ext4

The **ext4 journaling file system** or **fourth extended filesystem** is a journaling file system for Linux, developed as the successor to ext3.

ext4 was initially a series of backward-compatible extensions to ext3, many of them originally developed by Cluster File Systems for the Lustre file system between 2003 and 2006, meant to extend storage limits and add other performance improvements.[4] However, other Linux kernel developers opposed accepting extensions to ext3 for stability reasons,[5] and proposed to fork the source code of ext3, rename it as ext4, and perform all the development there, without affecting existing ext3 users. This proposal was accepted, and on 28 June 2006, Theodore Ts'o, the ext3 maintainer, announced the new plan of development for ext4.[6]

A preliminary development version of ext4 was included in version 2.6.19[7] of the Linux kernel. On 11 October 2008, the patches that mark ext4 as stable code were merged in the Linux 2.6.28 source code repositories,[8] denoting the end of the development phase and recommending ext4 adoption. Kernel 2.6.28, containing the ext4 filesystem, was finally released on 25 December 2008.[9] On 15 January 2010, Google announced that it would upgrade its storage infrastructure from ext2 to ext4.[10] On 14 December 2010, Google also announced it would use ext4, instead of YAFFS, on Android 2.3.[11]

Contents

- Adoption
- Features
- Limitations
- Delayed allocation and potential data loss
- Implementation
  - Compatibility with Windows and Macintosh
- See also
- References
- External links

Adoption

ext4	
Developer(s)	Mingming Cao, Andreas Dilger, Alex Zhuravlev (Tomas), Dave Kleikamp, Theodore Ts'o, Eric Sandeen, Sam Naghshineh, others
Full name	Fourth extended file system
Introduced	**Stable:** 21 October 2008 **Unstable:** 10 October 2006 with Linux 2.6.28, 2.6.19
Partition identifier	0x83: MBR / EBR. EBD0A0A2-B9E5-4433-87C0-68B6B72699C7: GPT Windows BDP.[1] 0FC63DAF-8483-4772-8E79-3D69D8477DE4: GPT Linux filesystem data.[1] 933AC7E1-2EB4-4F13-B844-0E14E2AEF915: GPT /home partition.[2] 3B8F8425-20E0-4F3B-907F-1A25A76F98E8: GPT /srv (server data) partition.
Structures	
Directory contents	Linked list, hashed B-tree
File allocation	Extents / Bitmap
Bad blocks	Table

ext4 is the default file system for many Linux distributions including Debian and Ubuntu.[12]

Features

Large file system
: The ext4 filesystem can support volumes with sizes up to 1 exbibyte (EiB) and single files with sizes up to 16 tebibytes (TiB) with the standard 4 KiB block size.[13] The maximum file, directory, and filesystem size limits grow at least proportionately with the filesystem block size up to the maximum 64 KiB block size available on ARM and PowerPC/Power ISA CPUs.

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. A single extent in ext4 can map up to 128 MiB of contiguous space with a 4 KiB block size.[4] There can be four extents stored directly in the inode. When there are more than four extents to a file, the rest of the extents are indexed in a tree.[14]

Backward compatibility
: ext4 is backward-compatible with ext3 and ext2, making it possible to mount ext3 and ext2 as ext4. This will slightly improve performance, because certain new features of the ext4 implementation can also be used with ext3 and ext2, such as the new block allocation algorithm, without affecting the on-disk format.
ext3 is partially forward-compatible with ext4. Practically, ext4 will not mount as an ext3 filesystem out of the box, unless certain new features are disabled when creating it, such as `^extent`, `^flex_bg`, `^huge_file`, `^uninit_bg`, `^dir_nlink`, and `^extra_isize`.[15]

Persistent pre-allocation
: ext4 can pre-allocate on-disk space for a file. To do this on most file systems, zeroes would be written to the file when created. In ext4 (and some other files systems such as XFS) `fallocate()`, a new system call in the Linux kernel, can be used. The allocated space would be guaranteed and likely contiguous. This situation has applications for media streaming and databases.

