Storing our digital lives

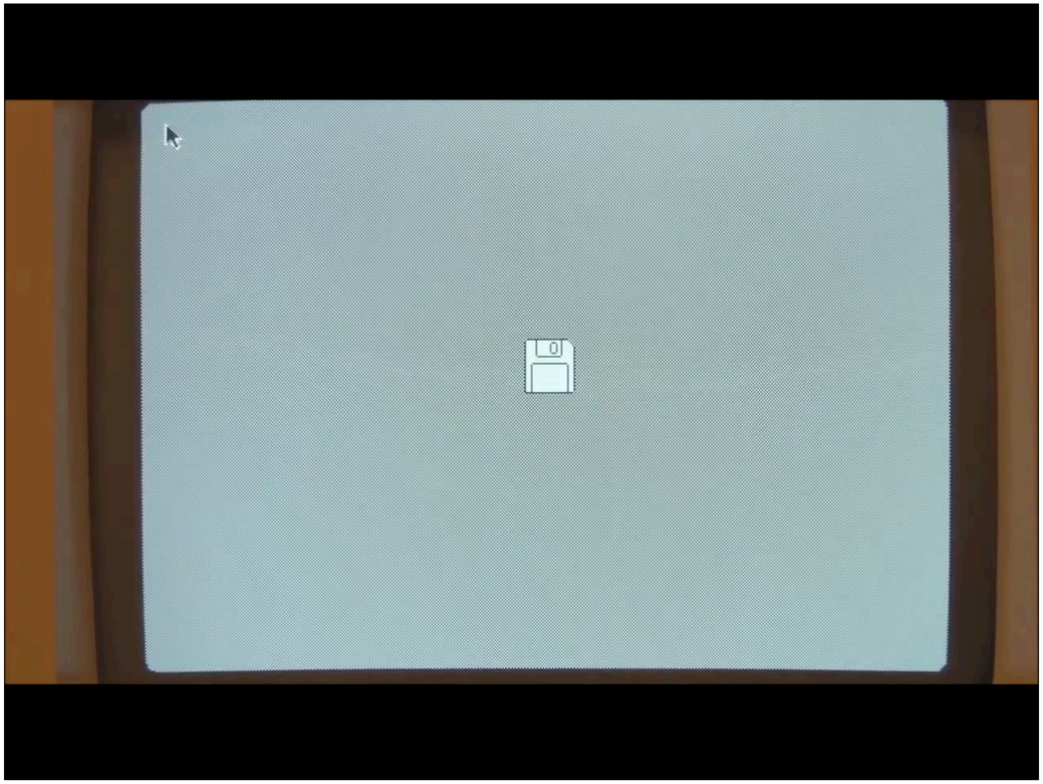
Mac filesystems from MFS to APFS

Rich Trouton

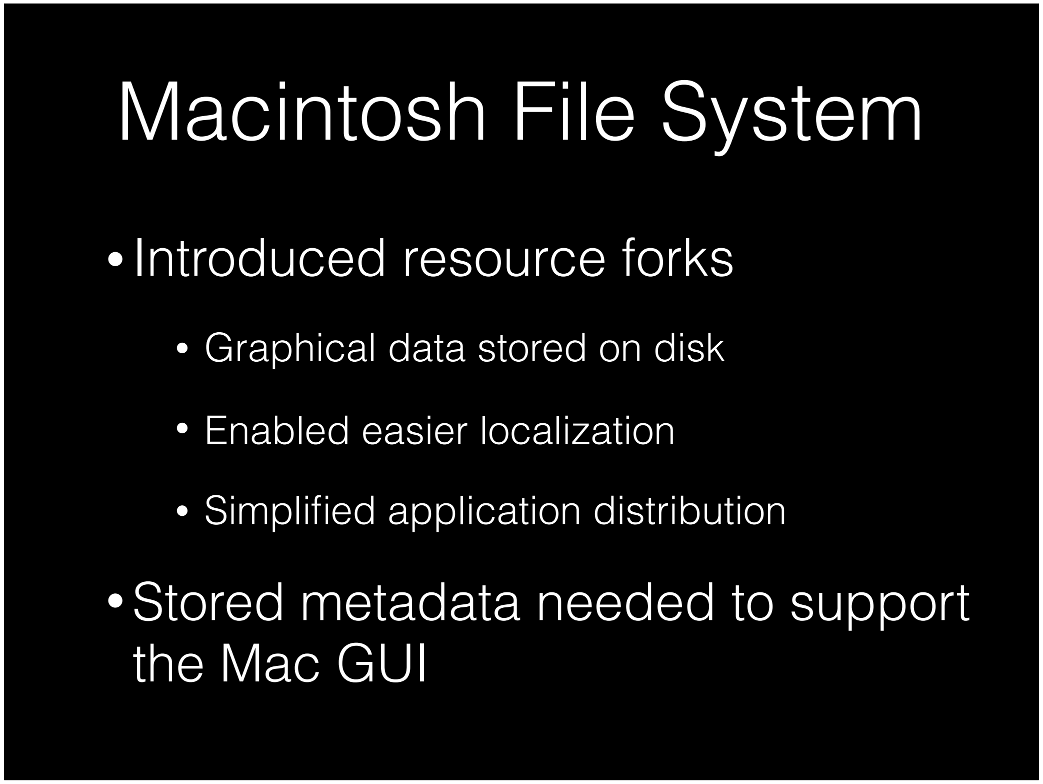
Mac CoE @ SAP

Before we get started, there’s two things I’d like to mention. The first is that, all of the sides, speakers’ notes and the demos are available for download and I’ll be providing a link at the end of the talk. I tend to be one of those folks who can’t keep up with the speaker and take notes at the same time, so for those folks in the same situation, no need to take notes. Everything I’m covering is going to be available for download.

The second is to please hold all questions until the end. If you’ve got questions, make a note of them and ask me afterwards. With luck, I’ll be able to answer most of your questions during the talk itself.



In the beginning, which for this discussion is 1984, there was the Macintosh File System (MFS). This file system was introduced with the original Apple Macintosh computer in January 1984 and was designed to store files on 400 kilobyte floppy disks.



Macintosh File System

- Introduced resource forks
  - Graphical data stored on disk
  - Enabled easier localization
  - Simplified application distribution
- Stored metadata needed to support the Mac GUI

MFS had a number of features that would be familiar to later Macintosh users:

A. It introduced resource forks as a way to accomplish the following:

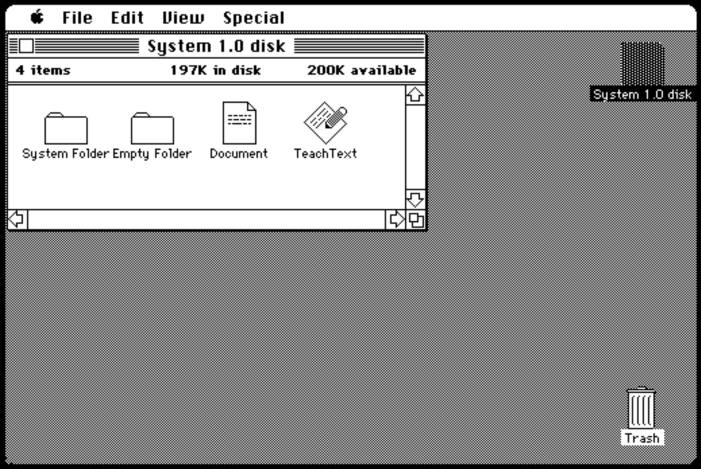
1. It allowed graphical data to be stored on disk until it was needed, allowing the OS to retrieve that data into memory, draw it on the screen, then discard it from memory.

2. They allowed easier translation of applications for a foreign market. An application's pictures and text were stored in the resource fork, so only the resource fork needed to be edited in order to make an application available for a different language or nation.

3. Resource folks also allowed the distribution of all or almost all of the components of an application inside a single file. This simplified application installation and removal.

B. The other feature MFS introduced was storing the metadata needed to support the graphical user interface of the Mac's

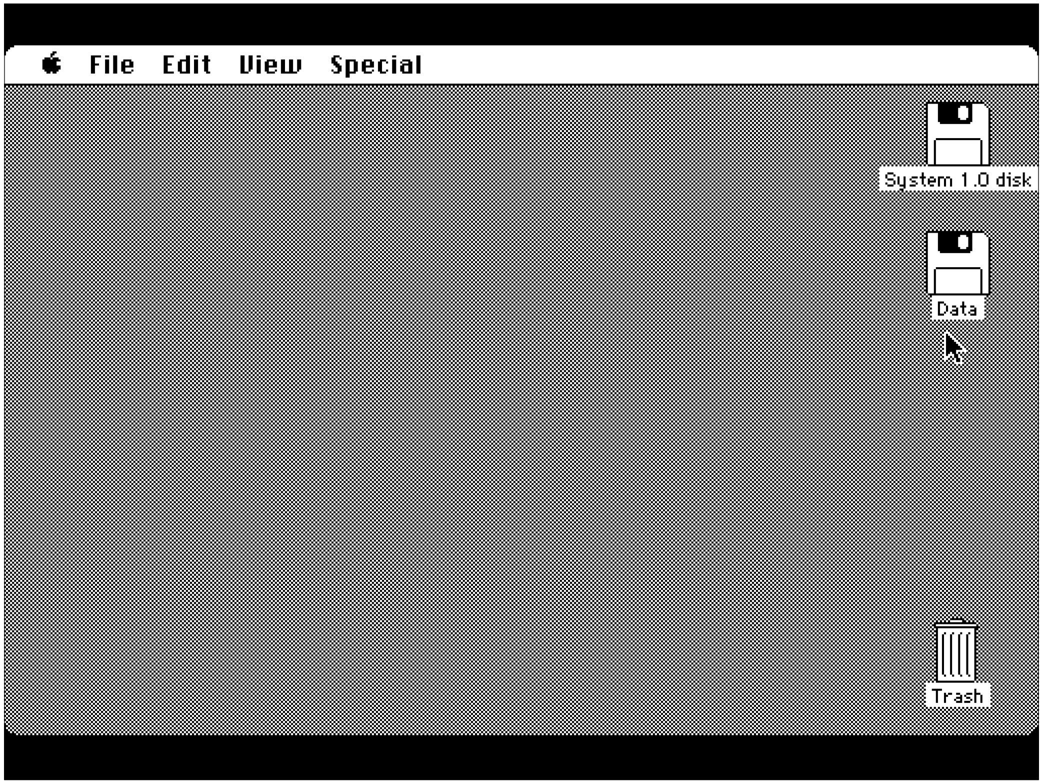
Macintosh File System

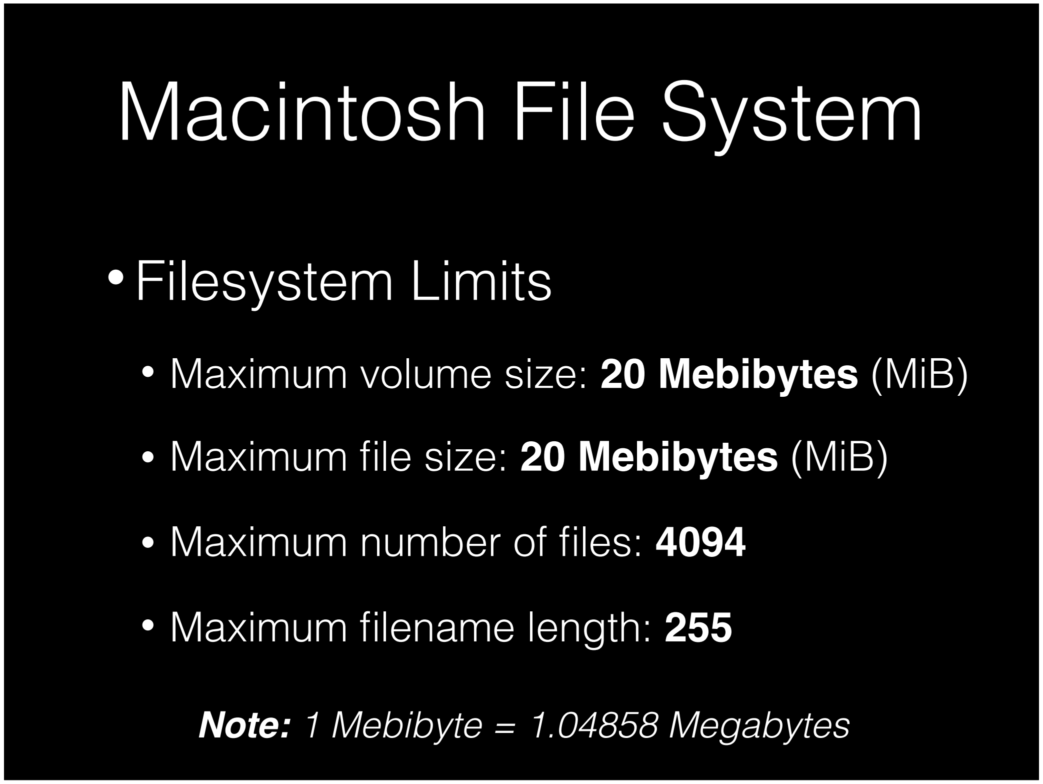


One feature that was different from how Mac filesystems work today was MFS did not support the concept of a hierarchy of directories. Instead, MFS was a flat filesystem where all the files were stored at the same level.

