# So many filesystems...

RIK FARROW

Rik is the editor of ;login:.
rik@usenix.org

When I was at FAST 2009, I listened as a security researcher asked a prominent filesystem researcher why there are so many file-systems. This got me thinking about it: Why are there so many?

Early computers didn't even have filesystems: storage was tape, and some computers actually loaded card images of programs to be compiled then executed from tape. But once IBM began building dishwasher-sized disks, systems programmers needed to design data structures to organize both the data and the metadata that described the organization and attributes of that data.

A quick look at the Wikipedia page on filesystems [1] makes it clear that there are many different filesystems. In a talk by Ted Ts'o [2] in 2010, Ted points out that there is support for 66 filesystems in the Linux 2.6 kernel. I asked Ted about that, and he told me something that should have been immediately obvious to me: some filesystems have specific use cases. For example, there are network filesystems, such as nfs, nfs4, cifs, afs, and 9p. There are also filesystems for compatibility with other systems: fat/vfat/msdos, iso9660, hfs, ntfs, minixfs, hfs, qnx4, qnx6, and so on. And finally, there are cluster filesystems, such as ceph, gfs2, and ocfs2, and special purpose filesystems, for example, for working well with SSDs.

But what about the "big four" disk filesystems in Linux: ext3, ext4, XFS, and btrfs? Why is there more than a single filesystem that gets used with modern versions of Linux?

Different filesystems have different strengths, and sometimes, weaknesses. XFS was designed to work with very large files and directories, for example, and was the filesystem of choice for this reason for many years. Now ext4 and btrfs can also handle large files, although their limits are still smaller than XFS, few people will be using 16 terabyte files (the limit of testing for ext4, according to Ric Wheeler of Red Hat during his BoF at FAST '13).

Ext3 added journaling to ext2, a method that writes a journal, a list of changes, before committing those changes. The purpose of journaling was to make recovering from system crashes or power failures much quicker. Running fsck on large filesystems can take hours, but with journaling, restoring filesystem integrity takes just seconds.

The ext4 filesystem includes changes that extend the capabilities of ext3 so that it can handle larger files and directories. Ext4 also uses extents, rather than indirect blocks, to handle large files more efficiently. Before that, only XFS (of this group) used extents, essentially, ranges of virtual blocks, instead of the lists of blocks found in ext2 and ext3. Google uses ext4 for the base of its cluster filesystem, but without journaling. Google, like many cluster filesystem users, uses replication as a backup strategy; they decided not to use journaling, which includes a 10% performance hit, because they can rebuild systems instead of running fsck on large volumes.

Btrfs was designed to fulfill features created by Oracle's ZFS: snapshotting, checksums on data and metadata (for detecting silent corruption), and the ability to expand filesystems (even across device boundaries). With btrfs, you can have backups made every time a file changes, and the checksums for these changes ripple up the directory hierarchy as well. So whereas btrfs works well for many uses, it would be a poor choice for databases and logfiles. In the 3.8 Linux kernel, support has been added for disabling checksums in files that change often.

Of these "big four", btrfs is the newest and just becoming stable enough for enterprise use. You might also have noticed that the newer filesystems, btrfs and ext4, have added features found in earlier filesystems, like those in XFS. So whereas XFS was once the only choice for very large filesystems, that has changed.

My security friend's interest in filesystems really had nothing to do with any of these issues: size, reliability, fast recovery, extensibility, or snapshotting. Simson Garfinkel was researching how much useful, and potentially dangerous, data was being left on discarded/recycled drives [4]. When you are scanning hundreds of drives looking for CAD files, personal information, and other sensitive material, having a plethora of filesystems is simply a nuisance.

### *References*

[1] File system: http://en.wikipedia.org/wiki/File_system.

[2] Ted Ts'o, "Making Production-Ready Filesystems: A Case Study Using Ext4": ftp://ftp.kernel.org/pub/linux/kernel/people/tytso/presentations/stabilizing-ext4.pdf.

[3] Jonathan Corbet, "Why Filesystems Are Hard": http://lwn.net/Articles/370419/.

[4] Simson Garfinkel, "Read Data Corpus": http://simson.net/page/Real_Data_Corpus.