Linux (General)

General topics relating to Linux and Unix operating systems

- Basic Commands Overview
- Filesystem Hierarchy
- Linux File Systems
    - Ext4: the reliable all-rounder
    - Btrfs: modern and feature-packed
    - XFS: the workhorse
    - ZFS: the file system to end all file systems

Basic Commands Overview

Command	Description	Package
`base64`	base64 encode/decode data and print to standard output	`coreutils`
`basename`	strip directory and suffix from filenames	`coreutils`
`cat`	concatenate — combine the output of files and print to standard output	`coreutils`
`cd`	Change Directory — navigate directories on the filesystem	built-in
`chmod`	change file mode bits (permissions) for files and directories	`coreutils`
`chown`	change file owner and group	`coreutils`
`clear`	clear the terminal content	`clear`
`cp`	copy files	`coreutils`
`csplit`	split a file into sections determined by context lines	`coreutils`
`dd`	convert and copy a file	`coreutils`
`df`	Disk Free — report overall file system space usage	`coreutils`
`diff`	show differences between two files line by line	`diffutils`
`dirname`	print the directory path if the passed file	`coreutils`
`du`	Disk Usage — show file system usage for a given directory	`coreutils`
`echo`	display a line of text	built-in
`find`	search for files in a given directory	`findutils`
`grep`	search for text	`grep`
`head`	display the first X lines of a file (default 10)	`coreutils`
`history`	print the history of issued commands	built-in
`jobs`	display currently running jobs	built-in
`kill`	terminate a process	`util-linux`

Command	Description	Package
`ls`	list directory contents	`coreutils`
`man`	Manual — display the documentation of a given command	`man`
`mkdir`	create a directory	`coreutils`
`mv`	move (and rename) files	`coreutils`
`ping`	check if a server is reachable	`iputils`
`printenv`	print information about currently set environment variables	`coreutils`
`printf`	format and print data	`coreutils`
`pwd`	Print Working Directory — returns the current working directory	built-in
`rm`	remove files and directories (caution: no undo!)	`coreutils`
`scp`	OpenSSH secure file copy	`openssh`
`sha256sum`	compute and check SHA256 message digest (variants for SHA1, SHA224, SHA384 and SHA512 exist)	`coreutils`
`sort`	sort lines of text files	`coreutils`
`sudo`	Super User Do — execute commands as root	`sudo`
`tail`	display the last X lines of a file (default 10)	`coreutils`
`tar`	archiving utility, compression optional	`tar`
`top`	display Linux processes	`procps-ng`
`touch`	create an empty file	`coreutils`
`uname`	Unix Name — print system information	`coreutils`
`unzip`	list, test and extract compressed files in a ZIP archive	`unzip`
`useradd`	create a new user or update default new user information	`shadow`
`userdel`	delete a user account and related files	`shadow`
`wget`	download files from the internet	`wget`
`zip`	package and compress (archive) files	`zip`

Filesystem Hierarchy

The Filesystem Hierarchy Standard (FHS) is a set of guidelines that define the directory structure of a Linux operating system. This standard specifies the names and locations of directories, files, and other file system objects that are essential for the proper functioning of the Linux operating system.

Directory	Description
`/bin`	binary files of basic commands
`/boot`	static files of the bootloader
`/dev`	device files
`/etc`	host-specific system configuration
`/lib`	basic dynamic libraries and kernel modules
`/media`	mount point for removable media
`/mnt`	for temporarily mounted file system
`/opt`	additional application programs
`/run`	relevant data for running processes
`/sbin`	essential binary files of the system
`/srv`	data for services
`/tmp`	temporary files
`/usr`	secondary hierarchy
`/var`	variable data

Root Directory (`/`)

The root directory is the highest level of the file system hierarchy. It contains all other directories and files in the system. All directories and files in the system are located either directly or indirectly under the root directory.

`/bin` Directory

> **NOTE:** On most modern Linux distributions, `/bin` is actually a symbolic link pointing to `/usr/bin`.

The `/bin` directory contains essential binary files that are required for the system to function properly. These files are used by all users of the system, and they include basic system commands such as `ls`, `cp`, and `mv`. The `/bin` directory must not contain any subdirectories. Each installed package places its main executables here.