Folders existed as a concept in MFS, but they worked completely differently from the way they do on modern filesystems. The Finder created the illusion of folders which were visible in the Finder but were not visible in the Save and Open dialog boxes of applications. There was always a folder named Empty Folder, which could be used to create new folders in the Finder by altering the existing Empty Folder. As folders were altered by adding files to them or renaming the directory, a new Empty Folder would then appear.

In reality, the filesystem was storing all files using a directory and file handle pairing. To display the contents of a particular folder, MFS would scan the directory for all files which matched the handle and display the results of its search.


File
Edit
View
Special
System 1.0 disk
Data
Trash



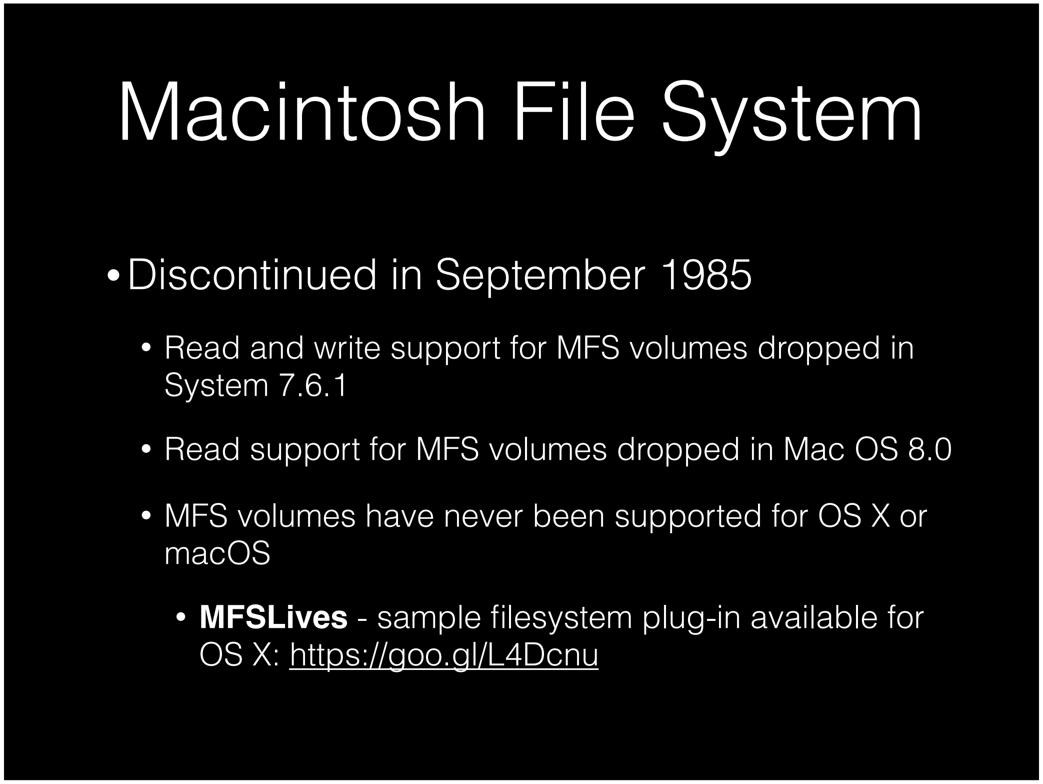
Macintosh File System

- Filesystem Limits
  - Maximum volume size: **20 Mebibytes** (MiB)
  - Maximum file size: **20 Mebibytes** (MiB)
  - Maximum number of files: **4094**
  - Maximum filename length: **255**

Note: *1 Mebibyte = 1.04858 Megabytes*

MFS had the following limitations:

Maximum volume size: 20 Mebibytes (MiB)
Maximum file size: 20 Mebibytes (MiB)
Maximum number of files: 4094
Maximum filename length: 255, although the Finder did not allow names longer than 63 characters to be created.



Macintosh File System

- Discontinued in September 1985
  - Read and write support for MFS volumes dropped in System 7.6.1
  - Read support for MFS volumes dropped in Mac OS 8.0
  - MFS volumes have never been supported for OS X or macOS
    - **MFSLives** - sample filesystem plug-in available for OS X: https://goo.gl/L4Dcnu

MFS was discontinued as the Mac's primary filesystem by Apple in September 1985. Reading and writing to MFS volumes continued to be supported up until System 7.6.1, when Apple dropped support for writing to MFS volumes. All support for MFS was dropped in Mac OS 8.0 and the filesystem was never natively supported on OS X and later. That said, one of Apple's provided example filesystem plug-ins for OS X is named MFSLives and it provides read-only support for MFS volumes.

Hierarchical File System

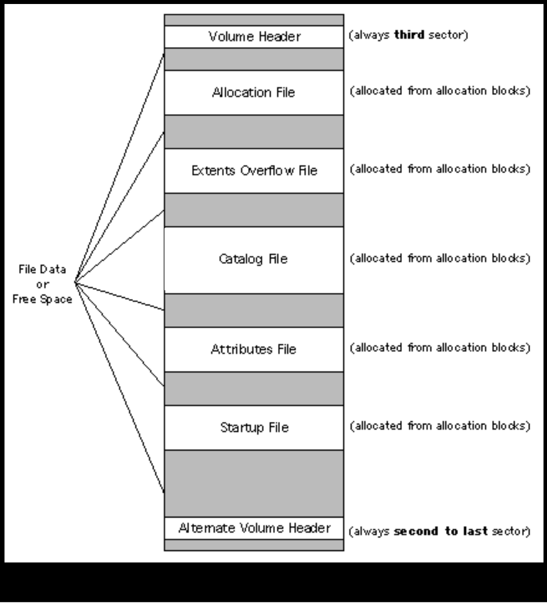
To replace MFS, Apple introduced Hierarchical File System (HFS) in September 1985. In a case of hardware needs driving software design, HFS was introduced to support the Hard Disk 20, Apple's first hard disk drive for the Macintosh.

The reason for the change was that MFS had been optimized to be used on very small and slow media, namely floppy disks.

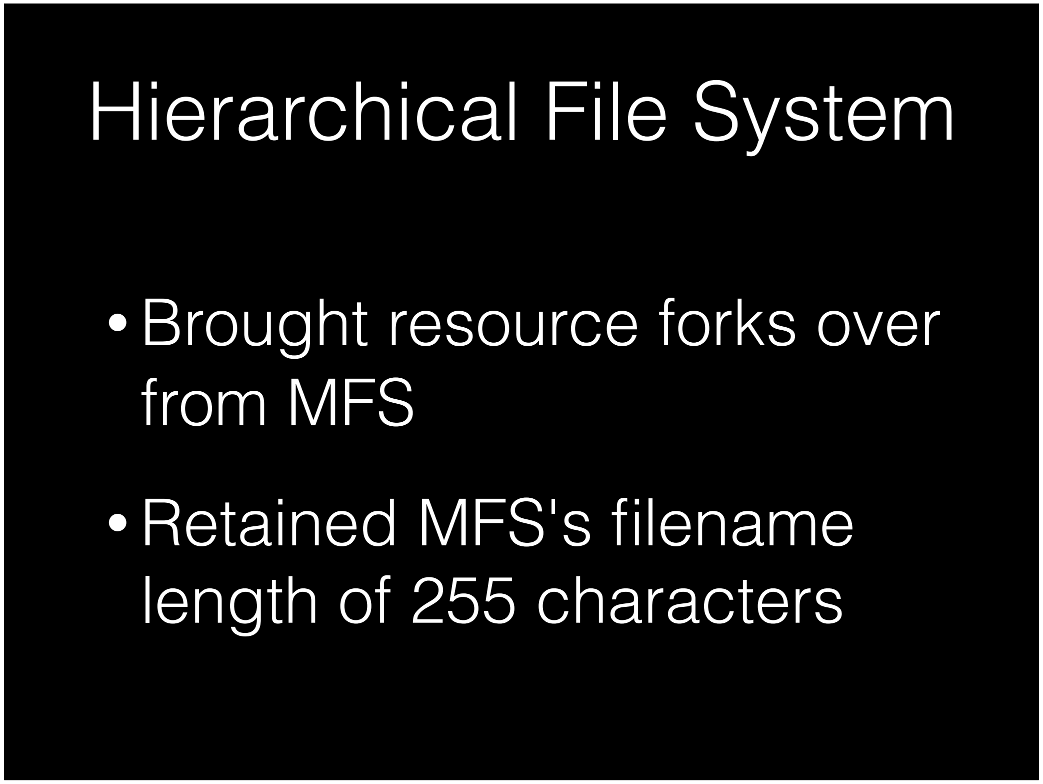
In contrast, the Hard Disk 20 contained a 20 megabyte hard drive, which provided over 50 times the data storage of the stock 400 kB disk drive.

The introduction of larger media like Apple's Hard Disk 20 exposed a scaling issue in MFS which needed to be addressed. MFS's method of displaying files in a folder by searching for directory and file handle pairing worked fine when working with for a few hundred kilobytes of storage and up to a hundred files. When dealing with megabytes of storage and thousands of files, this search method proved to be very inefficient and had a noticeable performance impact.

Hierarchical File System


Volume Header
(always third sector)
Allocation File
(allocated from allocation blocks)
Extents Overflow File
(allocated from allocation blocks)
File Data
or
Free Space
Catalog File
(allocated from allocation blocks)
Attributes File
(allocated from allocation blocks)
Startup File
(allocated from allocation blocks)
Alternate Volume Header
(always second to last sector)

Apple's solution to this problem was to replace MFS's flat file structure with a directory-based file system which leveraged a new file known as the Catalog File. The Catalog File incorporated a binary search tree structure, also known as a B-Tree, which could be searched quickly regardless of the size of the storage volume.