Delayed allocation
: ext4 uses a performance technique called allocate-on-flush, also known as *delayed allocation*. That is, ext4 delays block allocation until data is flushed to disk; in contrast, some file systems allocate blocks immediately, even when the data goes into a write

Limits	
Max. volume size	1 EiB (for 4 KiB block size)
Max. file size	16 TiB (for 4 KiB block size)
Max. number of files	4 billion (specified at filesystem creation time)
Max. filename length	255 bytes
Allowed characters in filenames	All bytes except NUL ('\0') and '/' and the special file names "." and ".." which are not forbidden but are always used for a respective special purpose.
Features	
Dates recorded	modification (mtime), attribute modification (ctime), access (atime), delete (dtime), create (crtime)
Date range	14 December 1901 - 10 May 2446[3]
Date resolution	Nanosecond
Forks	No
Attributes	acl, bh, bsddf, commit=nrsec, data=journal, data=ordered, data=writeback, delalloc, extents, journal_dev, mballoc, minixdf, noacl, nobh, nodelalloc, noextents, nomballoc, nombcache, nouser_xattr, oldalloc, orlov, user_xattr
File system permissions	POSIX, POSIX ACLs
Transparent compression	No
Transparent	Yes

cache. Delayed allocation improves performance and reduces fragmentation by effectively allocating larger amounts of data at a time.

Unlimited number of subdirectories

ext4 does not limit the number of subdirectories in a single directory, except by the inherent size limit of the directory itself. (In ext3 a directory can have at most 32,000 subdirectories.)[16] To allow for larger directories and continued performance, ext4 in Linux 2.6.23 and later turns on HTree indices (a specialized version of a B-tree) by default, which allows directories up to approximately 10–12 million entries to be stored in the 2-level HTree index and 2 GB directory size limit for 4 KiB block size, depending on the filename length. In Linux 4.12 and later the `largedir` feature enabled a 3-level HTree and directory sizes over 2 GB, allowing approximately 6 billion entries in a single directory.

encryption	
Data deduplication	No
Other	
Supported operating systems	Linux FreeBSD (full read/write support since version 12.0) macOS (read-only with ext4fuse, full with ExtFS) Windows (read–write without journaling with ext2fsd) KolibriOS (read-only)

Journal checksums

ext4 uses checksums[17] in the journal to improve reliability, since the journal is one of the most used files of the disk. This feature has a side benefit: it can safely avoid a disk I/O wait during journaling, improving performance slightly. Journal checksumming was inspired by a research article from the University of Wisconsin, titled *IRON File Systems*[18] (specifically, section 6, called "transaction checksums"), with modifications within the implementation of compound transactions performed by the IRON file system (originally proposed by Sam Naghshineh in the RedHat summit).

Metadata checksumming

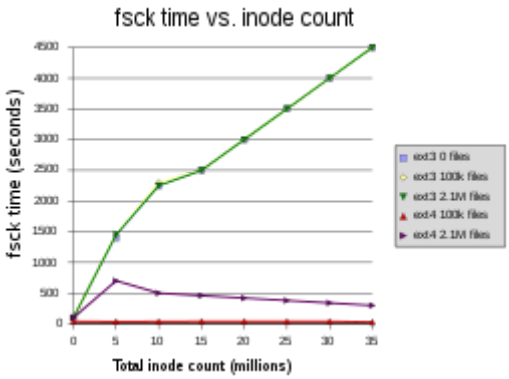
Since Linux kernel 3.5 released in 2012[19][20]

Faster file-system checking

In ext4 unallocated block groups and sections of the inode table are marked as such. This enables e2fsck to skip them entirely and greatly reduces the time it takes to check the file system. Linux 2.6.24 implements this feature.