`/boot` Directory

The `/boot` directory contains the files needed by the bootloader for the Linux kernel, including the kernel itself, initrd (**init**ial **R**AM **d**isk), and boot loader configuration files. The bootloader is the program that loads the operating system when the computer starts up. The files in `/boot` are essential for the bootloader to function properly.

`/dev` Directory

The `/dev` directory contains device files that represent hardware devices connected to the system. These files allow applications and users to interact with the hardware devices such as keyboards, mice, and disk drives using standard input/output operations.

There's different types of device files:

Type	Description
Character device	Communication with a character device is based on the exchange of characters, such as individual bytes
Block device	Communication with a block device is based on the exchange of entire blocks of data
Pseudo device	Devices that are not corresponding to any real hardware actually present on the system

Each device file in the `/dev` directory represents a specific hardware device. These files are created dynamically and disappear when the device is disconnected.

Some examples of device files:

File	Description

`/dev/dm-X`	Logical volumes created using LVM (**d**evice **m**anager). • X = index of device manager volume
`/dev/hdX`	IDE hard drives or other storage devices. Common in older systems. • X = lower-case letter indicating order of discovery
`/dev/hdXY`	Partitions on IDE storage device. • X = disk • Y = partition on that disk
`/dev/jsX`	Device file for a joystick/gamepad. • X = index of a joystick connected to the system
`/dev/mdX`	Software RAID devices (**m**eta **d**isk). • X = meta disk group index
`/dev/mmcblkX`	MMC/SD cards and eMMC storage devices. • X = index of an MMC storage device
`/dev/nvmeXnY`	NVMe solid-state drives. • X = controller index • Y = device index on that controller
`/dev/nvmeXnYpZ`	Partitions on NVMe storage device. • X = controller index • Y = device index on that controller • Z = partition on that device
`/dev/null`	Pseudo-device that always outputs EOF (end of file). Output redirected to it is immediately discarded.
`/dev/random`	Pseudo-device that generates random data (blocking).
`/dev/sdX`	SCSI or SATA hard drives and other storage devices. Common in modern systems. • X = lower-case letter indicating order of discovery.
`/dev/sdXY`	Partitions on SCSI or SATA storage device. • X = disk • Y = partition on that disk

`/dev/srX`	CD-ROM or DVD-ROM drives. • X = index of an optical drive connected to the system
`/dev/ttyX`	Device file for the terminal. • X = index of a terminal
`/dev/urandom`	Pseudo-device that generates random data (non-blocking)
`/dev/vdX`	Virtual storage device. • X = lower-case letter indicating order of discovery
`/dev/vdXY`	Partitions on a virtual storage device. • X = disk • Y = partition on that disk
`/dev/videoX`	Device file for a webcam. • X = index of a webcam connected to the system

`/etc` Directory

The `/etc` (**e**ditable **t**ext **c**onfiguration) directory contains system configuration files. These files are used to configure the system and its applications. Examples of files in this directory include `passwd` and `group` files, which contain user and group information, and `fstab` file, which contains information about file systems that are mounted at boot time. The `/etc` directory is one of the most important directories in the system, as it contains configuration files that enable system administrators to customize the system to their needs.

Some examples of what might be of particular interest:

Directory/File	Description
`/etc/opt/`	Configuration files for software installed to `/opt`.
`/etc/security/`	Configuration files for PAM (**P**luggable **A**uthentication **M**odules).
`/etc/skel/`	Configuration files that are copied to a newly created user's `/home` directory.
`/etc/X11/`	Configuration files for the X Window System which provides a graphical user interface for Linux.

Directory/File	Description
`/etc/crontab`	Configuration file uniquely formatted to automate system tasks on a set schedule.
`/etc/fstab`	Contains information about file systems and partitions automatically mounted at system startup.
`/etc/group`	Contains information about user groups, including group names and group IDs.
`/etc/hosts`	Configuration file that maps hostnames to IP addresses.
`/etc/logrotate.conf`	Configuration file for the `logrotate` utility, managing system log files.
`/etc/passwd`	Stores user account information such as usernames, user IDs, home directories, and login shells.
`/etc/profile`	Contains system-wide environment variables and other startup scripts.
`/etc/resolv.conf`	DNS resolver file which specifies how the system leverages DNS to resolve hostnames.
`/etc/shadow`	Contains encrypted user passwords and other password-related information.
`/etc/ssh/sshd_config`	Configuration file for the SSH server, which allows secure remote access to the system.
`/etc/sudoers`	Contains rules for allowing or denying `sudo` privileges to users and groups.