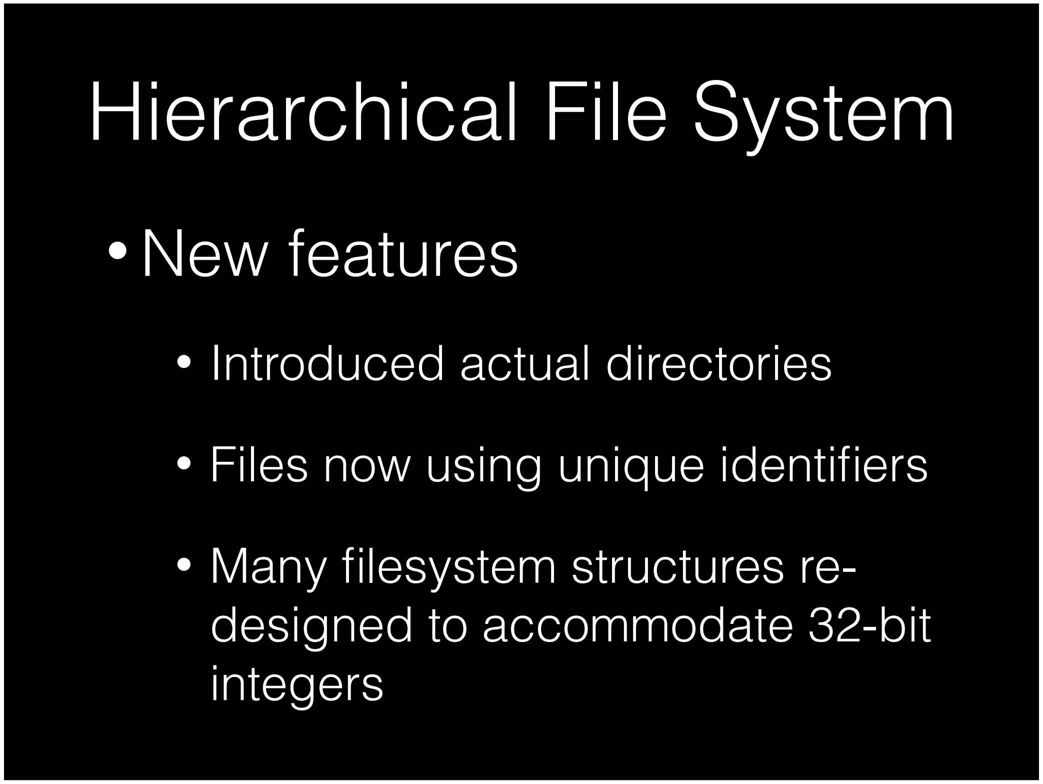


Hierarchical File System

- Brought resource forks over from MFS
- Retained MFS's filename length of 255 characters

HFS incorporated a number of features first introduced in MFS, including:

- The use of resource forks
- Having a maximum filename length of 255 characters, although the Finder did not allow names longer than 31 characters to be created. This 31 character limit is shorter than MFS's 63 characters but this was still a Finder limitation rather than an HFS limitation.

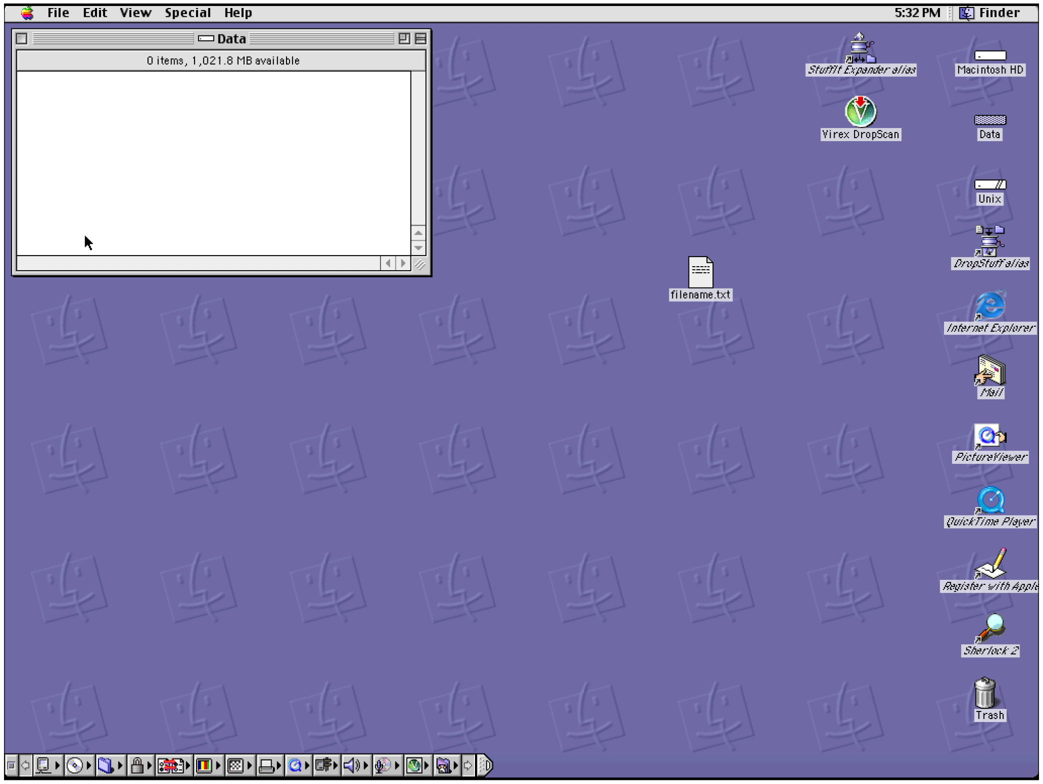


Hierarchical File System

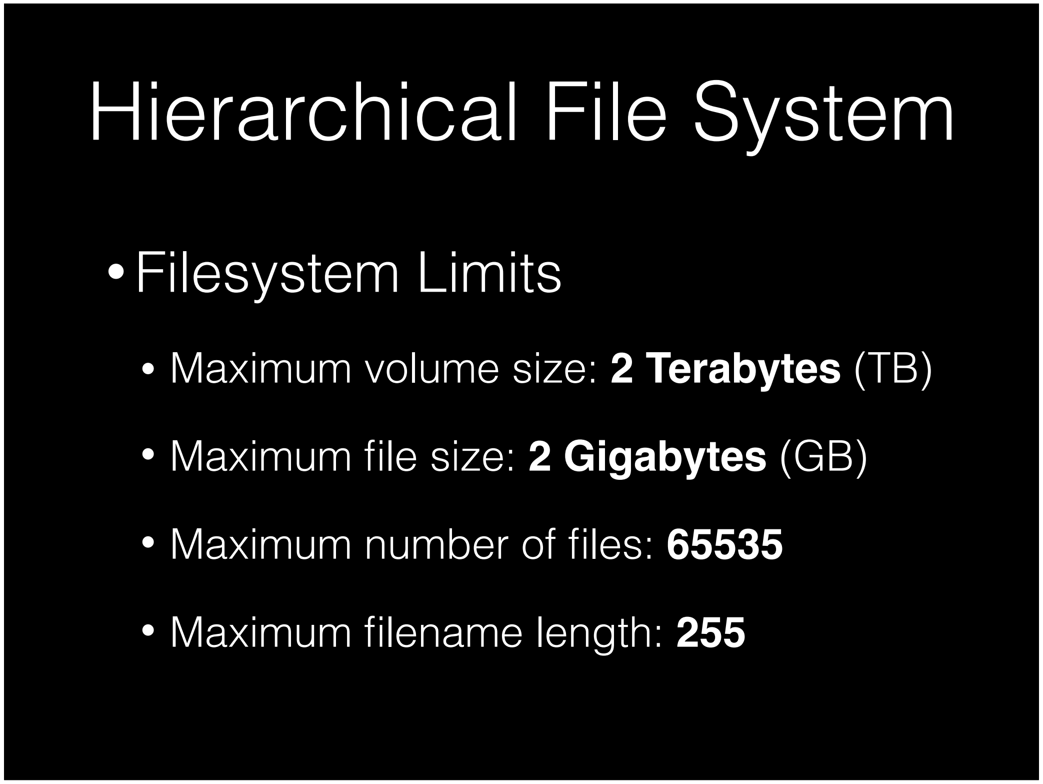
- New features
  - Introduced actual directories
  - Files now using unique identifiers
  - Many filesystem structures re-designed to accommodate 32-bit integers

HFS also introduced new features, including actual directories stored in a hierarchy. Files were now also referenced by the filesystem with unique file identifiers, rather than by the file names, which allowed files to be named almost anything without losing track of which application was supposed to be able to open them.

The designers of HFS also re-developed various filesystem structures in order to be able to handle larger numbers, with most being upgraded from MFS's 16-bit integers to 32-bit. That said, one place where upsizing did not take place was in the number of files supported. That remained a 16-bit integer, with HFS being able to support up to 65,535 files.



0 items, 1,021.8 MB available

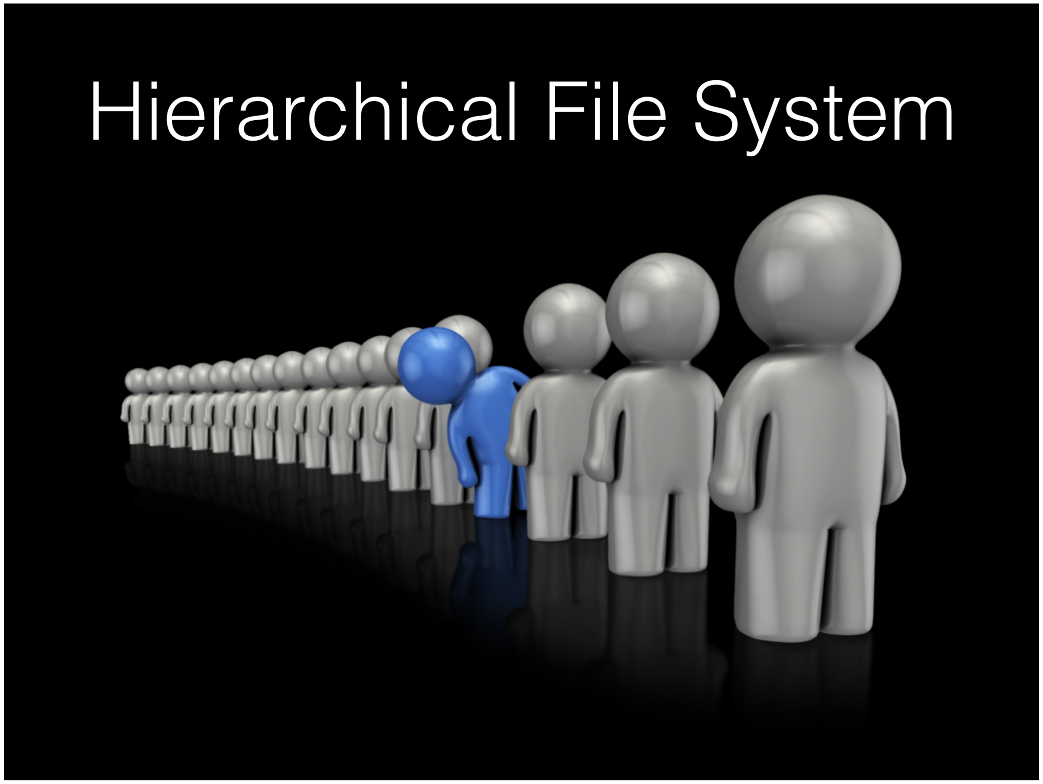


Hierarchical File System

- Filesystem Limits
  - Maximum volume size: **2 Terabytes** (TB)
  - Maximum file size: **2 Gigabytes** (GB)
  - Maximum number of files: **65535**
  - Maximum filename length: **255**

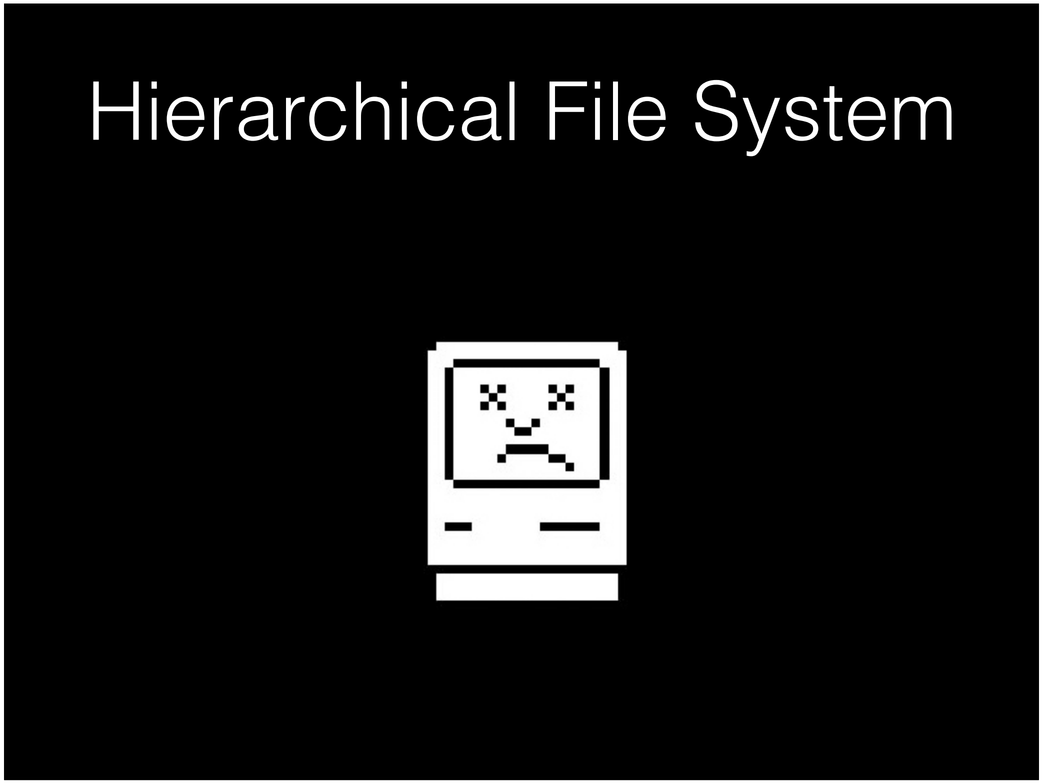
HFS had the following limitations:

Maximum volume size: 2 Terabytes
Maximum file size: 2 Gigabytes
Maximum number of files: 65535
Maximum filename length: 255, although the Finder disallowed names more than 31 characters long.



