Filesystems

What is a filesystem?

- How your operating system stores files and directories on disk
- Often provides some useful features in addition to just deciding how data gets organized on your hard drive
  - Error correction/data recovery
  - RAID
  - Snapshotting
  - Compression
  - Encryption
  - Permissions
  - Deduplication
  - ...

Some common filesystems

- Linux
  - ZFS
  - EXT4/3/2
  - BTRFS
  - ...others
- Mac OS X
  - HFS/HFS+
- Windows
  - exFAT/NTFS
  - FAT16/32

Some common filesystems

- Linux
  - ZFS
  - EXT4/3/2
  - BTRFS
  - ...others
- Mac OS X
  - HFS/HFS+
- Windows
  - exFAT/NTFS
  - FAT16/32

As a sidenote...

- I won't cover the features of every filesystem in depth. Filesystems have more features and limitations than are covered here which may or may not be a consideration for you.

Considerations for Choosing a Filesystem

- How large are the files you'll store? (FAT32 only allows up to 4Gb files)
- How big will it be? (NTFS, and EXT4 support only up to ~1 Exabyte)
  - An exabyte is larger than most of us will need. If you run a cloud service, 1 Exabyte may be too small.
- How long are typical filenames? (NTFS only allows up to 255 characters)
  - When does this matter? Log filenames, version control filenames, etc
- Does it need to span multiple storage devices or provide redundancy?
  - We're mainly considering software RAID in this presentation
- Do you want to be able to roll back to a previous version of your data?
- Are you using a solid state drive?
- What operating systems need to read your data? (Windows will only read MS filesystems, while Linux and Mac can read many more)
- Which one is nicer to work with? (ZFS is designed to be user friendly and easy)

Which ones can you install Linux on?

- Depending how much magic you want to do, most of them
  - NTFS is POSIX compliant and Linux can technically be installed on it. Probably don't.
- Linux has wide support for a lot of different filesystems
- Linux doesn't ship with support for all filesystems out of the box
  - You can usually use them, but with a little extra work

Common Filesystem Features

Journaling

- Used to make filesystems more reliable in the case of a power failure by making corruption less likely - easier to come back online
- Keeps track of changes to data before it's written to disk
  - When you "save" to disk, things are not written right away because disks are so slow
- There are a few different ways filesystems can do journaling
  - Keep track of metadata (data about the data) to improve performance, but not reliability
  - Keep track of data to improve reliability
  - Keep track of both
- This can be a downside if you're on a solid state drive because it has more writes

Error Correction/Data Recovery

- Error Correction is detecting and fixing errors in the stored data. Data Recovery is recovering deleted or corrupted data
  - No, these are not the same but they are similar
- These are often referred to as “Filesystem Checks”, “Filesystem Repair”, or “Disk Checks”
- There are various ways of handling this, depending on your filesystem
- In Linux, this is often done using a utility called “fsck” which can repair many filesystems

RAID

- “Redundant Array of (Inexpensive|Independent) Disks”
- Allows you to use multiple hard drives
  - For better speed (worse for data recovery, usually)
  - For better data integrity (worse for speed, usually)
- All “big” servers use some form of RAID
- You generally will not use RAID on your own system, but you can
- There is software RAID (what we’ll look at here) and hardware RAID (which is somewhat faster, less flexible, and more expensive)

Snapshotting

- Ever unzipped a lot of files into a folder you didn't want to put them in? This lets you roll back to before that happened
- Allows you to take a "snapshot" of what your filesystem looked like at a particular time
- You can often browse the contents of your snapshots to recover data
- You can roll back your files to an earlier point in time

Compression

- Compress your data so it takes up less space so you can make more of your hard drive space
- Often (counterintuitively to most people) can actually improve system performance
  - Hard drives are *incredibly* slow from an engineering standpoint, so the less data you have to read or write, the better
  - This depends a little on processor speed. If your CPU is slow, compression is slow.
- If you run Windows 10, compression is likely enabled for you out of the box

Encryption

- Encrypt the contents of your storage so they can only be accessed with your key
  - If you attach an unencrypted drive to any system, it can read it assuming it has the drivers for that filesystem
- Makes data recovery much harder in many cases
  - Losing your encryption key generally means your data is gone
- Can be done on a filesystem level or on a file level depending on your preferences and system
  - Windows typically uses Bitlocker
  - Everyone can use TrueCrypt (which can also encrypt entire filesystems)
  - Linux has several technologies

Permissions

- If you use Linux, you're likely familiar with permissions
  - Restrict which system users can read, write, and run files
  - Keep track of who owns what, no matter where it is
- Believe it or not, Windows (on NTFS or exFAT) also supports permissions, but Windows doesn't make them as readily available for you to change
  - Linux can observe these with a little set up

