Linux File Systems: Ext2 vs Ext3 vs Ext4 vs Xfs

Linux File Systems: Ext2 vs Ext3 vs Ext4 vs Xfs

ext2, ext3 and ext4 are all filesystems created for Linux. This article explains the following:

- **High level difference between these filesystems.**
- **How to create these filesystems.**
- **How to convert from one filesystem type to another.**

Ext2

- **Ext2 stands for second extended file system.**
- **It was introduced in 1993. Developed by Rémy Card.**
- **This was developed to overcome the limitation of the original ext file system.**
- **Ext2 does not have journaling feature.**
- **On flash drives, usb drives, ext2 is recommended, as it doesn't need to do the over head of journaling.**
- **Maximum individual file size can be from 16 GB to 2 TB**
- **Overall ext2 file system size can be from 2 TB to 32 TB**

How to create an ext2 filesystem

```
# mke2fs /dev/sda1
```

Ext3

- **Ext3 stands for third extended file system.**
- **It was introduced in 2001. Developed by Stephen Tweedie.**
- **Starting from Linux Kernel 2.4.15 ext3 was available.**
- **The main benefit of ext3 is that it allows journaling.**
- **Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.**
- **Maximum individual file size can be from 16 GB to 2 TB**
- **Overall ext3 file system size can be from 2 TB to 32 TB**
- **There are three types of journaling available in ext3 file system.**
  - **Journal – Metadata and content are saved in the journal.**
  - **Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.**
  - **Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.**
- **You can convert a ext2 file system to ext3 file system directly (without backup/restore).**

How to create ext3 file system :-

```
# mkfs.ext3 /dev/sda1
```

(or)

```
# mke2fs –j /dev/sda1
```

(-j for adding journaling capability)

How to convert ext2 to ext3 :-

```
# umount /dev/sda2
# tune2fs -j /dev/sda2
# mount /dev/sda2  /var
```

Ext4

- Ext4 stands for fourth extended file system.
- It was introduced in 2008.
- Starting from Linux Kernel 2.6.19 ext4 was available.
- Supports huge individual file size and overall file system size.
- Maximum individual file size can be from 16 GB to 16 TB
- Overall maximum ext4 file system size is 1 EB (exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte).
- Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
- You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
- Several other new features are introduced in ext4: multiblock allocation, delayed allocation, journal checksum. fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the filesystem when compared to ext3.
- In ext4, you also have the option of turning the journaling feature "off".

Creating ext4 file system :-

```
# mkfs.ext4 /dev/sda1
```

(or)

```
# mke2fs -t ext4 /dev/sda1
```

Converting ext3 to ext4

(Warning :- Never try this live or production servers)

```
# umount /dev/sda2
# tune2fs -O extents,uninit_bg,dir_index  /dev/sda2
# e2fsck -pf /dev/sda2
# mount /dev/sda2 /var
```

Find your servers filesystem type

We can find the filesystem type used in our servers using any one of the following commands

```
# mount
/dev/sda3 on / type ext3 (rw)
proc on /proc type proc (rw)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
# file -sL /dev/sda1
/dev/sda1: Linux rev 1.0 ext3 filesystem data (needs journal recovery)
# df -T | awk '{print $1,$2,$7}' | grep "^/dev"
/dev/sda3 ext3 /
/dev/sda1 ext3 /boot
```