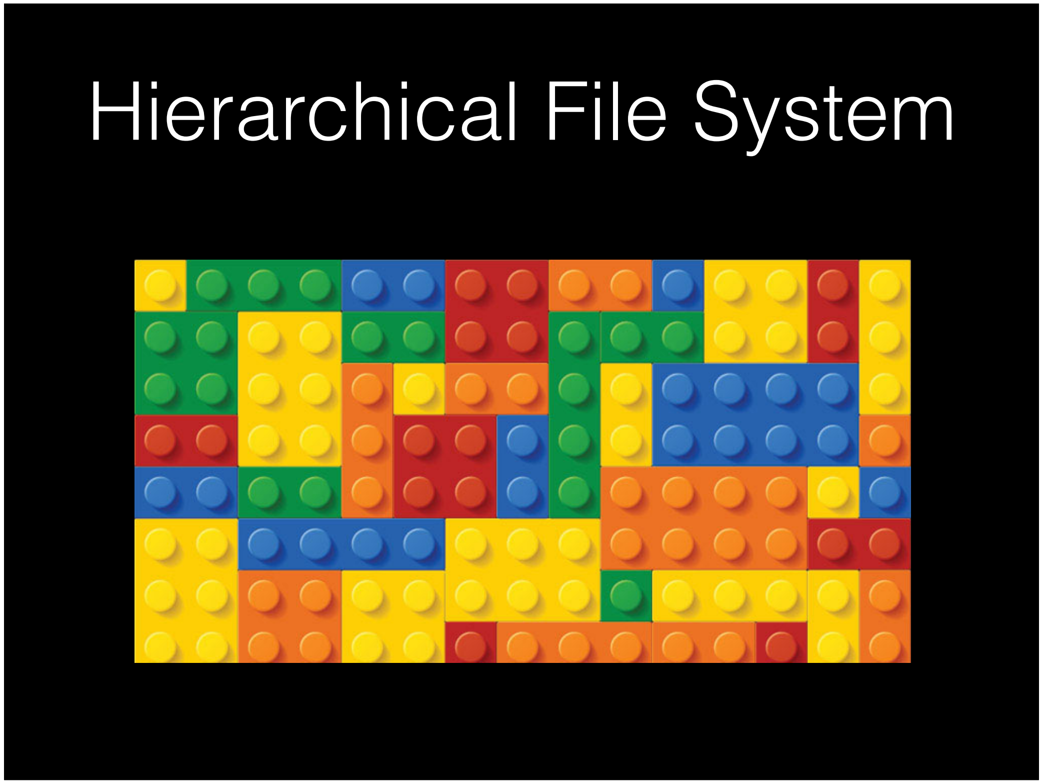
Hierarchical File System

Along solving MFS's issues, HFS had a few of its own. The Catalog File, the replacement for MFS's directory file handle pairing, stores all the file and directory records in a single data structure. This can result in performance problems when multitasking because only one program at a time can write to this structure. If an application is "hogging" the system, other programs are waiting for their turn to write their data to the Catalog File.



Hierarchical File System

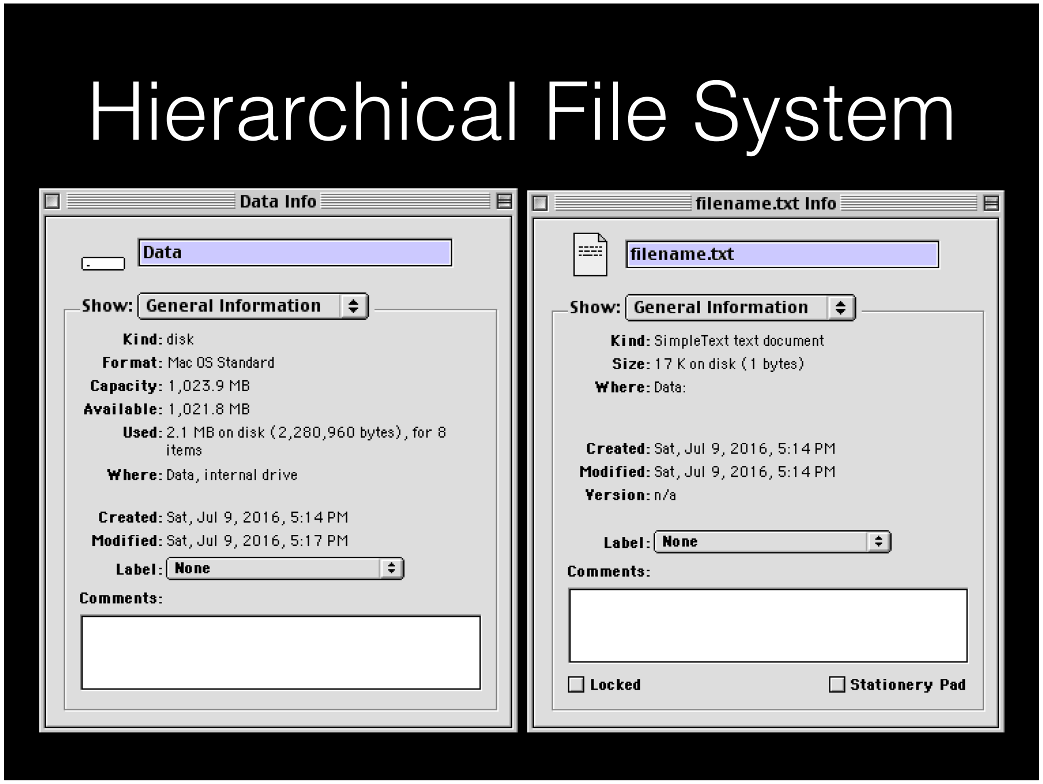
Damage to the Catalog File can also result in catastrophic file system damage because of the Catalog File's role of being the only place where all of the file and directory records are stored.



Hierarchical File System

Another issue with HFS had to do with block allocation. A storage volume is inherently divided into logical blocks of 512 bytes. HFS groups these logical blocks into allocation blocks, which can contain one or more logical blocks, depending on the total size of the volume. HFS uses a 16-bit value to address allocation blocks, limiting the number of allocation blocks to 65,535.

HFS could support a maximum of 65,535 files with each file being assigned at least one allocation block on the file system, regardless of the size of the storage volume. This maximum limit also introduced a minimum file size limit as a logical consequence, where any file could not be smaller than one allocation block or 1/65535 of the size of the disk.



Hierarchical File System

Data

Show: General Information

Kind: disk
Format: Mac OS Standard
Capacity: 1,023.9 MB
Available: 1,021.8 MB
Used: 2.1 MB on disk (2,280,960 bytes), for 8 items
Where: Data, internal drive

Created: Sat, Jul 9, 2016, 5:14 PM
Modified: Sat, Jul 9, 2016, 5:17 PM
Label: None
Comments:

filename.txt

Show: General Information

Kind: SimpleText text document
Size: 17 K on disk (1 bytes)
Where: Data:

Created: Sat, Jul 9, 2016, 5:14 PM
Modified: Sat, Jul 9, 2016, 5:14 PM
Version: n/a

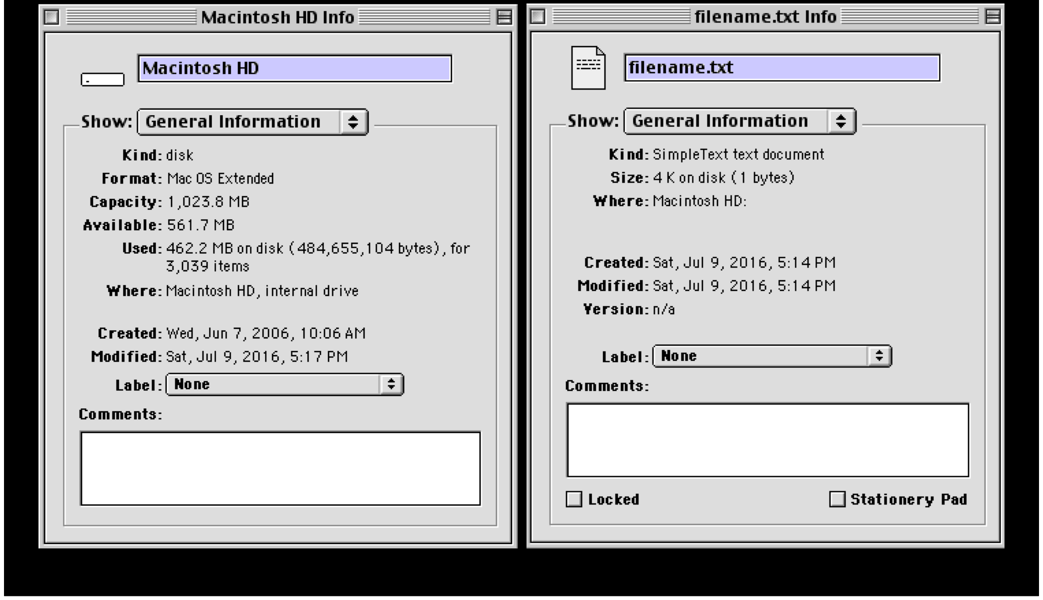
Label: None
Comments:

□ Locked □ Stationery Pad

When available drives were well under 1 gigabyte in size, this was not a particular problem, but as drives got larger, the smallest amount of space that any file could occupy became excessively large. To illustrate the issue, let's look at how HFS operates on a 1 gigabyte disk. 1 GB divided by 65,535 ~= 16 kilobytes. That meant that the minimum file size of any file on a 1 GB drive using HFS was effectively 16 KB, regardless of the file's actual size. That meant even a a one-byte file would take up a minimum of 16 KB of disk space. For users with a lot of small files, this meant that a lot of disk space could be lost to this minimum file size limitation.

To fix this storage allocation issue and to add other improvements, Apple introduced another filesystem in 1998.

