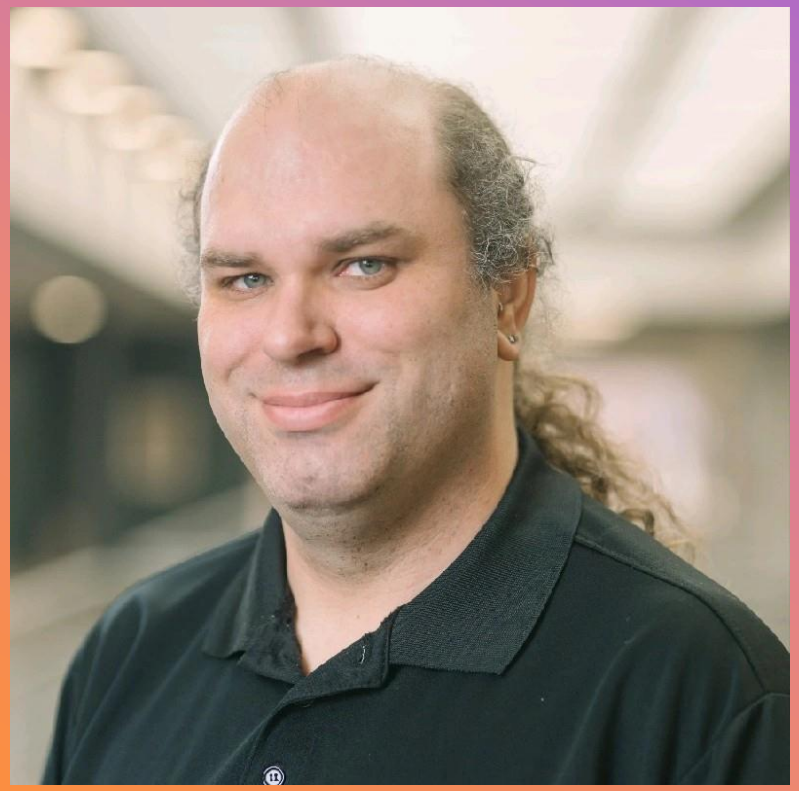
# DISKS AND FILESYSTEMS IN LINUX

ANDREW DENNER

CENTRAL IOWA LINUX USERS GROUP



# ABOUT ME

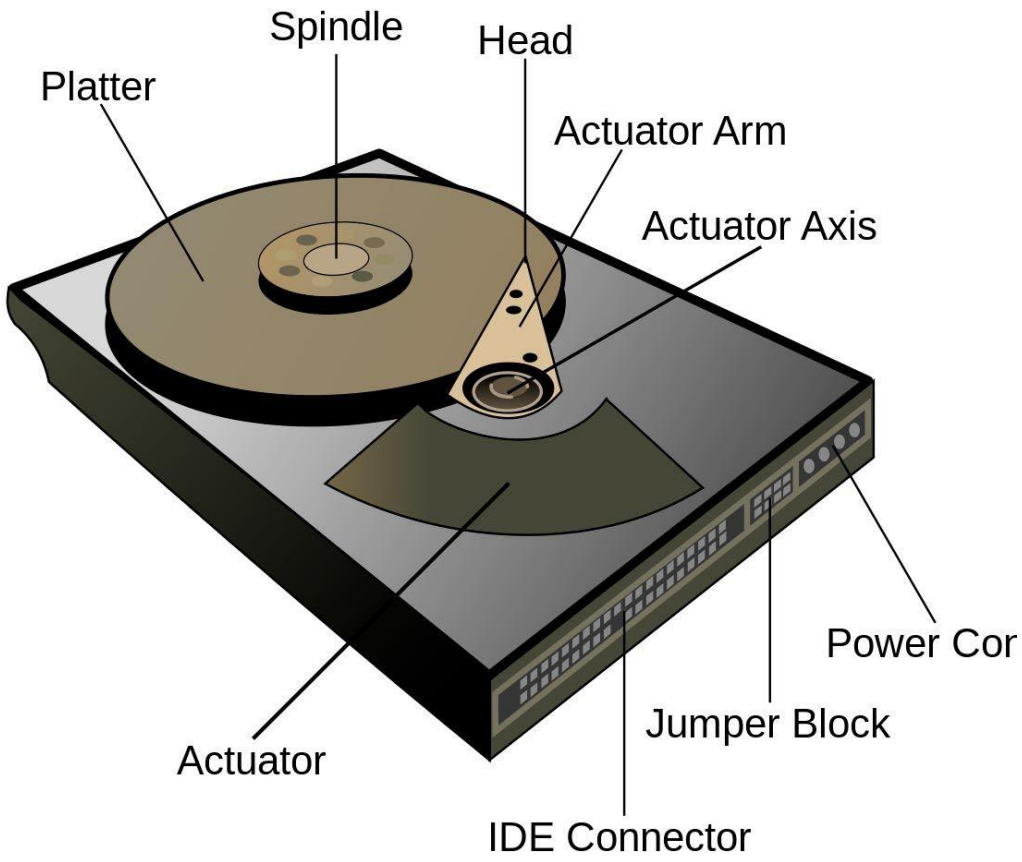
Andrew Denner

Linux User since 2003

@adenner

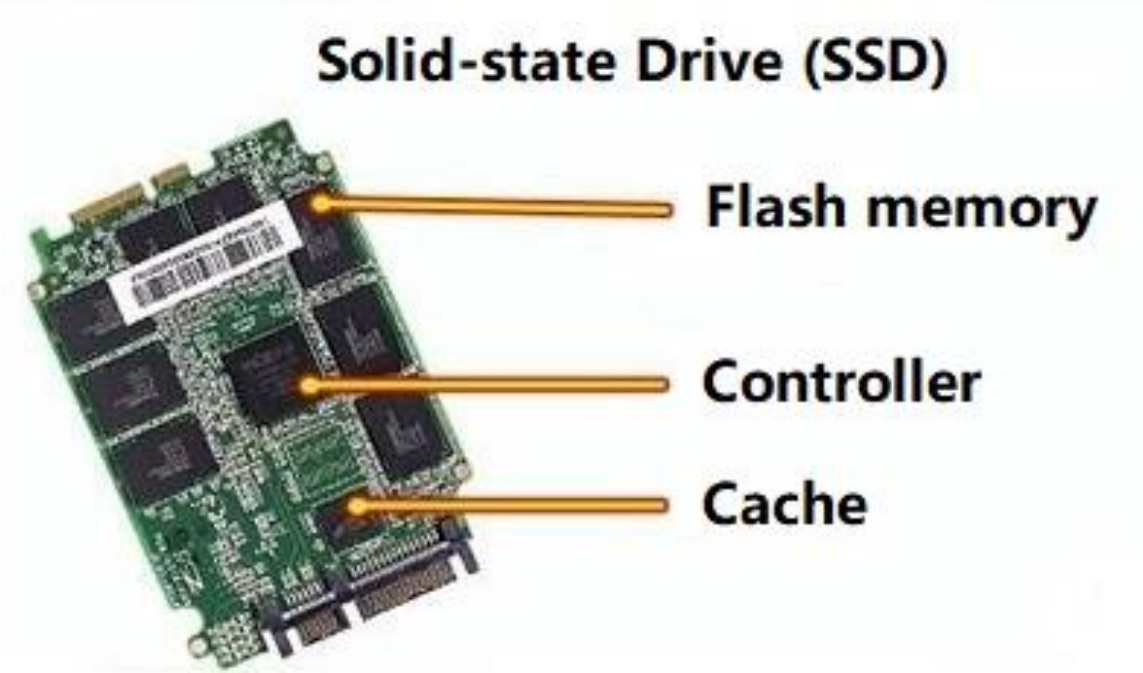
https://denner.co

# INTRO




# PARTS OF A HARD DISK

# SSD



# OLD TECH NEVER DIES, JUST MUTATES

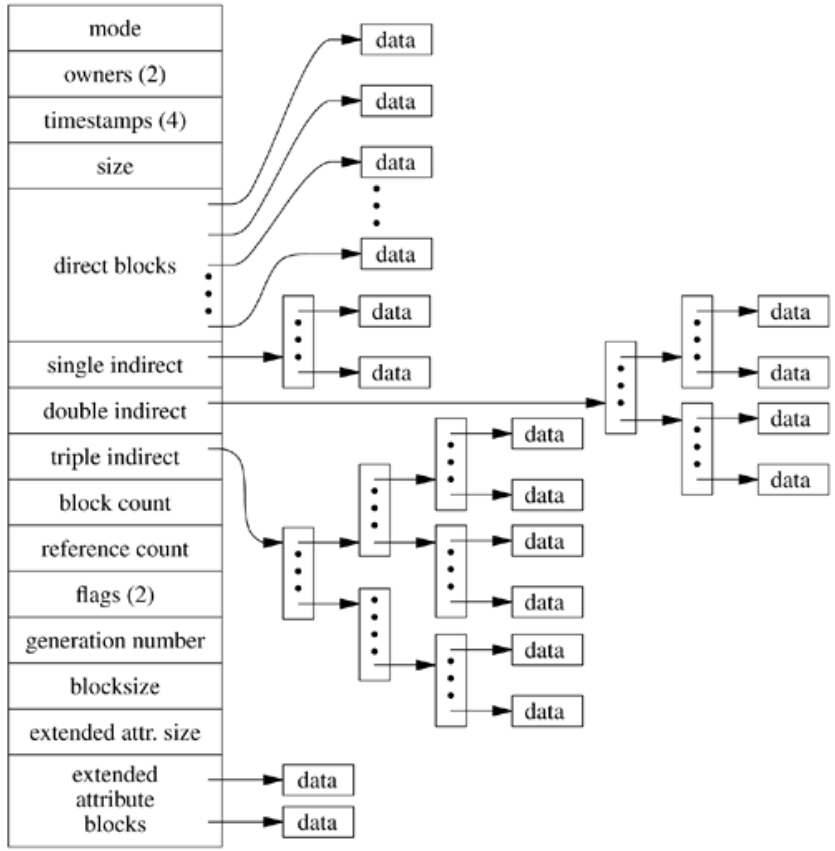
# INODES

- Fundamental building block of linux fs
- “index node”
- Metadata
  - Size
  - Permissions
  - Ownership
  - Timestamp
  - Address of disk blocks

ls –i

df –i

find / -inum[number]



# HOW ARE FILE NAMES CONNECTED?

## Directories [ edit ]

Each directory is a list of directory entries. Each directory entry associates one file name with one inode number, and consists of the inode number, the length of the file name, and the actual text of the file name. To find a file, the directory is searched front-to-back for the associated filename. For reasonable directory sizes, this is fine. But for very large directories this is inefficient, and ext3 offers a second way of storing directories (HTree) that is more efficient than just a list of filenames.

The root directory is always stored in inode number two, so that the file system code can find it at mount time. Subdirectories are implemented by storing the name of the subdirectory in the name field, and the inode number of the subdirectory in the inode field. Hard links are implemented by storing the same inode number with more than one file name. Accessing the file by either name results in the same inode number, and therefore the same data.

The special directories "." (current directory) and ".." (parent directory) are implemented by storing the names "." and ".." in the directory, and the inode number of the current and parent directories in the inode field. The only special treatment these two entries receive is that they are automatically created when any new directory is made, and they cannot be deleted.

# JOURNALS

Keeps track of changes not yet committed to file system main parts

Examples:

Ext3

Ext4

Tools fsck:

```
e2fsck -p /dev/sda1
```

# TOOLS AND UTILITIES

Fdisk

Lsblk

Df

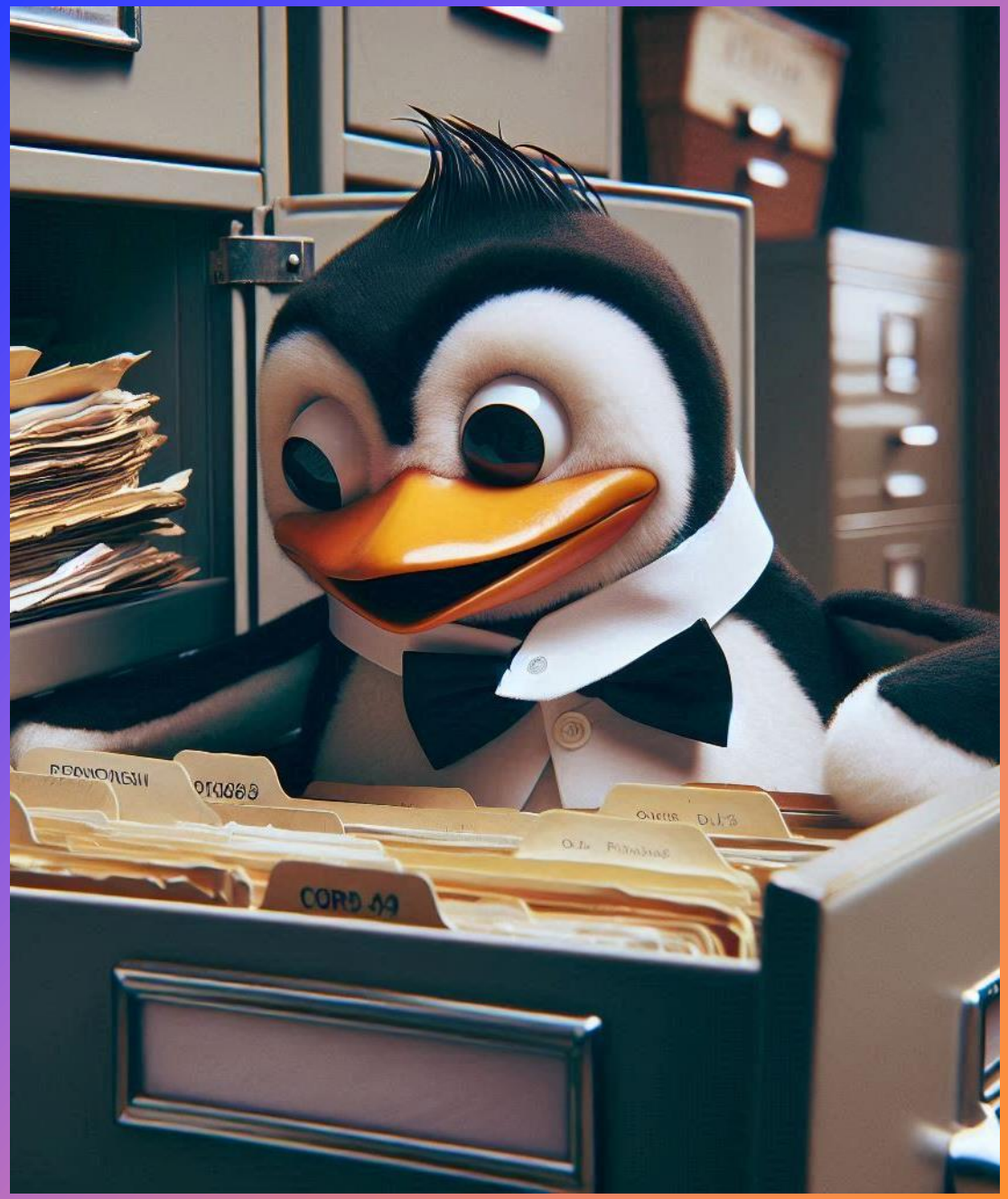
Du

Smartctl

# FILE SYSTEMS

# OLD FILESYSTEMS

- Ext2
  - Default from long ago
  - No journaling
- Ext3
  - Journaling and somewhat backwords compatible with ext2
  - Performance Issues vs Ext4
- JFS
  - IBM Journaled filesystem
  - Very stable but lacks features of newer fs

