi

gid=: Set the owner of all files and directories e.g. gid=pi

umask=: Set which linux permissions are not present on all files and directories e.g. umask=000. Granting write permissions will not make things writeable as the driver only supports reading.

11.4.2 When Using ntfs-3g

uid=: Set the owner of all files and directories e.g. uid=pi

gid=: Set the owner of all files and directories e.g. gid=pi

umask=: Set which linux permissions are not present on all files and directories e.g. umask=000

fmask=: As umask but applies only to files.

dmask=: As umask but applies only to directories.

silent: Allow `chmod` and `chown` to fail silently. On by default.

11.5 For CIFS

forceuid: Ignore any user ID provided by the server and use the one provided by the `uid` option instead.

forcegid: Ignore any group ID provided by the server and use the one provided by the `gid` option instead.

port=: Connect using the specified port. Only needed if the server is using a non standard port.

nounix: Disable the CIFS Unix Extensions. Most servers don’t support these anyway.

vers=: Force a specific CIFS/SMB protocol version. Usually not needed unless the server only supports v1.

11.6 For NFS

proto=: IP protocol to use. Either `tcp` or `udp`. Only needed to force a specific protocol or when autodetection fails.

port=: Connect using the specified port. Only needed if the server is using a non standard port.

nfsvers=: Force a specific version of the nfs protocol.

11.7 All Other File Systems

Please refer to the relevant documentation: `man mount` or man `mount.filesystem` (e.g. `man mount.cifs`).

Some file systems (e.g. zfs) cannot be mounted via the `mount` command or `/etc/fstab`.

12 Change Log

2021-02-08

Numerous tweaks

Add two new trouble shooting items: 9.3 and 9.4

Added section 11

2021-10-12

Addressed github issue #3

Updated 11.1 with additional systemd mount options

Changed heading for 11.3 to include exFAT

Adjusted page breaks

Corrected date in previous change log entry.