Multiblock allocator

When ext3 appends to a file, it calls the block allocator, once for each block. Consequently, if there are multiple concurrent writers, files can easily become fragmented on disk. However, ext4 uses delayed allocation, which allows it to buffer data and allocate groups of blocks. Consequently, the multiblock allocator can make better choices about allocating files contiguously on disk. The multiblock allocator can also be used when files are opened in O_DIRECT mode. This feature does not affect the disk format.



fsck time dependence on inode count (ext3 vs. ext4)

Improved timestamps

As computers become faster in general, and as Linux becomes used more for mission-critical applications, the granularity of second-based timestamps becomes insufficient. To solve this, ext4 provides timestamps measured in nanoseconds. In addition, 2 bits of the expanded timestamp field are added to the most significant bits of the seconds field of the timestamps to defer the year 2038 problem for an additional 408 years.[3]

ext4 also adds support for time-of-creation timestamps. But, as Theodore Ts'o points out, while it is easy to add an extra creation-date field in the inode (thus technically enabling

support for these timestamps in ext4), it is more difficult to modify or add the necessary system calls, like stat() (which would probably require a new version) and the various libraries that depend on them (like glibc). These changes will require coordination of many projects.[21] Therefore, the creation date stored by ext4 is currently only available to user programs on Linux via the `statx()` API.[22]

Project quotas
: Support for project quotas was added in Linux kernel 4.4 on 8 Jan 2016. This feature allows assigning disk quota limits to a particular project ID. The project ID of a file is a 32-bit number stored on each file and is inherited by all files and subdirectories created beneath a parent directory with an assigned project ID. This allows assigning quota limits to a particular subdirectory tree independent of file access permissions on the file, such as user and project quotas that are dependent on the UID and GID. While this is similar to a directory quota, the main difference is that the same project ID can be assigned to multiple top-level directories and is not strictly hierarchical.[23]

Transparent encryption
: Support for transparent encryption was added in Linux kernel 4.1 on June 2015.[24]

Lazy initialization
: The lazyinit feature allows to clean inode tables in background, speeding initialization when creating a new ext4 file system.[25] It is available since 2010 in Linux kernel version 2.6.37.[26]

Write barriers
: ext4 enables write barriers by default. It ensures that file system metadata is correctly written and ordered on disk, even when write caches lose power. This goes with a performance cost especially for applications that use fsync heavily or create and delete many small files. For disks with a battery-backed write cache, disabling barriers (option 'barrier=0') may safely improve performance.[27]

Limitations

In 2008, the principal developer of the ext3 and ext4 file systems, Theodore Ts'o, stated that although ext4 has improved features, it is not a major advance, it uses old technology, and is a stop-gap. Ts'o believes that Btrfs is the better direction because "it offers improvements in scalability, reliability, and ease of management".[28] Btrfs also has "a number of the same design ideas that reiser3/4 had".[29] However, ext4 has continued to gain new features such as file encryption and metadata checksums.

The ext4 file system does not honor the "secure deletion" file attribute, which is supposed to cause overwriting of files upon deletion. A patch to implement secure deletion was proposed in 2011, but did not solve the problem of sensitive data ending up in the file-system journal.[30]

Delayed allocation and potential data loss

Because delayed allocation changes the behavior that programmers have been relying on with ext3, the feature poses some additional risk of data loss in cases where the system crashes or loses power before all of the data has been written to disk. Due to this, ext4 in kernel versions 2.6.30 and later automatically handles these cases as ext3 does.

The typical scenario in which this might occur is a program replacing the contents of a file without forcing a write to the disk with fsync. There are two common ways of replacing the contents of a file on Unix systems:[31]

- `fd=open("file", O_TRUNC); write(fd, data); close(fd);`

In this case, an existing file is truncated at the time of open (due to `O_TRUNC` flag), then new data is written out. Since the write can take some time, there is an opportunity of losing contents even with ext3, but usually very small. However, because ext4 can delay writing file data for a long time, this opportunity is much greater.
There are several problems that can arise:

1. If the write does not succeed (which may be due to error conditions in the writing program, or due to external conditions such as a full disk), then both the original version *and* the new version of the file will be lost, and the file may be corrupted because only a part of it has been written.
2. If other processes access the file while it is being written, they see a corrupted version.
3. If other processes have the file open and do not expect its contents to change, those processes may crash. One notable example is a shared library file which is mapped into running programs.