The EXT Family

Basic Information about EXT

- Available on nearly every Linux distribution out of the box
  - Can be used with Windows using extra drivers, but with many limitations and missing features. This frequently is broken by windows updates.
- Current generation is EXT4 (EXT3 is still seen in the wild, and EXT2 is considered very obsolete)
- Considered to be a mature, stable filesystem
- Allows a filesystem to be upgraded to the new generations

EXT4 Features

- Journaling for better reliability
- Encryption
- Permissions (POSIX)
- Backwards compatible
- Repairable using the FSCK utility

EXT4 Limitations

- Maximum size: 1EiB
- Maximum file size (using defaults): 16Tb
- No compression supported
- Maximum number of files: 4 billion
- All characters except “/”, NUL”, “.” and “..” as filenames
- No Deduplication
- Does not support “Secure Deletion”
  - Overwriting files on deletion
- No Snapshotting

Basic Commands

- Check/Repair a filesystem: `fsck /dev/sdb1`
- Create a filesystem: `mkfs.ext4 /dev/sdb1`

BTRFS

BTRFS Basic Information

- Relatively new (introduced in the past few years)
- Not universally considered “stable”
  - Whether you consider it stable is up to you. For most purposes it is but for big data and high reliability it may not fit the bill.
- EXT4 can be “upgraded” to BTRFS
- BTRFS is a copy on write filesystem
  - Data is not written to the original file, it is written elsewhere and the pointers are updated to point to the new location

BTRFS Features

- Copy on write for better reliability (not journaling)
- Permissions (POSIX)
- Compression
- RAID
- Snapshotting
- Repairable with the FSCK utility
- Subvolumes

BTRFS Limitations

- Does not support encryption (it's planned)
- Does not support deduplication (it's in development)
- Maximum size: 16Eib
- Maximum filesize: 16Eib
- Any character in filenames except "/", "..", NUL, and a single "." as the filename

Basic BTRFS Commands

- Create a filesystem: `mkfs.btrfs /device/path`
- Create across multiple disks: `mkfs.btrfs /dev/1 /dev/2 …`
- Create a RAID10 array: `mkfs.btrfs -d raid10 -m raid10 /dev/1 /dev/2 …`
  - RAID has requirements for the number of disks in an array
- Convert from ext3/4: `btrfs-convert /device/path`
- Manage a subvolume: `btrfs subvolume (create|delete) /path/to/subvolume`
- Create a snapshot: `btrfs snapshot /path/to/subv1 /btrfs/subv1/<name>`
- Roll back to a snapshot:
  - Unmount the subvolume
  - Move the subvolume into the main filesystem

ZFS

ZFS Basic Information

- ZFS is super cool, so this doesn't do it justice
- Can be used under other filesystems with the same featureset
- Copy on write (not journaling)
- Designed to be easy to manage
- Designed for massive-data

ZFS Features

- Deduplication
- Compression
- Encryption
- Snapshotting
- RAID
- Permissions (POSIX)
- Can send and receive filesystems and filesystem changes
- Will not allow you to read corrupted data (either you can read the correct data, or nothing at all)

ZFS Limitations

- Maximum size: 256 zebibytes (yes. zebibytes). The amount of energy required to flip a bit in a filesystem that large would do bad things to the universe.
- Maximum filesize 16 EiB
- Filling a ZFS filesystem to over 80% full will make it extremely slow
- ZFS requires a lot of memory
- Not standard in most distributions and requires some extra magic to use
- Caching can be a problem without error correcting ram

ZFS Commands

- Create a storage pool: `zpool create <name> <disk(s)>`
- Create a RAID array: `zpool create <name> <raid> <disk(s)>`
  - RAID imposes disk requirements
- Create a subvolume: `zfs create <name>/<subvolume>`
  - You can format subvolumes to other filesystem types, including NTFS and FAT
- Create a snapshot: `zfs snapshot <name>/<subvolume>@<time>`
- Roll back to a snapshot: `zfs rollback <name>/<subvolume>@<time>`

Closing Thoughts

Which one is best?

- Whichever one is easiest for you to manage and whose limitations won't mess you up
- I use BTRFS on my laptop. My server runs ZFS for the storage pool and EXT4 for the operating system drive
- EXT4 is the default on most distros
    - Canonical (Ubuntu) is now working on making ZFS the default

There are more options

- ReiserFS, XFS, and others
- Installing Linux on a “non-Linux” filesystem is usually possible but frequently painful
- Using a filesystem with permissions on an external drive can be painful, although you can do it