HFS Plus



Macintosh HD Info

Macintosh HD

Show: General Information

Kind: disk
Format: Mac OS Extended
Capacity: 1,023.8 MB
Available: 561.7 MB
Used: 462.2 MB on disk (484,655,104 bytes), for 3,039 items
Where: Macintosh HD, internal drive

Created: Wed, Jun 7, 2006, 10:06 AM
Modified: Sat, Jul 9, 2016, 5:17 PM
Label: None
Comments:

filename.txt Info

filename.txt

Show: General Information

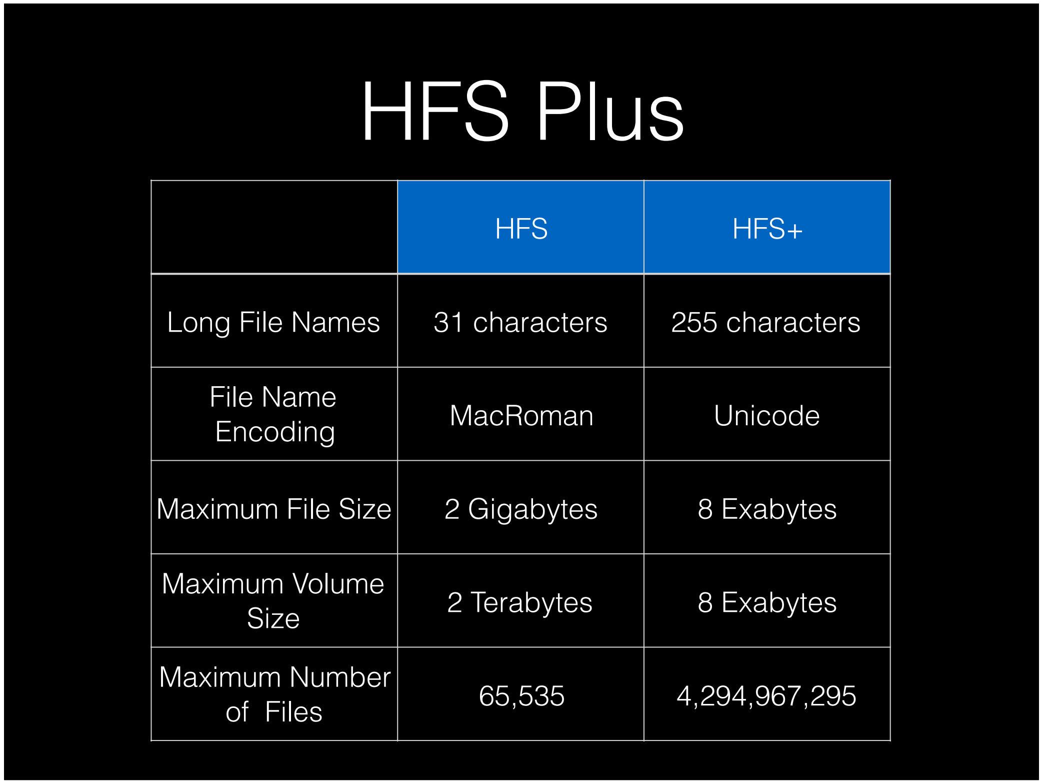
Kind: SimpleText text document
Size: 4 K on disk (1 bytes)
Where: Macintosh HD:

Created: Sat, Jul 9, 2016, 5:14 PM
Modified: Sat, Jul 9, 2016, 5:14 PM
Version: n/a

Label: None
Comments:

□ Locked □ Stationery Pad

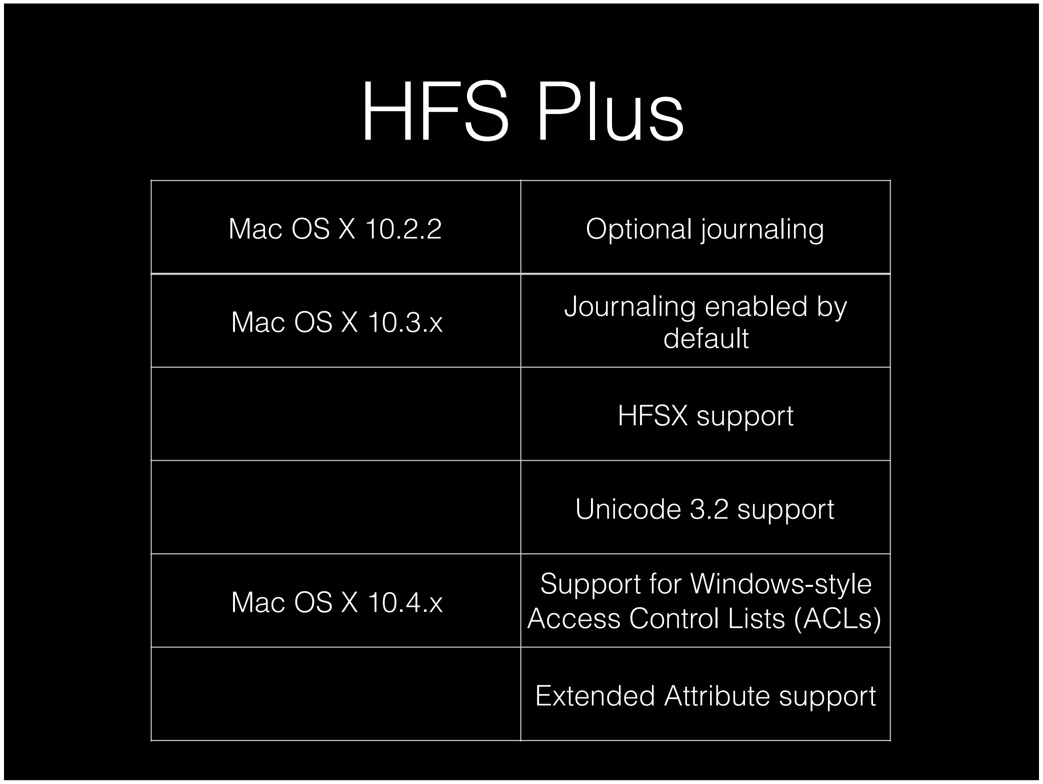
The HFS Plus filesystem was introduced with Mac OS 8.1 in 1998 and was designed to fix the block allocation issue in HFS. This was accomplished by upgrading block allocation to use 32-bit integers in place of HFS's 16-bit, which allowed for a much lower minimum file size.



HFS Plus

	HFS	HFS+
Long File Names	31 characters	255 characters
File Name Encoding	MacRoman	Unicode
Maximum File Size	2 Gigabytes	8 Exabytes
Maximum Volume Size	2 Terabytes	8 Exabytes
Maximum Number of Files	65,535	4,294,967,295

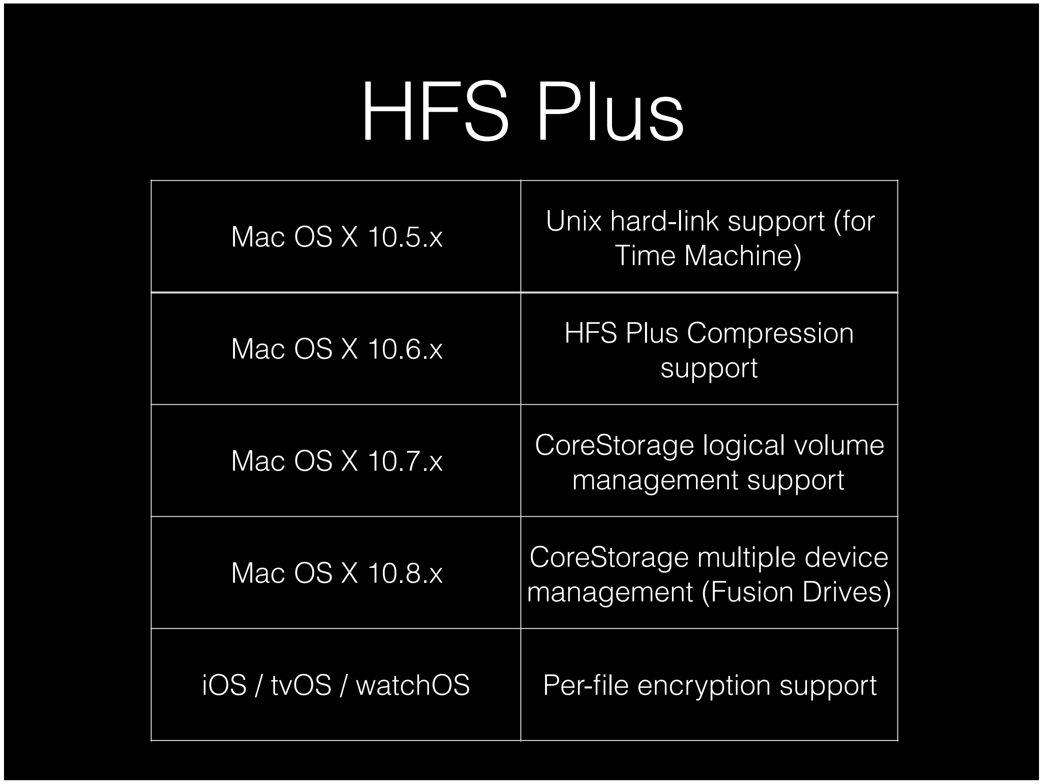
Along with the block allocation issue, there were a number of other issues in HFS which Apple addressed as part of creating HFS Plus. These changes included implementing Unicode support and addressing the issue of long file names, so that you could actually use 255 characters in a file name.



HFS Plus

Mac OS X 10.2.2	Optional journaling
Mac OS X 10.3.x	Journaling enabled by default
	HFSX support
	Unicode 3.2 support
Mac OS X 10.4.x	Support for Windows-style Access Control Lists (ACLs)
	Extended Attribute support

Over the years, Apple has added new features to HFS Plus.



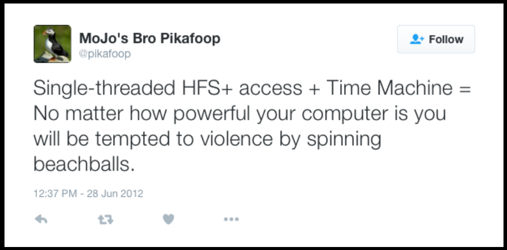
HFS Plus

Mac OS X 10.5.x	Unix hard-link support (for Time Machine)
Mac OS X 10.6.x	HFS Plus Compression support
Mac OS X 10.7.x	CoreStorage logical volume management support
Mac OS X 10.8.x	CoreStorage multiple device management (Fusion Drives)
iOS / tvOS / watchOS	Per-file encryption support

HFS Plus is not only used on Macs. iOS / tvOS / watchOS were also using it until this year, with HFS Plus being extended to add support for per-file encryption.

HFS Plus

- Limitations
  - Unable to checksum metadata
  - 1 second timestamp granularity
  - Single-threaded access to filesystem



MoJo's Bro Pikafoop
@pikafoop

Single-threaded HFS+ access + Time Machine = No matter how powerful your computer is you will be tempted to violence by spinning beachballs.

12:37 PM - 28 Jun 2012

HFS Plus first shipped in 1998, 19 years ago. Since that time, filesystems on other platforms have shipped which include features which are not present in HFS Plus.

- Data checksums

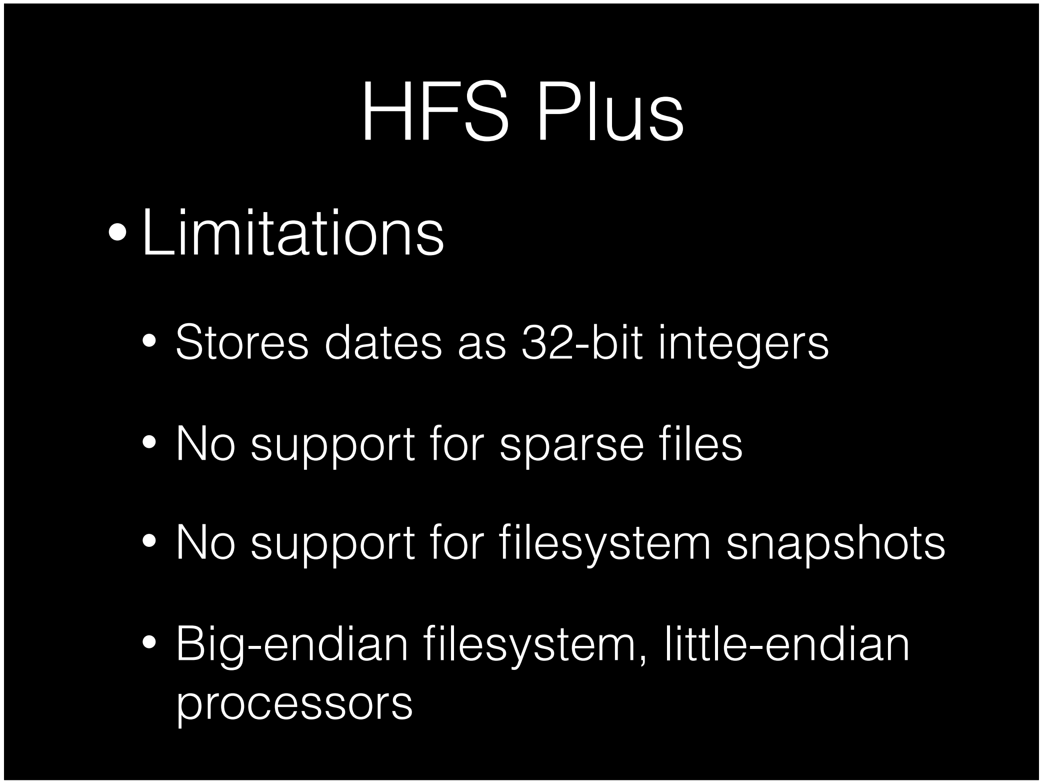
HFS Plus does not include the ability to checksum metadata, which helps to detect file corruption.