XFS

XFS is a high-performance file system which was designed by SGI for their IRIX platform. Since XFS was ported to the Linux kernel in 2001, XFS has remained a preferred choice for many enterprise systems especially with massive amount of data, due to its high performance, architectural scalability and robustness. For example, RHEL/CentOS 7 and Oracle Linux have adopted XFS as their default file system, and SUSE/openSUSE have long been an avid supporter of XFS.
XFS has a number of unique features that make it stand out among the file system crowd, such as scalable/parallel I/O, journaling for metadata operations, online defragmentation, suspend/resume I/O, delayed allocation for performance, etc.
If you want to create and mount an XFS file system on your Linux platform, here is how to do it.
XFS is packed full of cool features like guaranteed rate I/O, online resizing, built-in quota enforcement, and it can theoretically support filesystems up to 8 exabytes in size. It's been used on Linux since about 2001, and is available as an install option on many popular Linux distributions. With variable block sizes, you can tune your system like a sliding scale to tweak for space efficiency or read performance. Best for extremely large file systems, large files, and lots of files Journaled (an asymmetric parallel cluster file system version is also available) POSIX extended access controls The XFS file system is Open Source and included in major Linux distributions. It originated from SGI (Irix) and was designed specifically for large files and large volume scalability. Video and multi-media files are best handled by this file system. Scaling to petabyte volumes, it also handles great deals of data. It is one of the few filesystems on Linux which supports Data Migration (SGI contributed the Hierarchical Storage Management interfaces into the Linux Kernel a number of years ago). SGI also offers a closed source cluster parallel version of XFS called cXFS which like cVxFS is an asymmetrical model. It has the unique feature, however, that it's slave nodes can run on Unix, Linux and Windows, making it a cross platform file system. Its master node must run on SGI hardware. Recommended Use: If you really like to tweak your system to meet your needs, XFS is a great way to go.

The XFS file system is an extension of the extent file system .XFS is a high performance 64 bit journaling file system .Support of XFS
was merged into the linux kernel in around 2002 and In 2009 Red Hat Enterprise Linux version 5.4 usage of XFS file system .
Now RHEL 7.0 uses XFS as the default file system .
XFS supports maximum file system size of 8 exbibytes for 64 bit file system .Some comparison of XFS file system is XFS file system cannot be shrunk and poor performance with
deletions of large numbers of files.
32-bit system 64-bit system
File size: 16 Terabytes 16 Exabytes
File system: 16 Terabytes 18 Exabytes

Creating Xfs file system

```
#fdisk /dev/sdb <-create font="" partition="" the="">
#mkfs.xfs -f /dev/sdb1
#mount -t xfs /dev/sdb1 /storage
```

```
#df -Th /storage
#################################################################
```

What’s the Difference

Between Linux EXT, XFS, and BTRFS Filesystems?

Linux supports a range of file systems, including
ones used on other operating systems such as Windows FAT and NTFS. Those may
be
supported by embedded developers but normally a Linux file system like the 4
extended file system (ext4), XFS, or BTRFS will be used for most storage
partitions. Understanding the options can help in selecting the right file
system for an application.

The Linux file systems covered here include ones
that would typically be used in embedded applications. There is also a class of
clustered file systems designed for multi-node environments like Red Hat’s
Global File System (GFS), GlusterFS, and Lustre.

We start with an overview of features, followed by a more detailed description of each.

The Extended File System

The Extended File System is actually a family that
currently includes ext2, ext3, and ext4. It was the de facto standard for many years and it is still commonly used.

The ext2 file system was introduced in 1993 and
supported Linux features like symboliclinks and long file names. It can now handle volumes up to 32 Tbytes and file sizes up to 2 Tbytes. It is still used on many flash-based storage systems along with the FAT file system. It lacks the journal system found in ext3 and ext4.

The ext2 file allocation uses a multilevel
hierarchy that provides fast access for smaller files *(Fig. 1)*. The
first dozen links in a file's inode reference data blocks for the start of the file. Larger files add a single- and then double-level reference before data is accessible.

1. The ext2 file system has a file definition (inode)
that includes details about the file and links to data nodes with links to
index nodes at the end. Small files would not use those.

An ext2 file system can be upgraded to ext3. The
ext3 file system adds journaling, online file system growth capabilities, and
an HTree indexing system for larger directories. If these features are not used
then the ext3 file system matches an ext2 file system.

The ext3 file system does not include newer
features like dynamic inode allocation and extents. The advantage is that the
file system metadata is in fixed, known locations. There are redundant data
structures in place to improve recoverability.

