

The overview is divided into several sections. I'd encourage you to jump around to topics of interest or skip right to the conclusion (or to the tweet summary). Highest praise goes to encryption; ire to data integrity.

Table of Contents

Basics

Hands-on with the emulator that adds depth to old 2D NES games

- Paying down debt
- Encryption
- Snapshots and backup
- Management
- Space efficiency
- Performance
- Data Integrity
 - Redundancy
 - Crash Consistency
 - Checksums
 - Scrub
- Summing Up

Basics

APFS, the Apple File System, was itself started in 2014 with Giampaolo as its lead engineer. It's a stand-alone, from-scratch implementation (an earlier version of this post noted a dependency on Core Storage, but Giampaolo set me straight in this comment). I asked him about looking for inspiration in other modern file systems such as BSD's HAMMER, Linux's btrfs, or OpenZFS (Solaris, illumos, FreeBSD, Mac OS X, Ubuntu Linux, etc.), all of which have features similar to what APFS intends to deliver. (And note that Apple built a fairly complete port of ZFS, though Giampaolo was not apparently part of the group advocating for it.) Giampaolo explained that he was aware of them as a self-described file system guy (he built the file system in BeOS, unfairly relegated to obscurity when Apple opted to purchase NeXTSTEP instead), but didn't delve too deeply for fear, he said, of tainting himself.

Giampaolo praised the APFS testing team as being exemplary. This is absolutely critical. A common adage is that it takes a decade to mature a file system, and my experience with ZFS more or less confirms this. Apple will be delivering APFS broadly with 3-4 years of development so will need to accelerate quickly to maturity.

Paying down debt

HFS was introduced in 1985 when the Mac 512K (of memory! Holy smokes!) was Apple's flagship. HFS+, a significant iteration, shipped in 1998 on the G3 PowerMacs with 4GB hard drives. The typical storage capacity of a home computer has increased by a factor of over 1,000 since 1998 (and let's not even talk about 1985).. HFS+ has been pulled in a bunch of competing directions with different forks for different devices (e.g. I've been told by inside sources that the iOS team created their own HFS variant, working so covertly that not even the Mac OS team knew) and different features (e.g. journaling, case sensitivity). It's old. It's a mess. And, critically, it's missing a bunch of features that are really considered basic costs of doing business for most operating systems. Wikipedia lists nanosecond timestamps, checksums, snapshots, and sparse file support among those missing features. Add to that the obvious gap of large device support and you've got a big chunk of the APFS feature list.

APFS first and foremost pays down the unsustainable technical debt that Apple has been carrying in HFS+. (In 2001 ZFS grew from a similar need where UFS had been evolved since 1977.) It unifies the multifarious forks. It introduces the expected features. In general it first brings the derelict building up to code.

Compression is an obvious common feature that's missing in the APFS feature list. It's conceptually quite easy, I told the development team (we had it in ZFS from the outset), so why not include it? To appeal to Giampaolo's BeOS nostalgia I even recalled my job interview with Be in 2000 when they talked about how compression actually improved overall performance since data I/O is far more expensive than computation (obvious now, but novel then). The Apple folks agreed, and—in typical Apple fashion—neither confirmed nor denied while strongly implying that it's definitely a feature we can expect in APFS. I'll be surprised if compression isn't included in its public launch.

Encryption

Encryption is clearly a core feature of APFS. This comes from diverse requirements from the various devices; for example, multiple keys within file systems on the iPhone, or per-user keys on laptops. I heard the term "innovative" quite a bit at WWDC, but here the term is aptly applied to APFS. It supports several different encryption choices for a file system:

- Unencrypted
- Single-key for metadata and user data
- Multi-key with different choices for metadata, files, and even sections of a file ("extents")

Standalone 3DNES emulator is a promising starting effort with some rough edges.