Because of these issues, often the following idiom is preferred over the one above:

- `fd=open("file.new"); write(fd, data); close(fd); rename("file.new", "file");`

  A new temporary file ("file.new") is created, which initially contains the new contents. Then the new file is renamed over the old one. Replacing files by the `rename()` call is guaranteed to be atomic by POSIX standards – i.e. either the old file remains, or it's overwritten with the new one. Because the ext3 default "ordered" journaling mode guarantees file data is written out on disk before metadata, this technique guarantees that either the old or the new file contents will persist on disk. ext4's delayed allocation breaks this expectation, because the file write can be delayed for a long time, and the rename is usually carried out before new file *contents* reach the disk.

Using `fsync()` more often to reduce the risk for ext4 could lead to performance penalties on ext3 filesystems mounted with the `data=ordered` flag (the default on most Linux distributions). Given that both file systems will be in use for some time, this complicates matters for end-user application developers. In response, ext4 in Linux kernels 2.6.30 and newer detect the occurrence of these common cases and force the files to be allocated immediately. For a small cost in performance, this provides semantics similar to ext3 ordered mode and increases the chance that either version of the file will survive the crash. This new behavior is enabled by default, but can be disabled with the "noauto_da_alloc" mount option.[31]

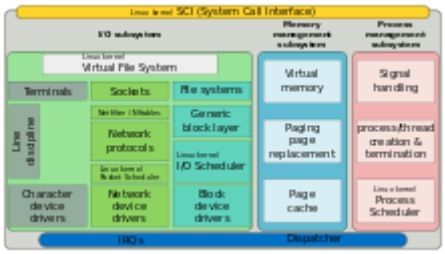
The new patches have become part of the mainline kernel 2.6.30, but various distributions chose to backport them to 2.6.28 or 2.6.29.[32]

These patches don't completely prevent potential data loss or help at all with new files. The only way to be safe is to write and use software that does `fsync()` when it needs to. Performance problems can be minimized by limiting crucial disk writes that need `fsync()` to occur less frequently.[33]

Implementation

Linux kernel Virtual File System is a subsystem or layer inside of the Linux kernel. It is the result of the very serious attempt to integrate multiple file systems into an orderly single structure. The key idea, which dates back to the pioneering work done by Sun Microsystems employees in 1986,[34] is to abstract out that part of the file system that is common to all file systems and put that code in a separate layer that calls the underlying concrete file systems to actually manage the data.

All system calls related to files (or pseudo files) are directed to the Linux kernel Virtual File System for initial processing. These calls, coming from user processes, are the standard POSIX calls, such as `open`, `read`, `write`, `lseek`, etc.



Simplified structure of the Linux kernel: ext4 is implemented between the Linux kernel Virtual File System and the generic block layer.

Compatibility with Windows and Macintosh

Currently, ext4 has full support on non-Linux operating systems.

Windows can access ext4 since Windows 10 Insider Preview Build 20211.[35][36][37] It is possible thanks to Windows Subsystem for Linux (WSL) which was introduced with Windows 10 Anniversary Update (version 1607) on 2 August 2016. WSL is available only in 64-bit versions of Windows 10 from version 1607. It is also available in Windows Server 2019. Big changes to the WSL architecture came with the release of WSL 2 on 12 June 2019.[38] WSL 2 requires Windows 10 version 1903 or higher, with build 18362 or higher, for x64 systems, and version 2004 or higher, with build 19041 or higher, for ARM64 systems.[39]

Paragon offers its commercial product Linux File Systems for Windows[40] which allows read/write capabilities for ext2/3/4 on Windows 7 SP1/8/8.1/10 and Windows Server 2008 R2 SP1/2012/2016.

macOS has full ext2/3/4 read–write capability through the extFS for Mac by Paragon Software,[41] which is a commercial product. Free software such as ext4fuse has read-only support with limited functionality.