`/home` Directory

The `/home` directory contains the personal files for all users in the system. Each user has a subdirectory of their name in the `/home` directory that contains their documents, pictures, music, and individual application settings. Each user's home directory is private, and other users cannot access the files in that directory without the user's permission.

> **NOTE:** A user's personal home directory can be shortened to `~`.

> **NOTE:** Directories and files starting with a period `.` are considered hidden.

Some examples for directories/files in a user's home directory could include:

Directory/File	Description

`~/.config/`	Contains configuration files for applications that are specific to a user.
`~/.gnupg/`	Contains user-specific configuration files for GnuPG (GNU Privacy Guard).
`~/.local/`	Contains user-specific data files for applications that are installed system-wide.
`~/.mozilla/`	Contains user-specific configurations and settings for Firefox.
`~/.ssh/`	Contains user-specific configuration files and private/public keys for SSH (Secure Shell) connections.
`~/.bashrc`, `~/.zshrc`	Contains user-specific configurations for the user's terminal emulator.

These hidden directories in a user's home directory are important for ensuring that user-specific data and configuration files are stored in a separate location from system-wide files. This separation enables users to customize their system to their needs without affecting other users or the core system.

`/lib` Directory

> **NOTE:** On most modern Linux distributions, `/lib` is actually a symbolic link pointing to `/usr/lib`.

The `/lib` directory (or `/lib64` for 64-bit systems) is a crucial component of the operating system. It contains shared libraries that are required by programs in `/bin` and `/sbin` directories, as well as other applications. These libraries provide functionality for many basic system operations, such as networking, file handling, and encryption. Without these libraries, the system would not be able to perform many of its fundamental functions. It is important to ensure that the files in `/lib` are always up-to-date and compatible with the system's other components to guarantee smooth operation.

In addition to the shared libraries, the `/lib` directory contains other important files, such as kernel modules (`/lib/modules/`) and firmware files (`/lib/firmware/`). Kernel modules are small programs that can be loaded and unloaded dynamically into the kernel without having to reboot the system. These modules provide additional functionality to the system, such as support for new hardware or file systems. Firmware files, on the other hand, contain low-level software that is used to control hardware devices, such as network adapters (especially WiFi) or graphics cards.

Directories/Files	Description

`/lib/firmware/`	Contains firmware files needed by devices to function properly, e.g. WiFi adapters and graphics cards.
`/lib/security/`	Contains authentication modules for PAM (**P**luggable **A**uthentication **M**odules).
`/lib/systemd/`	Contains files for the systemd system and service manager. Also contains systemd unit files.

`/media` Directory

The `/media` directory contains mount points for removable media devices such as USB drives and CD/DVDs. When a removable device is mounted, its contents appear in a subdirectory under the `/media` directory. The `/media` directory is used to manage removable media devices, and it allows users to access the files on those devices as if they were part of the file system.

`/mnt` Directory

The `/mnt` directory is used to temporarily mount file systems. This directory is typically used by system administrators when they need to mount a file system for a short period of time. The `/mnt` directory is used to access file systems that are not part of the main file system, such as network file systems or file systems on removable media devices.

`/opt` Directory

The `/opt` directory is used for optional software packages that are not part of the main operating system. Examples of software that might be installed in the `/opt` directory include proprietary software packages or third-party software that is not included with the distribution's main software repositories.

Examples of software you might see in this directory:

Directory/File	Description
`/opt/1Password/`	Password manager from Agile Bits
`/opt/discord/`	Discord VoiP client
`/opt/google/`	Google Chrome web browser
`/opt/quake3/`	Quare 3 Arena from id Software

Directory/File	Description
`/opt/spotify/`	Spotify music streaming client
`/opt/visual-studio-code/`	Visual Studio Code editor from Microsoft

`/proc` Directory

The `/proc` directory is a virtual directory that contains information about running processes and system resources. This directory provides a way for processes to communicate with the kernel and allows system administrators to monitor system performance. The files in the `/proc` directory are not actual files, but rather a virtual representation of system resources. The `/proc` directory is used by system administrators to monitor system performance and diagnose system problems.