- Timestamp granularity

HFS Plus has 1-second time stamp granularity, which means that reads and writes to the file system can only be tracked down to the second. Other filesystems are able to track changes with nanosecond granularity, which allows reads and write to be managed better.

- Concurrent access to the filesystem

In an issue which carried over from HFS, HFS Plus is a single-threaded filesystem, which means that only one process can



HFS Plus

- Limitations
  - Stores dates as 32-bit integers
  - No support for sparse files
  - No support for filesystem snapshots
  - Big-endian filesystem, little-endian processors

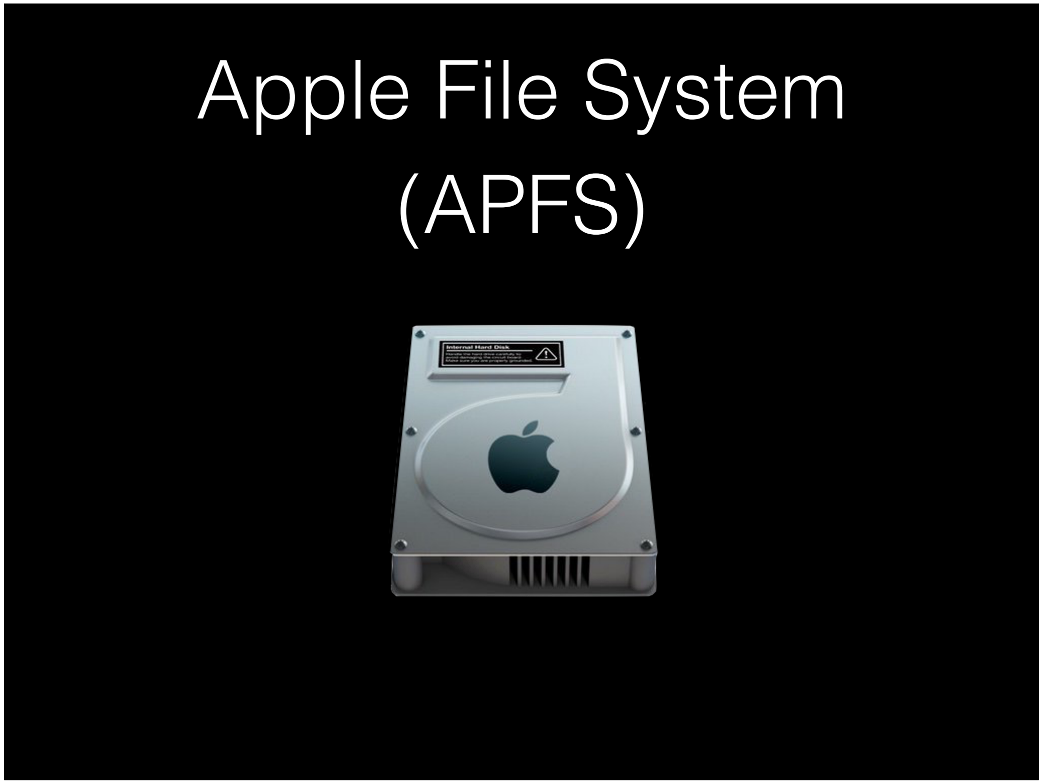
Other limitations include:

- Date limitations: HFS Plus stores dates in 32-bit integers containing the number of seconds since midnight, January 1, 1904, GMT. The consequence is that HFS Plus's maximum representable date is February 6, 2040 at 06:28:15 GMT.

- Sparse file support: HFS Plus does not support sparse files, which are a type of computer file which attempts to use file system space more efficiently by writing reference metadata for the empty space in files. HFS Plus instead allocates space on the filesystem for the empty space in the file.

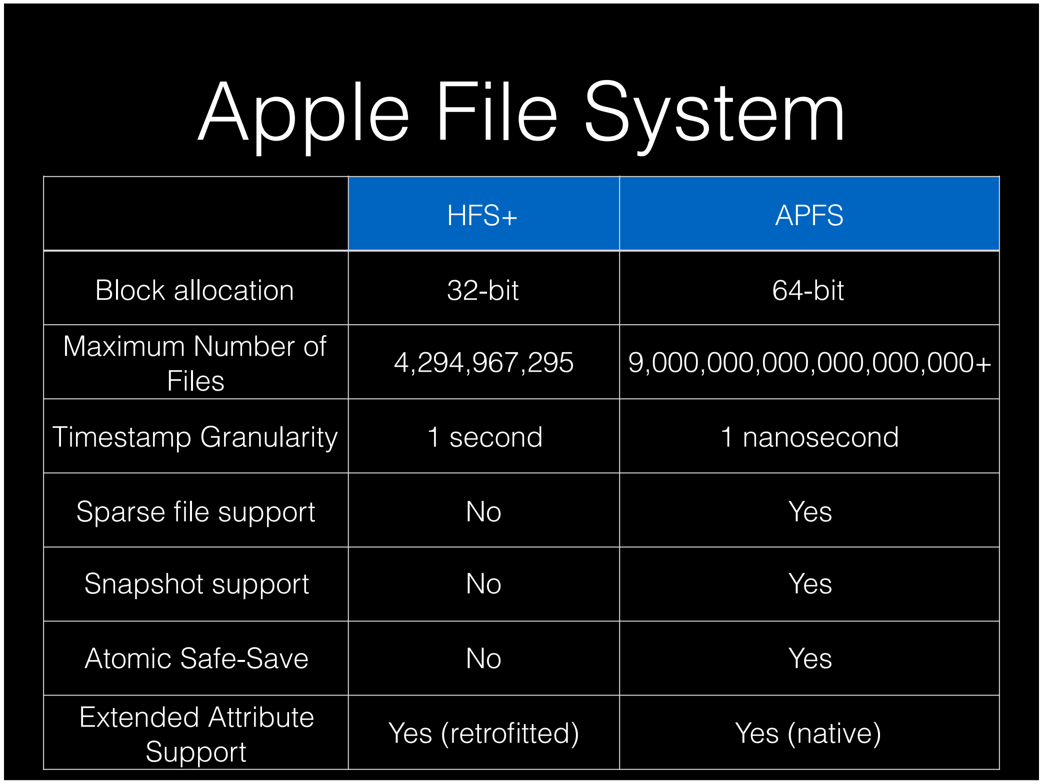
- Snapshot support: HFS Plus does not support capturing filesystem snapshots, which means creating a read-only copy of the state which the filesystem was in at a specified point in time.

Another legacy issue is that HFS Plus was not designed for use with Intel processors. Motorola and PowerPC processors were the order of the day for 90s Macs. The reason this matters is that those processors were big-endian and the expected transmission order of bytes is set in one specific way. However, Intel processors are little-endian and the expected transmission



Apple File System (APFS)

With HFS Plus showing its age and its legacy roots, Apple has made the judgement that continuing to maintain and evolve this 19 year old file system is no longer tenable. Apple needs a new filesystem and Apple File System is being born from that need.



Apple File System

	HFS+	APFS
Block allocation	32-bit	64-bit
Maximum Number of Files	4,294,967,295	9,000,000,000,000,000,000+
Timestamp Granularity	1 second	1 nanosecond
Sparse file support	No	Yes
Snapshot support	No	Yes
Atomic Safe-Save	No	Yes
Extended Attribute Support	Yes (retrofitted)	Yes (native)

APFS includes a number of features that are currently not available in HFS Plus:

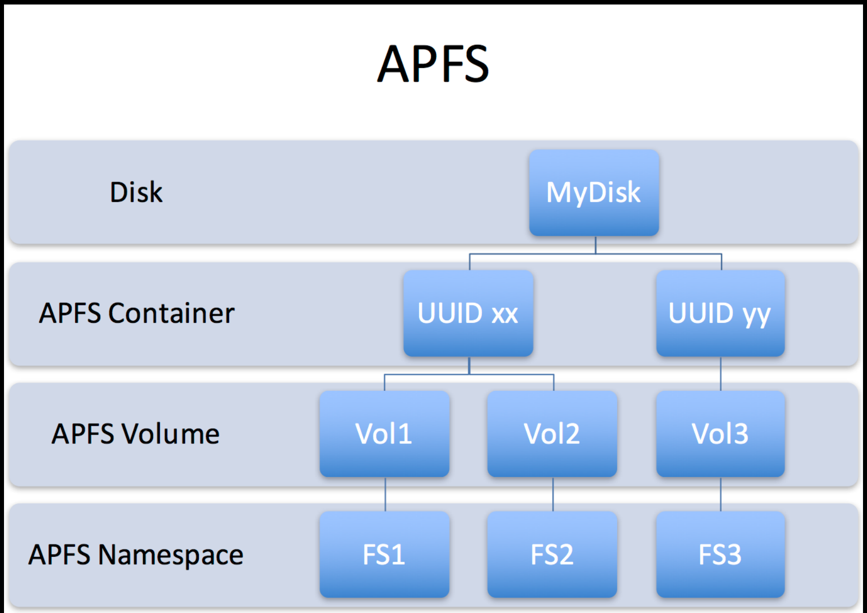
64-bit block allocation: APFS improved on HFS Plus 32-bit support for block allocation by upgrading to 64-bit. This allows APFS to support over 9 quintillion files on a single APFS volume, in place of HFS Plus's 4 billion files on a single volume.

Nanosecond time stamps: APFS supports 1 nanosecond timestamp granularity, which improves on HFS Plus's 1-second time stamp granularity.

Sparse file support: APFS supports the use of sparse files, which allows APFS to handle empty space in files more efficiently than HFS Plus can.

Snapshot support: APFS includes support for capturing filesystem snapshots. Those snapshots are also able to be mounted as read-only volumes using the mount_apfs command.

Atomic Safe-Save: This capability performs saves in a single transaction that, from the user's perspective, either complete successfully or the save does not happen. This addresses the problem of a file write only partially completing, which may cause

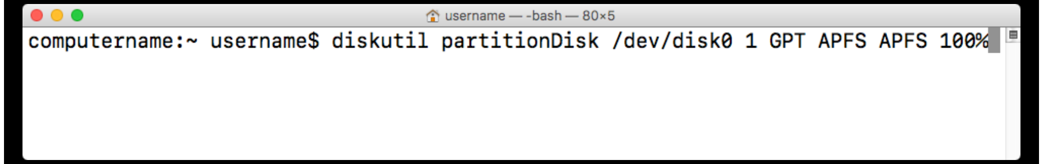


Graphic by Stephen Foskett - http://goo.gl/bmRXFU

APFS is structured as shown, with containers being the base storage unit of APFS. Containers are pools of storage, which are conceptually similar to CoreStorage's logical volume groups.

In turn, the containers host APFS volumes which are the file systems. Each volume then generally maps to a matching namespace, which Apple is defining as meaning sets of files and directories.