See also

- Btrfs
- Comparison of file systems
- Extended file attributes
- e2fsprogs
- Ext2Fsd
- JFS
- List of file systems
- Reiser4
- XFS
- ZFS

References

1. Previously, Linux used the same GUID for the data partitions as Windows (Basic data partition: `EBD0A0A2-B9E5-4433-87C0-68B6B72699C7`). Linux never had a separate unique partition type GUID defined for its data partitions. This created problems when dual-booting Linux and Windows in UEFI-GPT setup. The new GUID (Linux filesystem data: `0FC63DAF-8483-4772-8E79-3D69D8477DE4`) was defined jointly by GPT fdisk and GNU Parted developers. It is identified as type code `0x8300` in GPT fdisk. (See definitions in gdisk's parttypes.cc (http://gptfdisk.git.sourceforge.net/git/gitweb.cgi?p%3Dgptfdisk/gptfdisk;a%3Dblob_plain;f%3Dparttypes.cc;hb%3DHEAD))
2. "DiscoverablePartitionsSpec" (http://www.freedesktop.org/wiki/Specifications/DiscoverablePartitionsSpec/). *freedesktop.org*. Retrieved 7 April 2018.

3. "ext4: Fix handling of extended tv_sec" (https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=a4dad1ae24f850410c4e60f22823cba1289b8d52). Linux-stable kernel tree. Retrieved 14 February 2017.
4. Mathur, Avantika; Cao, MingMing; Bhattacharya, Suparna; Dilger, Andreas; Zhuravlev (Tomas), Alex; Vivier, Laurent (2007). "The new ext4 filesystem: current status and future plans" (https://web.archive.org/web/20100706040230/http://www.linuxsymposium.org/archives/OLS/Reprints-2007/mathur-Reprint.pdf) (PDF). *Proceedings of the Linux Symposium*. Ottawa, ON, CA: Red Hat. Archived from the original (http://www.linuxsymposium.org/archives/OLS/Reprints-2007/mathur-Reprint.pdf) (PDF) on 6 July 2010. Retrieved 15 January 2008.
5. Torvalds, Linus (9 June 2006). "extents and 48bit ext3" (https://lkml.org/lkml/2006/6/9/183). Linux kernel mailing list.
6. Ts'o, Theodore (28 June 2006). "Proposal and plan for ext2/3 future development work" (https://lkml.org/lkml/2006/6/28/454). Linux kernel mailing list.
7. Leemhuis, Thorsten (23 December 2008). "Higher and further: The innovations of Linux 2.6.28 (page 2)" (https://web.archive.org/web/20090103164710/http://www.heise-online.co.uk/open/Kernel-Log-Higher-and-Further-The-innovations-of-Linux-2-6-28--/features/112299). Heise Online. Archived from the original (http://www.h-online.com/open/features/Kernel-Log-Higher-and-Further-The-innovations-of-Linux-2-6-28-746805.html) on 3 January 2009. Retrieved 9 January 2010.
8. "ext4: Rename ext4dev to ext4" (https://archive.is/20120529150649/http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=03010a3350301baac2154fa66de925ae2981b7e3). Linus' kernel tree. Archived from the original (https://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=03010a3350301baac2154fa66de925ae2981b7e3) on 29 May 2012. Retrieved 20 October 2008.
9. Leemhuis, Thorsten (23 December 2008). "Higher and further: The innovations of Linux 2.6.28" (http://www.heise-online.co.uk/open/Kernel-Log-Higher-and-Further-The-innovations-of-Linux-2-6-28--/features/112299). Heise Online.
10. Paul, Ryan (15 January 2010). "Google upgrading to Ext4, hires former Linux Foundation CTO" (https://arstechnica.com/open-source/news/2010/01/google-upgrading-to-ext4-hires-former-linux-foundation-cto.ars). *Ars Technica*.
11. "Android 2.3 Gingerbread to use Ext4 file system" (http://www.h-online.com/open/news/item/Android-2-3-Gingerbread-to-use-Ext4-file-system-1152775.html). *The H Open*. 14 December 2010.
12. "FileSystem in debian" (https://wiki.debian.org/FileSystem). 14 September 2019.
13. "Migrating to Ext4" (https://web.archive.org/web/20081201104450/http://www.ibm.com/developerworks/linux/library/l-ext4/). *DeveloperWorks*. IBM. Archived from the original (http://www.ibm.com/developerworks/linux/library/l-ext4/) on 1 December 2008. Retrieved 14 December 2008.
14. Hal Pomeranz (28 March 2011). "Understanding EXT4 (Part 3): Extent Trees" (https://digital-forensics.sans.org/blog/2011/03/28/digital-forensics-understanding-ext4-part-3-extent-trees). *SANS Digital Forensics and Incident Response Blog*.
15. "Mount of ext4 (created without extents) as ext3 fails on RH6.2" (http://www.linuxquestions.org/questions/red-hat-31/mount-of-ext4-created-without-extents-as-ext3-fails-on-rh6-2-a-936813/). *www.linuxquestions.org*. Retrieved 7 April 2018.
16. "Ext4 - Linux Kernel Newbies" (https://kernelnewbies.org/Ext4). *kernelnewbies.org*.
17. "New ext4 features - Ext4" (https://ext4.wiki.kernel.org/index.php/New_ext4_features#Metadata_Checksums). *ext4.wiki.kernel.org*.
18. Prabhakaran, Vijayan; Bairavasundaram, Lakshmi N.; Agrawal, Nitin; Gunawi, Haryadi S.; Arpaci-Dusseau, Andrea C.; Arpaci-Dusseau, Remzi H. "IRON File Systems" (http://www.cs.wisc.edu/wind/Publications/iron-sosp05.pdf) (PDF). CS Dept, University of Wisconsin.
19. "Ext4 Metadata Checksums - Ext4" (https://ext4.wiki.kernel.org/index.php/Ext4_Metadata_Checksums). *ext4.wiki.kernel.org*.