Directory/File	Description
`/proc/cpuinfo`	Contains information about the CPU, e.g. model name, number of cores, and clock speed.
`/proc/meminfo`	Contains memory usage information including swap storage use.
`/proc/modules`	Contains a list of all the modules being used by the kernel.
`/proc/swaps`	Contains information only about swap storage.
`/proc/sys`	Contains files that are used to manage kernel parameters.
`/proc/version`	Contains Linux version information.

`/root` Directory

The `/root` directory is the home directory for the root user. This directory contains the root user's personal files and settings. The root user is the most powerful user on the system, and has access to all files and directories on the system.

`/run` Directory

The `/run` directory is a unique and important directory in the Linux operating system that contains runtime data that is used by the system and applications. This data is volatile as it is mounted as a `tmpfs` filesystem that is only stored in system memory and its contents are lost when the system is rebooted.

The data stored in the `/run` directory is updated frequently and automatically by the system and applications. The directory is also used as a mount point for systemd, a system and service manager that is used in most modern Linux distributions. Systemd is responsible for managing system services, and it also uses the `/run` directory to store runtime data for these services.

The `/run` directory is also used for user mounted storage mediums like CDs, DVDs, USB drives and network shares.

`/sbin` Directory

> **NOTE:** On most modern Linux distributions, `/sbin` is actually a symbolic link pointing to `/usr/bin`.

The `/sbin` directory contains essential system binaries that are used for system administration tasks, such as configuring the network or managing user accounts. These files are typically used by system administrators and are not intended for everyday use.

`/srv` Directory

The `/srv` directory is used to store data that is served by the system, and it is typically used by web servers, FTP servers, and other server applications.

Other directories that are used for different purposes include:

Directory/File	Description
`/srv/ftp/`	Contains files that an FTP server uses to store all of the files that can be accessed by the FTP server.
`/srv/www/`, `/srv/http/`	Contains files that are served by a web server (sub-directory is distribution specific).

`/sys` Directory

The `/sys` directory is a fundamental component of the Linux filesystem hierarchy and an essential tool for hardware management and configuration. This directory is structured in such a way that each hardware device is represented by a directory tree and a set of files, providing users with detailed information about the device's properties and allowing them to adjust, among other things, power management settings, device parameters, and monitor the device's status. The files

in the `/sys` directory are not actual files, but rather a virtual representation of hardware devices. They are re-created upon every boot of the system and mounted as a virtual `sysfs` filesystem.

`/tmp` Directory

The `/tmp` directory is used for temporary files. This directory is typically used by applications to store non-critical files and it is typically cleaned out on a regular basis to prevent the accumulation of unnecessary files. It is writable to any user and is ideal for using it as a scratchpad type of directory. The `/tmp` directory is mounted as a `tmpfs` type of filesystem, which means its contents are stored in the system's memory for fast access. However, this also means that it is generally not recommended putting large files there as to not exhaust the system's memory with large amoounts of data.

`/usr` Directory

The `/usr` (**U**nix **S**ystem **R**esources) directory is used to store programs, libraries, documentation and data that are not part of the core operating system, but that are still important for the system to function. Examples of programs that might be installed in the `/usr` directory include text editors, email clients, and games.

Some other examples of contents of the `/usr` directory:

Directory/File	Description
`/usr/bin/`	Contains executable binaries of applications
`/usr/bin/gcc/`	Binaries for compiling all sorts of source code of different programming languages
`/usr/include/`	Source code header files for including functionality of one application into another during development
`/usr/lib/`	Shared system libraries, e.g. `libc.so` (C library), `libssl.so` (SSL library)
`/usr/lib/systemd/`	Systemd service unit files
`/usr/local/`	Typically used to install applications compiled from source, otherwise goes largely unused
`/usr/share/`	Architecture-independent data files, e.g. game assets, icons, cursors, desktop themes, wallpapers, etc.
`/usr/share/doc/`	System documentation
`/usr/share/locale/`	Translations for applications

Directory/File	Description
`/usr/share/man/`	Application manual pages
`/usr/src/`	Linux kernel source code files

`/var` Directory

The `/var` directory contains variable data files. These files are typically files that change frequently, such as log files, locally delivered mail and spooler files for printer job queues.

