SEE TEXT ONLY


# WeGetletters

by Michael W Lucas

**Hey, FJ Letters Dude,**
**Which filesystem should I use?**
**—FreeBSD Newbie**

Dear FreeBSD Newbie,
First off, welcome to FreeBSD. The wider community is glad to help you.

Second, please let me know who told you to start off by writing me. I need to properly… "thank" them.

Filesystems? Sure, let's talk filesystems.

Discussing which filesystem is the worst is like debating the merits of two-handed swords as compared to lumberjack-grade chainsaws and industrial tulip presses. While every one of them has perfectly legitimate uses, in the hands of the novice they're far more likely to maim everyone involved. It doesn't matter what operating system you use: FreeBSD, any BSD, Linux, Windows, illumos, whatever. Filesystems are the literal worst.

I mean, let's look at memory filesystems. The base idea, taking a chunk of memory and using it for temporary storage, seems sound enough. Most non-virtual computers these days have more than enough memory that they can blow a few gigabytes for a speedy /tmp or perhaps even compiler scratch space. Configuring poudriere to use memory for temporary files will vastly accelerate your package builds.

But FreeBSD has two different memory filesystems, mfs(5) and tmpfs(5). Old-fashioned MFS blats a UFS filesystem down on top of a chunk of memory. It's fast, sure. But any space MFS uses is unavailable for other use as long as the filesystem exists. Suppose you create a 5 GB /tmp with MFS, copy 4.9 GB to it, and erase it. That 4.9 GB is still tied up. You can instruct MFS to free unused memory by enabling TRIM with tunefs(8), but nobody bothers with that.

The newer alternative, tmpfs, is specifically designed for temporary filesystems. A default tmpfs has a maximum size equal to the system memory plus the system's swap space. "How much memory do you have? Give it to me." Be sure to specify the size= flag when you create a tmpfs, or be careful to monitor tmpfs space use. Not that you'll configure your monitoring system to watch tmpfs, because it's temporary.

And no matter what, one day you'll forget that you used memory space as a filesystem. You'll stash something vital in that temporary space, then reboot. And get really annoyed when that vital data vanishes into the ether.

Some other filesystems aren't actively terrible. The device filesystem devfs(5) provides device nodes. Filesystems that can't store user data are the best filesystems. But then some clever sysadmin decides to hack on /etc/devfs.rules to change the standard device nodes for their special application, or /etc/devd.conf to create or reconfigure device nodes, and the whole system goes down the tubes.

Speaking of clever sysadmins, now and then people decide that they want to optimize disk space or cut down how many copies of a file they need to maintain by reusing a partition or dataset elsewhere on the system. FreeBSD's nullfs(5) lets you mount a partition multiple times, essentially recycling the same chunk of disk space. Folks who use a bunch of jails use read-only nullfs mounts to have a single FreeBSD base install support multiple jails.

FreeBSD's unionfs(5) lets you merge multiple filesystems. Many people successfully use unionfs to provide custom views of a filesystem, again for jails. Unionfs is perhaps the least popular filesystem in the FreeBSD ecosystem though. I know several developers who won't go near it. I know others who say it's perfectly safe. All I know is, backups are good.

Network filesystems? Oh please. A dedicated 6GB/s SATA controller is always going to outperform anything that runs over gigabit Ethernet, especially if you're using that same network interface to manage the host on. Yes, six gig is more than one gig—but that comparison also has bits versus bytes. You're looking at a 48-fold difference in optimal throughput. And always remember that not all network switches are created equal. I have a whole stack of so-called "gigabit" switches that utterly refuse to pass more than a quarter gigabit a second.

I must unwillingly concede that FreeBSD's new

iSCSI stack is rock solid. And FreeBSD's NFS implementation is among the best in the world. Many people use these in high-performance applications… but they're still networked filesystems. These people battering them in production have top-notch network cards and switches that live up to the hype. If you ask on the mailing lists or forums, they'll offer their advice.

FreeBSD has excellent support for the new NFSv4 protocol. While earlier versions of NFS interoperate pretty well and have identical behavior, NFSv4 is a whole different beast with different semantics. You really need to do some reading before deploying it. NFSv4 does have an extensive access control list system that lets you perfectly implement the worst abominations a large corporation's IT department can dream up, so that's something.

You'll occasionally see mentions of the process filesystem, procfs(5). Many FreeBSD developers really, really don't want procfs to exist. When I documented a need for procfs in the 2018 version of ***Absolute FreeBSD***, technical reviewer John Baldwin rewrote ps(1) to make procfs unnecessary. As far as I can tell, the quickest way to goad a FreeBSD developer into action is to need /proc.

Autofs(5) was written for desktop users. It automatically identifies filesystems and mounts them for you. If you enable autofs and plug in a USB drive, the various partitions and labels on the drive will appear as directories in /media. Going into one of those directories will automatically mount that partition. Similarly, autofs makes NFS mount points available in /net. Listing the contents of /net/fileserver displays all the NFS mount points on the host fileserver, and going into one of those directories automatically mounts the share. It's still using a networked filesystem though, so it'll almost certainly end in tears.

In the defense of all of FreeBSD's filesystems, though, I must say: at least they're not EXTFS. Although FreeBSD supports extfs(5) as well, so that's not much help.

Really, the only smart move with filesystems is not to play.

**FJ Letters Dude,**
**I meant, I'm looking at the installer and it wants to know if I want to use UFS or ZFS? And George Neville-Neil said you needed letters.**

**—FreeBSD Newbie**

Oh!
Use ZFS, unless you can't.

As a new user, don't use ZFS on systems with less than two GB of RAM. Four GB or more would be wiser. Don't use ZFS on non-64-bit platforms.

Some virtualization systems don't properly label disk images during migration from one host to another. ZFS pools migrated on such systems won't boot. If you're running on a virtualization platform, test migration on a ZFS host before deploying it everywhere.

And thanks for the tip. Next time I run into GNN, we'll discuss his unfortunate tendency to encourage people.

**Michael W Lucas (https://mwl.io) is the author of too many books, including *Absolute FreeBSD*, *FreeBSD Mastery: Specialty Filesystems*, and *git commit murder*. Send your questions to letters@freebsdjournal.com. Letters will be answered in the order in which they amuse, annoy, or inspire the columnist, and may be edited for his own purposes.**