20. "Linux_3.5 - Linux Kernel Newbies" (https://kernelnewbies.org/Linux_3.5?highlight=(Metadata)%7C(checksumming)). *kernelnewbies.org*.
21. Ts'o, Theodore (5 October 2006). "Re: creation time stamps for ext4 ?" (https://www.redhat.com/archives/ext3-users/2006-October/msg00015.html).
22. Edge, Jake (31 March 2017). "Extending statx()" (https://lwn.net/Articles/718222/).
23. Li, Xi (12 January 2016). "Ext4 encryption" (https://lwn.net/Articles/671627/).
24. Ts'o, Theodore (8 April 2015). "Ext4 encryption" (https://lwn.net/Articles/639427/).
25. "Ext4 Filesystem - Thomas-Krenn-Wiki" (https://www.thomas-krenn.com/en/wiki/Ext4_Filesystem). *www.thomas-krenn.com*.
26. "kernel/git/torvalds/linux.git - Linux kernel source tree" (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=bfff68738f1cb5c93dab1114634cea02aae9e7ba). *git.kernel.org*.
27. "Ext4 - ArchWiki" (https://wiki.archlinux.org/index.php/ext4#Turning_barriers_off). *wiki.archlinux.org*.
28. Paul, Ryan (14 April 2009). "Panelists ponder the kernel at Linux Collaboration Summit" (https://arstechnica.com/open-source/news/2009/04/linux-collaboration-summit-the-kernel-panel.ars). *Ars Technica*. Retrieved 22 August 2009.
29. Theodore Ts'o (1 August 2008). "Re: reiser4 for 2.6.27-rc1" (https://lkml.org/lkml/2008/8/1/217). *linux-kernel* (Mailing list). Retrieved 31 December 2010.
30. Corbet, Jonathan (11 October 2011). "Securely deleting files from ext4 filesystems" (https://lwn.net/Articles/462437/).
31. "ext4 documentation in Linux kernel source" (https://www.kernel.org/doc/Documentation/filesystems/ext4.txt). 28 March 2009.
32. Ubuntu bug #317781 (https://bugs.launchpad.net/ubuntu/+source/linux/+bug/317781?comments=all) Long discussion between Ubuntu developers and Theodore Ts'o on potential data loss
33. Thoughts by Ted blog entry, 12 March 2009 (http://thunk.org/tytso/blog/2009/03/12/delayed-allocation-and-the-zero-length-file-problem/) A blog posting of Theodore Ts'o on the subject
34. Kleiman
35. Brandon LeBlanc (10 September 2020). "Announcing Windows 10 Insider Preview Build 20211" (https://blogs.windows.com/windows-insider/2020/09/10/announcing-windows-10-insider-preview-build-20211/). *Windows Blogs*. Retrieved 25 May 2021.
36. Pierre Boulay (10 September 2020). "Access Linux filesystems in Windows and WSL 2" (https://devblogs.microsoft.com/commandline/access-linux-filesystems-in-windows-and-wsl-2/). *Windows Command Line*. Retrieved 25 May 2021.
37. "Get started mounting a Linux disk in WSL 2" (https://docs.microsoft.com/en-gb/windows/wsl/wsl2-mount-disk). *Microsoft Docs*. Retrieved 25 May 2021.
38. Craig Loewen (12 June 2019). "WSL 2 is now available in Windows Insiders" (https://devblogs.microsoft.com/commandline/wsl-2-is-now-available-in-windows-insiders/). *Windows Command Line*. Retrieved 25 May 2021.
39. "Windows Subsystem for Linux Installation Guide for Windows 10" (https://docs.microsoft.com/en-gb/windows/wsl/install-win10). *Windows Docs*. Retrieved 25 May 2021.
40. "Linux File Systems for Windows" (https://www.paragon-software.com/home/linuxfs-windows/). *Paragon Software*. Retrieved 25 May 2021.
41. "extFS for Mac" (https://www.paragon-software.com/home/extfs-mac/). *Paragon Software*. Retrieved 25 May 2021.