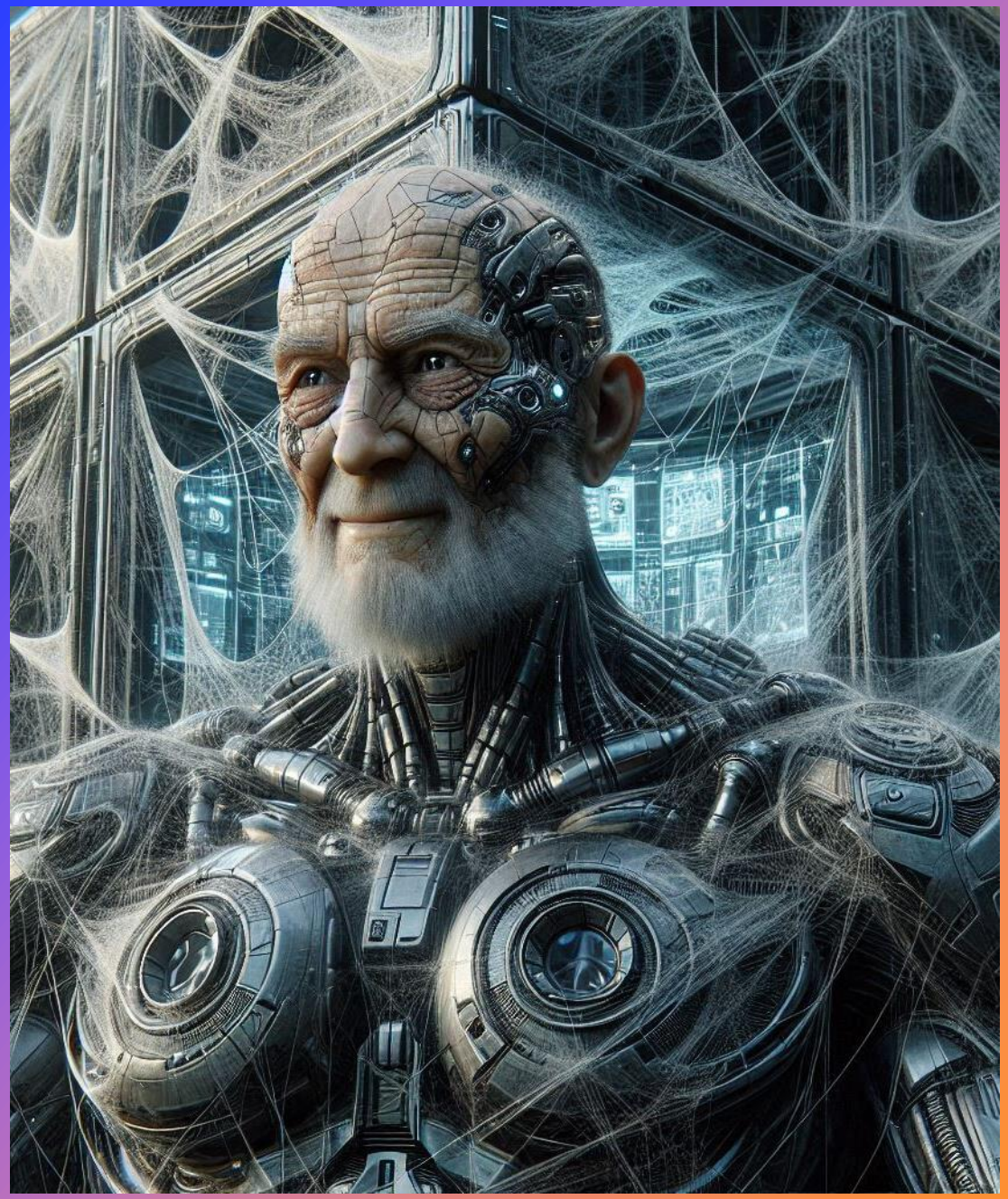



# DEAD FILESYSTEMS

- ReiserFS
  - Journaling
  - B+ tree data structures
  - File allocation by bitmap
  - Volume size of 16 TiB max file size of 1EiB
  - Obsolete and marked for removal 2025
  - Not maintained and has Y2k38

# OLD FILE SYSTEMS NOT ONLY LINUX

- FAT32
  - Most compatible with other os
  - Simple
  - Redundant -2 copies of table
  - Unorganized- Gets first disk space
  - Limited file size of 4 gb



# MODERN FS — EXT4

- Journaling
- Large fs support
  - Volume size to 1EiB
  - Files 16TiB to 256TiB
  - Extents
  - Delayed Allocation
  - Directory indexing
  - Backward compatible with ext2 and ext3
  - Persistent preallocation
  - Time ok till 2446