Apple File System

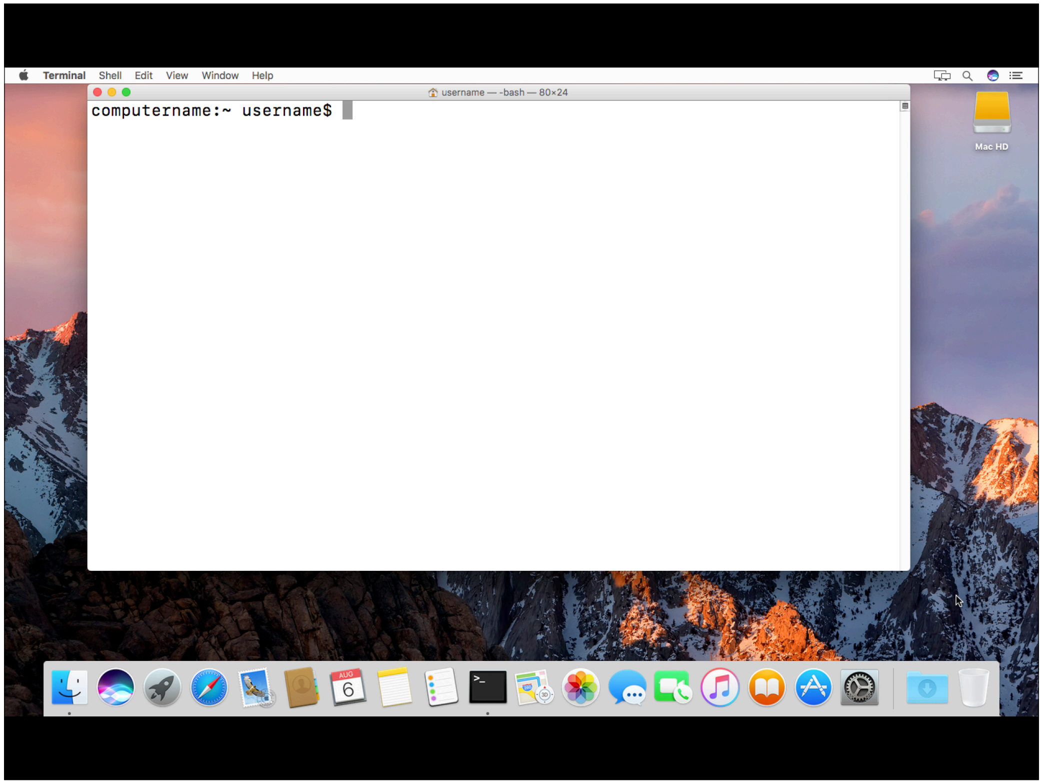


```
computername:~ username$ diskutil partitionDisk /dev/disk0 1 GPT APFS APFS 100%
```

To partition a disk and set up
one APFS container and one APFS volume:

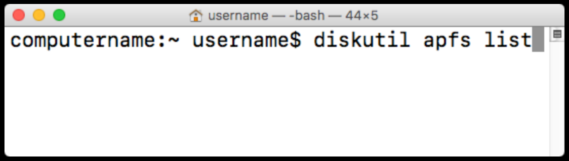
diskutil partitionDisk device_ID_here 1 GPT APFS APFS_volume_name_here 100%

While full support for managing APFS is not available at this time, it is possible to create containers, volumes and namespaces with diskutil. Let’s take a look at how this works, beginning with formatting a disk to use APFS and set up one container.



```
computername:~ username$ 
```

Apple File System

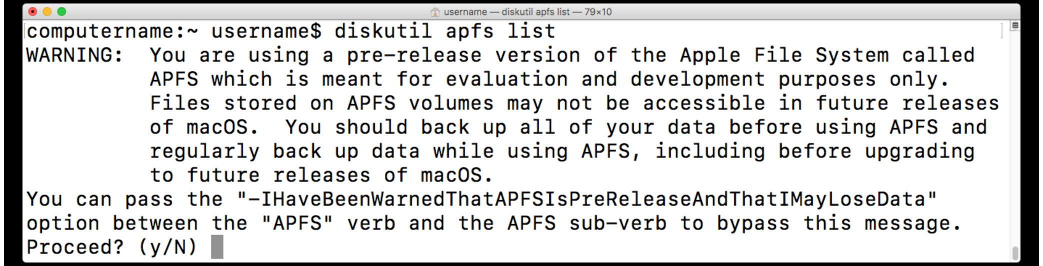


```
computername:~ username$ diskutil apfs list
```

To list available APFS objects:
diskutil apfs list

To do further work with APFS volumes, you'll need to use diskutil's new apfs functions in Sierra and later. To list available APFS objects, including containers and volumes, use diskutil apfs list.

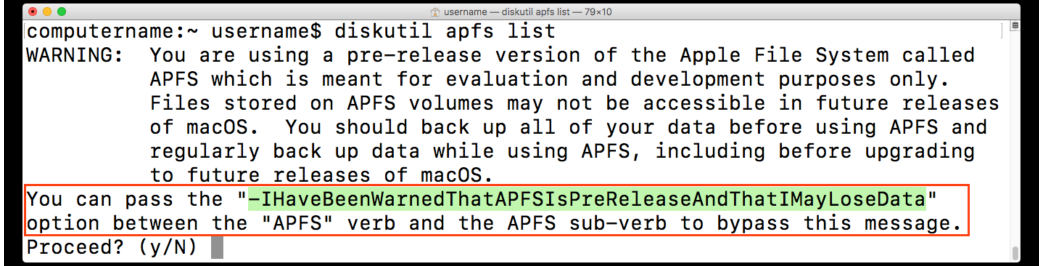
Apple File System



```
computername:~ username$ diskutil apfs list
WARNING:  You are using a pre-release version of the Apple File System called
          APFS which is meant for evaluation and development purposes only.
          Files stored on APFS volumes may not be accessible in future releases
          of macOS.  You should back up all of your data before using APFS and
          regularly back up data while using APFS, including before upgrading
          to future releases of macOS.
You can pass the "-IHaveBeenWarnedThatAPFSIsPreReleaseAndThatIMayLoseData"
option between the "APFS" verb and the APFS sub-verb to bypass this message.
Proceed? (y/N)
```

Apple never wants you to forget that APFS is still a work in progress, so anytime you run diskutil apfs commands, this warning will appear.

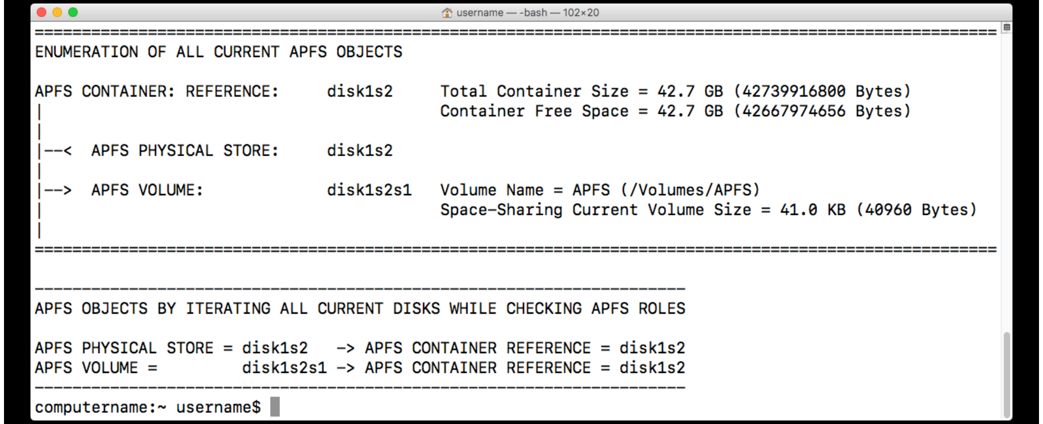
Apple File System



```
computername:~ username$ diskutil apfs list
WARNING:  You are using a pre-release version of the Apple File System called
          APFS which is meant for evaluation and development purposes only.
          Files stored on APFS volumes may not be accessible in future releases
          of macOS.  You should back up all of your data before using APFS and
          regularly back up data while using APFS, including before upgrading
          to future releases of macOS.
You can pass the "-IHaveBeenWarnedThatAPFSIsPreReleaseAndThatIMayLoseData"
option between the "APFS" verb and the APFS sub-verb to bypass this message.
Proceed? (y/N)
```

Apple recognized that folks would likely want to skip this message if possible, so they did include a command-line option for bypassing it. However, the bypass option makes perfectly clear Apple’s guidance with regards to storing your data on APFS volumes at this time.

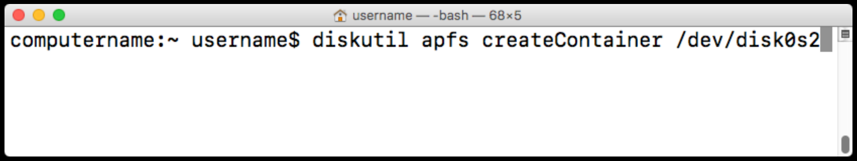
Apple File System



```
======================================================================================================
ENUMERATION OF ALL CURRENT APFS OBJECTS

APFS CONTAINER: REFERENCE:     disk1s2     Total Container Size = 42.7 GB (42739916800 Bytes)
|                                          Container Free Space = 42.7 GB (42667974656 Bytes)
|
|--<  APFS PHYSICAL STORE:     disk1s2
|
|-->  APFS VOLUME:             disk1s2s1   Volume Name = APFS (/Volumes/APFS)
|                                          Space-Sharing Current Volume Size = 41.0 KB (40960 Bytes)
|
======================================================================================================


---------------------------------------------------------------------
APFS OBJECTS BY ITERATING ALL CURRENT DISKS WHILE CHECKING APFS ROLES

APFS PHYSICAL STORE = disk1s2   -> APFS CONTAINER REFERENCE = disk1s2
APFS VOLUME =         disk1s2s1 -> APFS CONTAINER REFERENCE = disk1s2
---------------------------------------------------------------------
computername:~ username$ 
```

Once you’re past the APFS warning, the output of diskutil apfs list should look similar to this. In this case, there is one APFS container and one APFS volume present on this Mac.

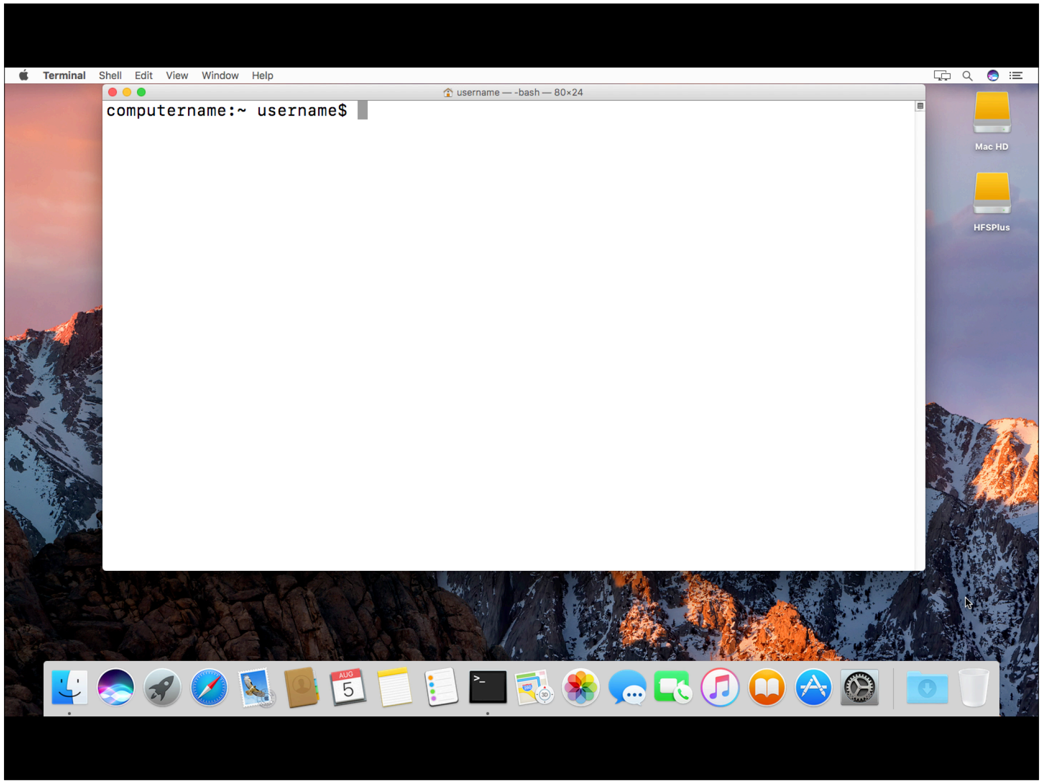
Apple File System



```
computername:~ username$ diskutil apfs createContainer /dev/disk0s2
```