External links

- ext4 documentation in Linux kernel source (https://www.kernel.org/doc/Documentation/filesystems/ext4.txt)

- Theodore Ts'o's discussion on ext4 (https://archive.is/20120712143749/kerneltrap.org/node/6776), 29 June 2006
- "ext4 online defragmentation" (https://ols.fedoraproject.org/OLS/Reprints-2007/sato-Reprint.pdf) (materials from Ottawa Linux Symposium 2007)
- "The new ext4 filesystem: current status and future plans" (https://www.kernel.org/doc/ols/2007/ols2007v2-pages-21-34.pdf) (materials from Ottawa Linux Symposium 2007)
- Kernel Log: Ext4 completes development phase as interim step to btrfs (http://heise-online.co.uk/news/Kernel-Log-Ext4-completes-development-phase-as-interim-step-to-btrfs--/111742), 17 October 2008
- "Ext4 block and inode allocator improvements" (https://ols.fedoraproject.org/OLS/Reprints-2008/kumar-reprint.pdf) (materials from Ottawa Linux Symposium 2008)
- "Ext4: The Next Generation of Ext2/3 Filesystem" (http://usenix.org/event/lsf07/tech/cao_m.pdf)
- Ext4 (and Ext2/Ext3) Wiki (https://ext4.wiki.kernel.org/)
- Ext4 (http://kernelnewbies.org/Ext4) wiki at kernelnewbies.org
- Native Windows port of Ext4 and other FS in CROSSMETA (https://web.archive.org/web/20120111062641/http://www.crossmeta.org/redmine)
- Ext2read (http://ext2read.sourceforge.net/) A windows application to read/copy ext2/ext3/ext4 files with extent and LVM2 support.
- Ext2Fsd (https://web.archive.org/web/20120723091043/http://www.ext2fsd.com/) Open source ext2/ext3/ext4 read/write file system driver for Windows. ext4 is supported from version 0.50 onwards
- Ext4fuse (https://github.com/gerard/ext4fuse) Open source read-only ext4 driver for FUSE. (Supports Mac OS X 10.5 and later, using MacFuse (https://code.google.com/p/macfuse/))

Retrieved from "https://en.wikipedia.org/w/index.php?title=Ext4&oldid=1025071376"

This page was last edited on 25 May 2021, at 15:23 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.