# XFS

1993 SGI IRIX

64 bit fs

Journaling

Scales to 8 exbibytes

Divides disk to allocation groups

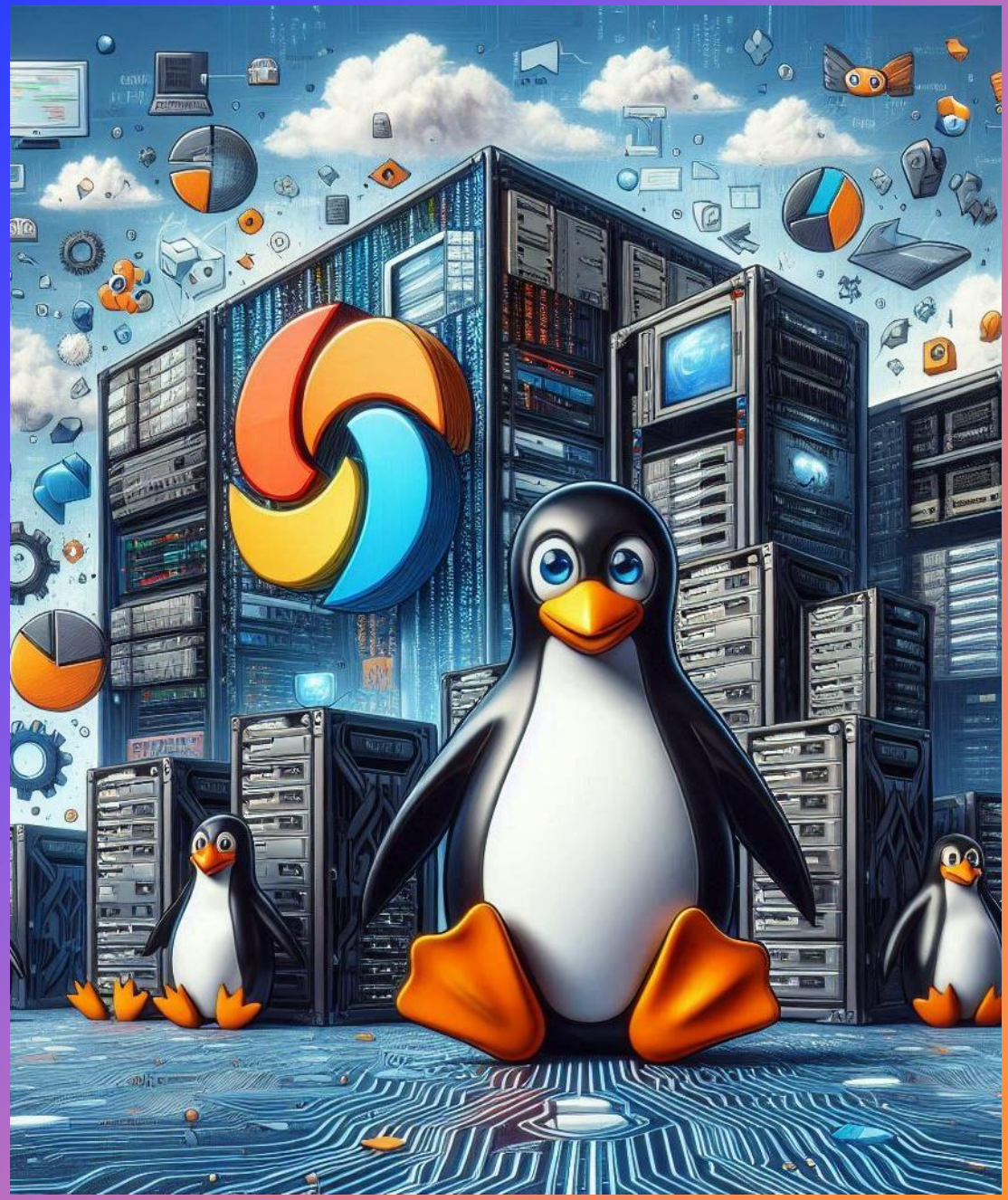
B+ trees

Delayed allocatoion

Online Defragmentation

Extended attributes

Default With RHEL systems



# BTRFS

COW

Snapshots and subvolumes

Checksums

Compression

Integrated multi-device support-RAID

Dynamic inode allocation not fixed

Defagmentation while mounted and use

Large FS and files up to 16 Eib

Advanced Features

# ZFS

Zettabyte file system from Sun

Data integrity---Checksum and COW

Pooled storage

Snapshots and clones

Dynamic striping

Scales 256 trillion yobibytes and 16 exbibytes per file

RAID support

Compression and Dedup on fly

Transparent Encryption

# NON LINUX-EXFAT

2006 for usb and sd cards

Large file support

128 PB size recommend 512 TB

2796202 files per directory

Cross compatible

Simple low power

Default fs for SDXC cards

# NTFS

New Technology File System win NT 3.1

- Journaling
- File level permissions, encryption including ACL and bitlocker
- Large volume of 8 petabytes
- Advanced features Hardlinks, sparse files disk quotas, reparse points, simlink
- Compression
- High performance
- Reliable, self healing