Some other examples of contents of the `/var` directory:

Directory/File	Description
`/var/cache/`	Frequently accessed data, e.g. browser cache and downloaded software packages
`/var/lib/`	Varying application data used by system libraries (e.g. `/var/lib/libvirt/images/` for `libvirt` virtual machine disk images)
`/var/log/`	Log files of all sorts of different applications
`/var/mail/`	Locally delivered mail messages
`/var/mysql/`	MySQL database files
`/var/spool/`	Printer job queue

Linux File Systems

Linux supports a wide range of file systems, from older ones like **Ext2** to more modern, feature-rich options like **Btrfs** and **ZFS**. Each file system offers a unique set of features, including support for journaling (protecting file system integrity), compression, encryption, and snapshots. The choice of a file system can affect system performance, reliability, scalability, and ease of management. Selecting the right file system is key to ensuring your system operates efficiently and reliably.

Linux's open-source nature allows for flexibility in choosing the right file system, whether you're looking for performance, data integrity, fault tolerance, or ease of use. This diversity provides many options to tailor the system's storage to specific requirements.

Ext4: the reliable all-rounder

Ext4 (Fourth Extended File System) is a high-performance, journaling file system widely used in Linux environments. It is an evolution of its predecessors, **Ext3** and **Ext2**. Ext4 is the default file system for many Linux distributions, as it is battle-tested and a good choice for general purpose desktop computing needs.

Key features of Ext4 include:

1. **Journaling**: Like Ext3, Ext4 is a journaling file system, meaning it keeps a log (or journal) of changes to the file system before committing them to disk. This helps prevent data corruption in the event of a system crash or power failure, as the journal can be used to roll back or replay incomplete operations. Additionally, Ext4 uses checksums in the journal to improve reliability. This feature has a side benefit: it can safely avoid a disk I/O wait during journaling, improving performance slightly.
2. **Larger File and Volume Support**: Ext4 supports file sizes between 16 - 256 TiB and volumes up to 1 EiB, making it suitable for modern storage needs. This was a significant improvement over Ext3, which, depending on block size, was limited to 2 - 32 TiB volumes and 16 GiB - 2 TiB file sizes.
3. **Extents**: Extents replace the traditional block mapping scheme used by ext2 and ext3. An extent is a range of contiguous physical blocks, improving large-file performance and reducing fragmentation.
4. **Backward Compatibility**: Ext4 is backward compatible with Ext3, meaning you can mount an Ext3 file system as Ext4 and take advantage of the newer features without needing to reformat the partition.
5. **Delayed Allocation**: Ext4 provides better performance compared to Ext3, thanks to delayed allocation, which improves write performance and multi-block allocation, optimizing space usage. This further helps to reduce fragmentation.
6. **Extended Attributes**: Ext4 supports extended attributes, allowing additional metadata to be attached to files (e.g. security labels, file system flags, user tags). This is useful for extending file properties with arbitrary application metadata to offer advanced file management.
7. **Online Defragmentation**: Unlike Ext3, Ext4 provides the ability to perform online defragmentation, which allows the file system to be defragmented while the system is running, reducing the impact on performance.

Ext4 is a solid choice for every-day computing needs. It's simple, robust and well tested in the field. However, its developers have stated that Ext4 is just a stop-gap until more modern file systems like Btrfs mature and reach the same performance and robustness levels as Ext4.

Btrfs: modern and feature-packed

Btrfs (B-tree File System) is a modern, copy-on-write (COW) file system for Linux designed to address the limitations of older file systems like Ext4. It provides advanced features such as snapshots, data integrity verification, built-in volume management, and more. Btrfs aims to combine the functionalities of traditional file systems and logical volume managers, making it a versatile and powerful tool for managing storage.

Key features of Btrfs include:

1. **Copy-on-Write (COW)**: Btrfs uses copy-on-write (COW) for data and metadata, meaning that when data is modified, the changes are written to new locations on disk instead of overwriting the original data. This approach ensures that the previous version of the data is preserved, which is crucial for features like snapshots and data integrity. It also makes copies of data on the same volume instantaneous, because both refer to the same block of data on disk. COW provides benefits such as *atomic writes*, where operations are either fully completed or not done at all, reducing the risk of data corruption.
2. **Snapshots**: Snapshots are read-only or read-write copies of the file system at a specific point in time. Snapshots are efficient and fast because they share common data blocks until changes to the data on the disk are made. Snapshots are ideal for backup, system rollback, and testing purposes. They allow you to capture the state of the file system and revert back to it later if necessary. With certain system configurations it is even possible to boot into a snapshot.
3. **Data Integrity and Checksumming**: Btrfs provides checksumming for both data and metadata. Every block of data and metadata is verified with a checksum when read or written. If corruption is detected (e.g. due to hardware failure), Btrfs can attempt to recover the data using its checksums. This feature significantly improves the file system's reliability, especially in environments where data integrity is critical.
4. **Built-in Volume Management**: Btrfs includes volume management features natively, which means it can handle multiple physical devices and manage them as a single logical volume. This feature eliminates the need for external volume managers like LVM (Logical Volume Manager). It supports RAID-like configurations (RAID 0, RAID 1, RAID 10) natively, allowing users to create storage pools with redundancy and striping.
5. **Compression and Deduplication**: Btrfs natively supports transparent compression, allowing data to be stored in a compressed format to save space. Several algorithms (such as LZO and zstd) are available for use. Deduplication (removal of duplicate data) can also be performed, making it ideal for environments with large amounts of redundant