The journaling system for ext3 operates in one of
three modes: journal (low risk), ordered (medium risk) and writeback (high
risk). The first mode writes metadata and file contents to the journal,
effectively saving the information twice. The ordered mode only puts the
metadata in the journal and writes the file data exclusively to the file system
data structure before the journal entry is committed. The writeback mode makes
no guarantee as to when the metadata is committed. This is faster, but affects
what can be recovered if a crash occurs.

The ext4 file system adds a number of major
features, including file systems as large as 1 Ebyte and files up to 16 Tbytes.
Extents replace the traditional block-mapping mechanism used with its earlier
siblings. Even though it’s backward compatible, an ext4 file system cannot be

mounted as an ext3 file system if newer features like the extent support are enabled.

The ext4 file system supports persistent
pre-allocation—useful for applications like streaming media where sequential access performance is paramount. It also supports delayed allocation. The journals are checksummed for improved reliability, and there is a multiblock allocator. File system checking is significantly faster.

The ext4 system was built on the existing
infrastructure, making certain features difficult to implement. It does not currently support the secure deletion file attribute. Newer file systems like XFS and BTRFS offer improvements that may or may not be amenable to certain applications.

XFS

XFS is a 64-bit journaling file system initially
developed by Silicon Graphics. It is designed for parallel I/O based on allocation groups. This allows a system to scale based on the number of I/O threads and file system bandwidth. It is designed to span multiple storage devices. XFS includes its own volume manager.

It uses B+ trees for the directories and file
allocation. The file system is partitioned into allocation groups (AG) that
have their own allocation and free space management. Files are allocated used

extents *(Fig. 2)* that use contiguous blocks when possible. The number of extents usually grows as a file's size increases. XFS can handle variable block sizes, sparse files, and snapshots.

2. Extent-based allocation uses larger continuous blocks for more efficient storage and access. Extents are referenced via a B+ tree.

XFS uses a logging system to track updates. This process can be synchronous or asynchronous with respect to file updates. The former is more-or-less prone to problems when errors occur, but the latter is faster and more efficient when multiple tasks and files are involved. The log can be store on a separate device to further improve reliability. XFS has Data Migration API (DMAPI) that can take advantage of hierarchical storage management services.

One notable feature of XFS is Guaranteed Rate I/O (GRIO). This allows applications to reserve bandwidth. This can be very useful in embedded applications. The file system calculates the available performance and adjusts its operation according to the existing reservations.

BTRFS

The BTRFS file system is based on a copy-on-write (COW) B-tree. According to Chris Mason, the principal BTRFS author, its goal was “to let Linux scale for the storage that will be available. Scaling is not just about addressing the storage but also means being able to administer and to manage it with a clean interface that lets people see what’s being used and makes it more reliable.”

BTRFS is also an extent-based storage system like XFS. It is space efficient for small files and indexed directories, and supports dynamic inode allocation. It handles multiple storage devices and provides support for RAID striping, mirroring and striping+mirroring. BTRFS is also flash drive aware with direct support for the TRIM/discard operations.

The system supports compression, writeable, and read-only snapshots, along with efficient incremental backup. The sub-volume system allows separate internal file system roots as well as quota system implementations. Out-of-band deduplication is supported. These features can be very useful in high-availability embedded systems.

BTRFS has minimal information stored in fixed
locations. This is an advantage for ext2/3/4 migration since in-place
conversion is possible, assuming that sufficient free space is available. In
addition, BTRFS is the underlying storage system for Ceph, an open-source
cluster file system.

Feature	EXT4	XFS	BTRFS
Architecture	Hashed B-tree	B+ tree	Extent based
Introduced	2006	1994	2009
Max volume size	1 Ebytes	8 Ebytes	16 Ebytes
Max file size	16 Tbytes	8 Ebytes	16 Ebytes
Max number of files	4 billion	264	264
Max file name size	255 bytes	255 bytes	255 bytes
Attributes	Yes	Yes	Yes
Transparent compression	No		No
Transparent encryption	Yes	No	Yes
Copy-on-Write (COW)	No	Planned	Yes
Snapshots	No	Planned	Yes