STAY IN THE KNOW WITH 🖌



LATEST NEWS



Tokyo Mirage Sessions: #FE: Auto-tuned expression



Y Combinator's Xerox Alto: restoring the legendary 1970s GUI computer



World going to hell? Here are the Solar System's five most livable places



The auto engineering marvels at the Lane Motor Museum are ridiculously cool

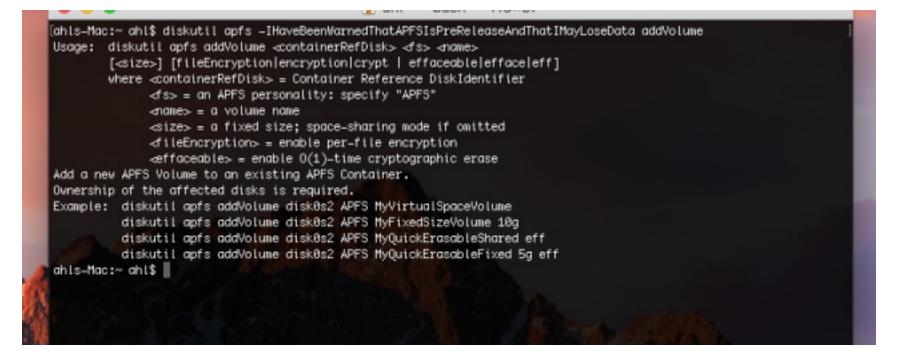


To mitigate poverty, Y Combinator set to launch minimum income plan

THE LONG MARCH

China's powerful new rocket makes a successful debut launch

Multi-key encryption is particularly relevant for portables where all data might be encrypted, but unlocking your phone provides access to an additional key and therefore additional data. Unfortunately this doesn't seem to be working in the first beta of macOS Sierra (specifying fileEncryption when creating a new volume with diskutil results in a file system that reports "Is Encrypted" as "No").



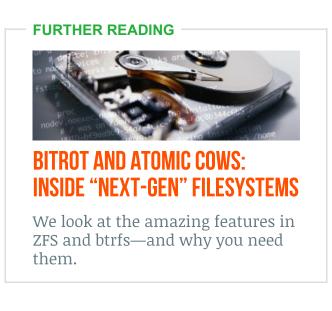
Enlarge / Can't even make this up!

Related to encryption, I noticed an undocumented feature while playing around with diskutil (which prompts you for interactive confirmation of the destructive power of APFS unless this is added to the command-line: -IHaveBeenWarnedThatAPFSIsPreReleaseAndThatIMayLoseData; I'm not making this up). APFS (apparently) supports the ability to securely and instantaneously erase a file system with the "effaceable" option when creating a new volume in diskutil. This presumably builds a secret key that cannot be extracted from APFS and encrypts the file system with it. A secure erase then need only delete the key rather than needing to scramble and re-scramble the full disk to ensure total eradication. Various iOS docs refer to this capability requiring some specialized hardware; it will be interesting to see what the option means on macOS. Either way, let's not mention this to the FBI or NSA, agreed?

Snapshots and backup

APFS brings a much-desired file system feature: snapshots. A snapshot lets you freeze the state of a file system at a particular moment and continue to use and modify that file system while preserving the old data. It does so in a space-efficient fashion where, effectively, changes are tracked and only new data takes up additional space. This has the potential to be extremely valuable for backup by efficiently tracking the data that has changed since the last backup.

ZFS includes snapshots and serialization mechanisms that make it efficient to back up file systems or transfer file systems to a remote location. Will APFS work like that? Probably not, answered Giampaolo. ZFS sends all changed data, while Time Machine can have exclusion lists and the like. That seems surmountable, but we'll see what Apple does. APFS right now is incompatible with Time Machine due to the lack of directory hard links, a fairly disgusting implementation that likely contributes to Time Machine's questionable reliability. Hopefully APFS will create some efficient serialization for Time Machine backup.



While APFS dev manager Eric Tamura demonstrated snapshots at WWDC, the required utilities aren't included in the macOS Sierra beta. I used DTrace (technology I'm increasingly amazed that Apple ported from OpenSolaris) to find a tantalizingly named new system call fs_snapshot; I'll leave it to others to reverse engineer its proper use.

Share - View Constant	Email 8 ⁺ Go		ADER COMMENTS – 51	
← OLDER STORY		NEWER STOR	ightarrow	
/OU MAY ALSO LIKE 🔺				
SITE LINKS About Us	MORE READING RSS Feeds		CONDE NAST SITES Reddit	
Advertise with us Contact Us	Newsletters		Wired Vanity Fair	
eprints	Visit Ars Technica UK		Style Details	
UBSCRIPTIONS Subscribe to Ars			Visit our sister sites	•
			Subscribe to a magazine	
		CONDÉ	NAST	

WIRED Media: Ars Technica and WIRED © 2016 Condé Nast. All rights reserved

Use of this Site constitutes acceptance of our User Agreement (effective 1/2/14) and Privacy Policy (effective 1/2/14), and Ars Technica Addendum (effective 5/17/2012) Your California Privacy Rights

/IEW MOBILE SITE

The material on this site may not be reproduced, distributed, transmitted, cached or otherwise used, except with the prior written permission of Condé Nast.

Ad Choices