data where storage space is precious.

6. **Dynamic Sizing and Subvolumes**: Btrfs allows the dynamic resizing of file systems, making it easy to increase or decrease the size of volumes as needed, without requiring reformatting or complex migrations. Subvolumes are logical divisions of a file system that can be treated like separate file systems. Each subvolume can have its own set of snapshots, making them useful for organizing data, creating backups, or managing different applications in isolation. Subvolumes can also be used to install multiple different operating systems on the same file system without getting into the way of one another.
7. **Online Defragmentation**: Btrfs supports online defragmentation, meaning that it can reorganize and defragment the file system while the system is running, which can help improve performance over time, especially on systems with high churn of small files.
8. **Self-healing**: In combination with its checksumming capabilities, Btrfs is able to self-heal data when used with RAID configurations. For example, in a RAID 1 setup, if one disk has corrupted data, Btrfs can detect the issue and restore the correct data from the mirrored copy.

The rich feature set of Btrfs makes it a modern and versatile choice for data storage on Linux. Built-in volume management at the file system level eliminates having to think about partition layouts and sizes, while the RAID capabilities make an underlying software RAID layer redundant. Snapshots allow for quick rollbacks to a previous state of the system without wasting precious space. Transparent compression and deduplication maximize efficient use of storage space. When used effectively, Btrfs is an excellent and future-proof choice.

However, because Btrfs is still a relatively new file system, some features are still not considered ready for daily use, e.g. integrated RAID 5/6 support currently being considered unstable. It also often ranks worse than other file systems in areas where high throughput is important, such as database applications or as backing storage for virtual machines. The complex feature set may also pose a high learning curve for people unfamiliar with its concepts.

XFS: the workhorse

XFS is a high-performance journaling file system created by Silicon Graphics, Inc. for their IRIX workstations. XFS is particularly proficient at parallel I/O due to its allocation group based design. This enables extreme scalability of I/O threads, filesystem bandwidth, file and filesystem size when spanning multiple storage devices.

Key features of XFS include:

1. **Journaling**: Like other journaling file systems, XFS keeps a log (journal) of file system changes before they are written to disk. This helps ensure data integrity in the event of a system crash or power failure, as the file system can use the journal to recover incomplete operations.
2. **Scalability**: XFS is designed to handle large volumes and large files efficiently. It supports file systems as large as 16 EiB and individual file sizes up to 8 EiB, making it suitable for modern data centers and high-performance applications that require managing vast amounts of data.
3. **Parallel I/O**: XFS is optimized for parallel I/O operations, which improves performance for applications with heavy read/write demands like databases or large media files.
4. **Extents-based Allocation**: XFS uses extents (contiguous blocks of storage) to manage files, which reduces fragmentation and improves throughput when working with large files.
5. **Online Defragmentation**: XFS supports online defragmentation, allowing for defragmentation of the file system without having to unmount it. This is particularly useful for maintaining performance on systems with minimal downtime.
6. **Dynamic Allocation**: XFS supports dynamic inode allocation, meaning that inodes (the data structures used to store metadata about files) can be created as needed, improving the file system’s efficiency when dealing with a large amount of files.
7. **Data Integrity**: In addition to journaling, XFS also provides checksumming of metadata, which helps ensure data integrity and detect corruption. This is important for environments where data reliability is critical.
8. **Efficient Space Management**: XFS is known for its efficient space management. It uses techniques like delayed allocation and aggressive pre-allocation to minimize disk fragmentation and optimize disk space usage

XFS is the file system of choice when high throughput and performance are important. It's the file system of choice for storage arrays found in servers and NAS systems. It is a good choice for the backing storage for virtual machine environments and high availability database applications. While it lacks some of the advanced features found in other file systems (like snapshots or compression), it excels in areas such as scalability, performance, and data integrity. XFS is well-suited for applications requiring fast, reliable access to large datasets.

Areas that XFS does not perform well in is handling large amounts of small files, as it stays true to its data center and workstation roots. Another downside of XFS is that while it's easy to grow the file system, it is not (yet) possible to shrink it. XFS recovery, while reliable, can be more complex than other file systems in cases of severe corruption, and its repair tools are not as user-friendly as those for Ext4.

ZFS: the file system to end all file systems

ZFS (Zettabyte File System) is a high-performance, advanced file system originally developed by Sun Microsystems (now part of Oracle) for the **Solaris** operating system. ZFS combines both a file system and a volume manager, providing integrated management of storage. It is known for its robust features, such as data integrity, high scalability, fault tolerance, and ease of management, making it a popular choice in enterprise environments, as well as for personal use by those requiring advanced features in Linux or FreeBSD.

Key features of ZFS include:

1. **Data Integrity and Checksumming**: One of ZFS's standout features is its data integrity capabilities. It uses checksums for all data and metadata. Every block of data is verified when read and written, ensuring that corruption is detected and corrected. If corruption occurs (e.g. from disk errors), ZFS can automatically attempt to repair it using redundant copies or RAID-like configurations.
2. **Copy-on-Write (COW)**: ZFS utilizes copy-on-write, meaning that when data is modified, it is not overwritten in place. Instead, new data is written to a different location, and once the write operation is complete, the system updates its pointers to the new location. This ensures that the file system always remains in a consistent state, even in the event of a power failure or system crash.
3. **Snapshots**: ZFS supports snapshots, which are read-only or read-write copies of the file system at a particular point in time. Snapshots are efficient in terms of storage because they only store changes made after the snapshot was taken, not the entire data set. This makes snapshots an ideal tool for rollbacks.
4. **Pooled Storage**: Unlike traditional file systems, ZFS combines file system and volume management, allowing it to create storage pools (called *Zpools*). Zpools abstract physical devices into a single logical pool of storage and data is automatically distributed across multiple devices. This allows ZFS to easily manage devices of varying sizes, providing better flexibility and redundancy. Additionally, pools can have cache devices, combinging the capacity of spinning hard drives with the speed of solid state drives to increase performance.
5. **RAID-Z**: ZFS includes its own RAID-like functionality, called RAID-Z, which provides data redundancy and improved performance. RAID-Z can be thought of as a more advanced version of RAID 5, with better protection against data loss due to disk failures. RAID-Z offers different configurations: RAID-Z1 (single parity), RAID-Z2 (double parity), and RAID-Z3 (triple parity), providing varying levels of fault tolerance.

6. **Compression**: ZFS supports inline compression, meaning that data is compressed as it is written to disk. This can significantly reduce storage space usage. ZFS supports multiple compression algorithms (e.g. LZ4, ZLE, zstd). Compression is transparent and data is automatically decompressed when read.
7. **Deduplication**: ZFS offers deduplication, which ensures that duplicate data is stored only once. This feature can be particularly useful in environments where many identical files are stored (e.g. backups or virtual machine images). However, deduplication can be resource-intensive and should be used with caution on systems with limited memory.
8. **High Scalability**: ZFS is highly scalable, supporting file systems and storage pools up to 256 trillion YiB (2^{128} bytes) in size with a maximum file size of 16 EiB. It can handle vast amounts of data and large numbers of files, making it ideal for enterprise-level storage solutions and large-scale data environments.
9. **Self-Healing**: ZFS provides self-healing capabilities. In case of data corruption (detected through checksums), ZFS can automatically repair data by accessing redundant copies (e.g. from RAID-Z). This ensures data reliability without requiring manual intervention.

With features like data integrity, compression, snapshots, RAID-Z, and self-healing, ZFS offers exceptional storage management capabilities, making it ideal for large-scale enterprise environments and applications requiring high availability and data protection. However, its resource requirements, complexity, and limited support on non-Solaris platforms makes it diffucult to use outside of these environments. For desktop computers, other file systems are generally a better choice. For those who need enterprise-grade features and are willing to manage its complexity, ZFS is the uncontested choice.