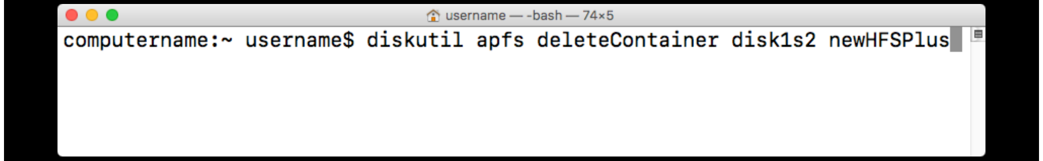
To create a new container on a physical device:
diskutil apfs createContainer /dev/<devicename>

If you have an existing APFS-formatted drive and you want to add additional containers to it, you can use the createContainer command. This command will also convert an HFS Plus drive to APFS and set up an empty container.



```
computername:~ username$ 
```

Apple File System

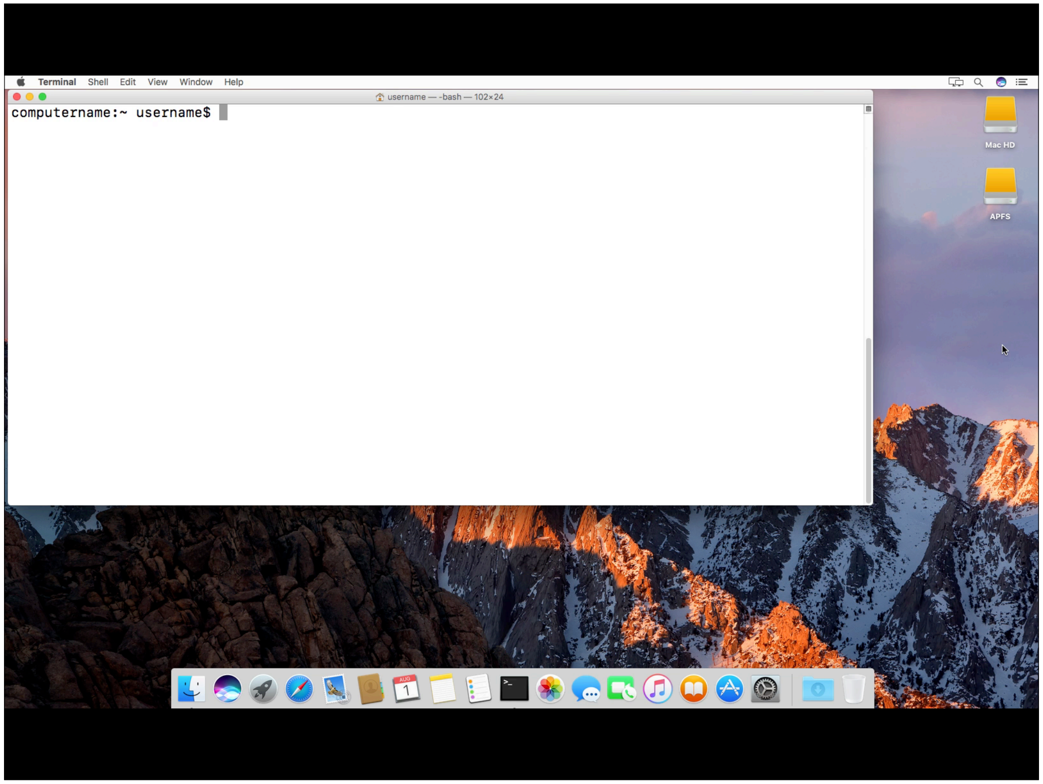


```
computername:~ username$ diskutil apfs deleteContainer disk1s2 newHFSPlus
```

To remove a container on a physical device:
diskutil apfs deleteContainer container_device_here new_name

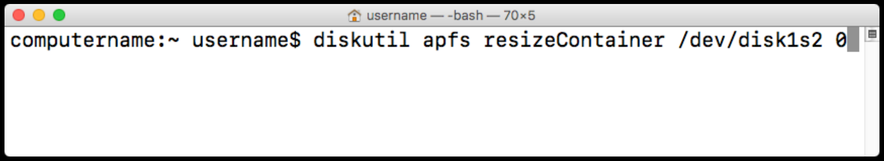
To destroy an existing APFS Container and all volumes and namespaces on it, you can use the the deleteContainer command. The APFS volumes are unmounted and all APFS volumes are deleted along with their parent container.

The APFS Container's former physical disks will be reformatted as HFS Plus as part of this process and you can optionally specify a name for the HFS Plus volume. If no name is chosen, the new HFS Plus volume will be named "Untitled".



```
computername:~ username$
```

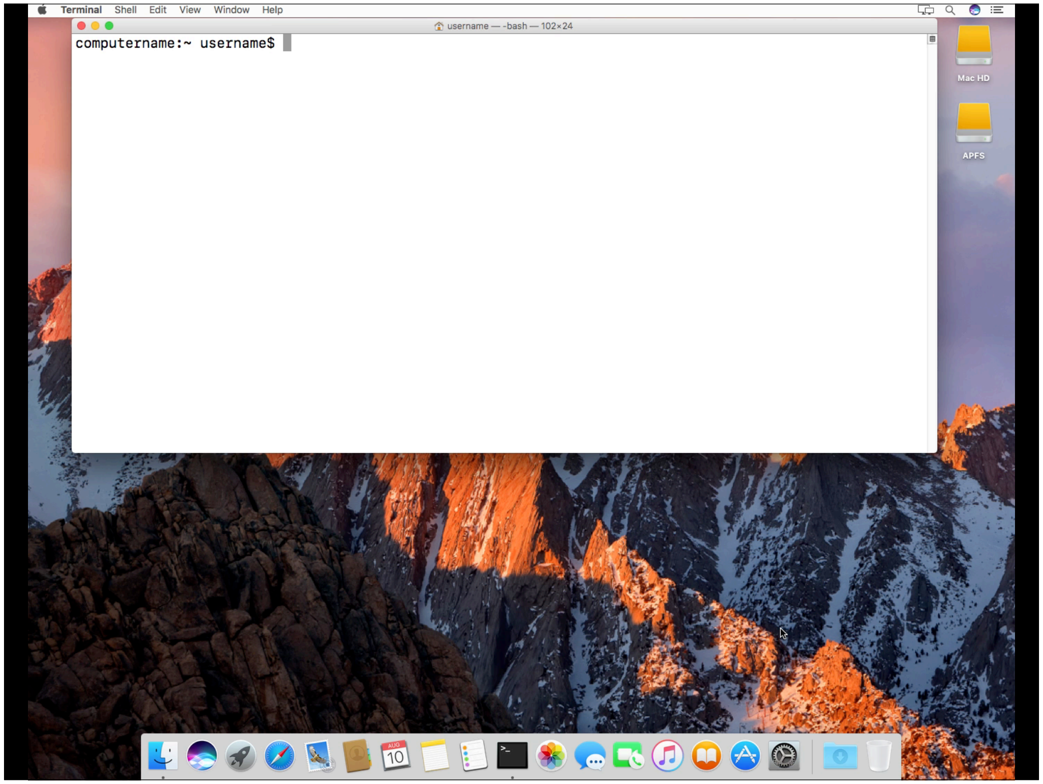
Apple File System



```
computername:~ username$ diskutil apfs resizeContainer /dev/disk1s2 0
```

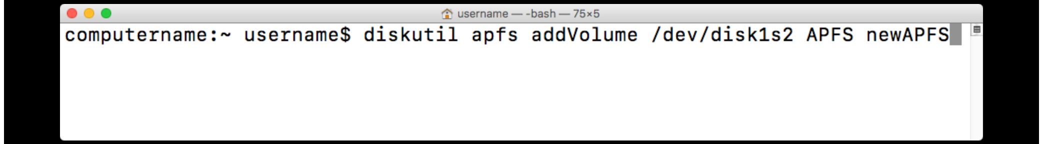
To remove a container on a physical device:
diskutil apfs resizeContainer container_device_here new_size

To grow a container, use the resizeContainer command. This allows the option of growing a container's available allocated space when more physical drive space is available. If you want to have your container resized to fill all available space, you would use the number zero for the size.



```
computername:~ username$ 
```

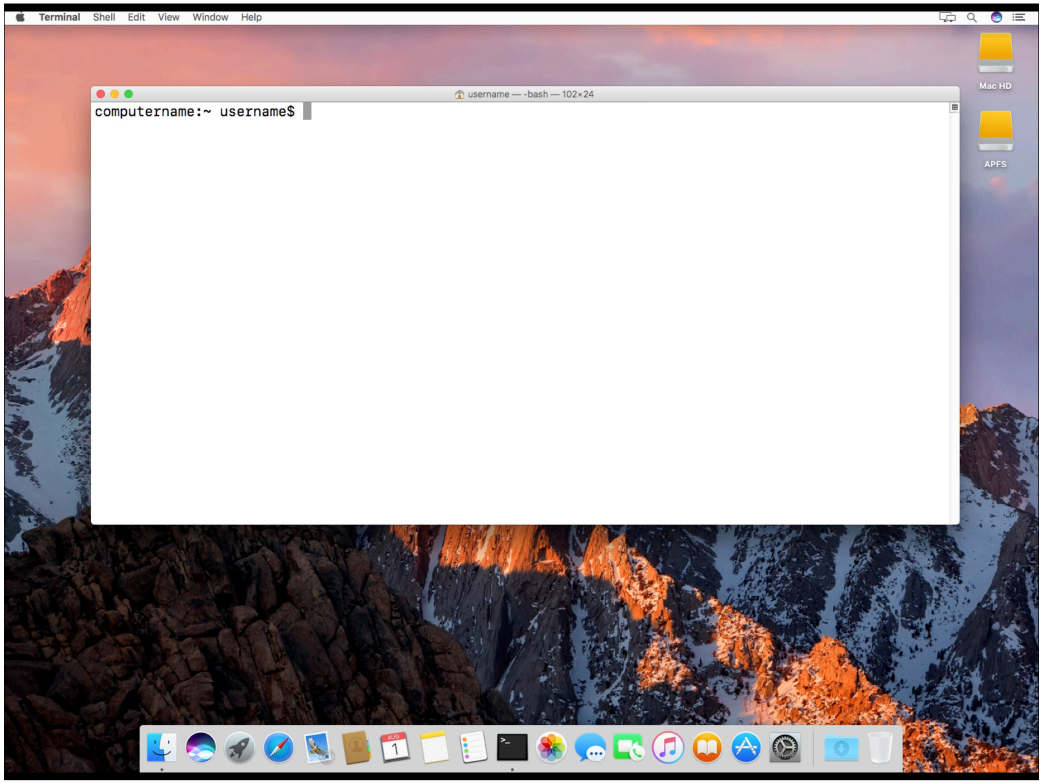
Apple File System



```
computername:~ username$ diskutil apfs addVolume /dev/disk1s2 APFS newAPFS
```

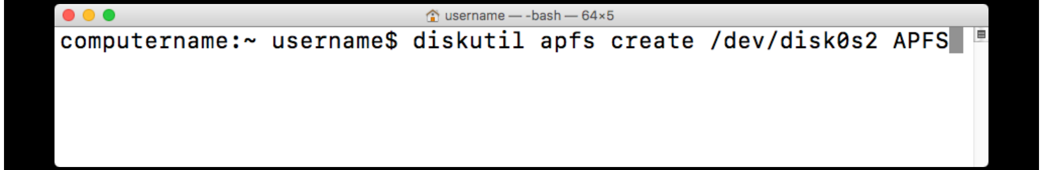
To add a new volume to a container:
diskutil apfs addVolume <devicename> APFS <volumename>

If you have an existing container and you want to add volumes to it, you can use the addVolume command.



```
computername:~ username$ 
```

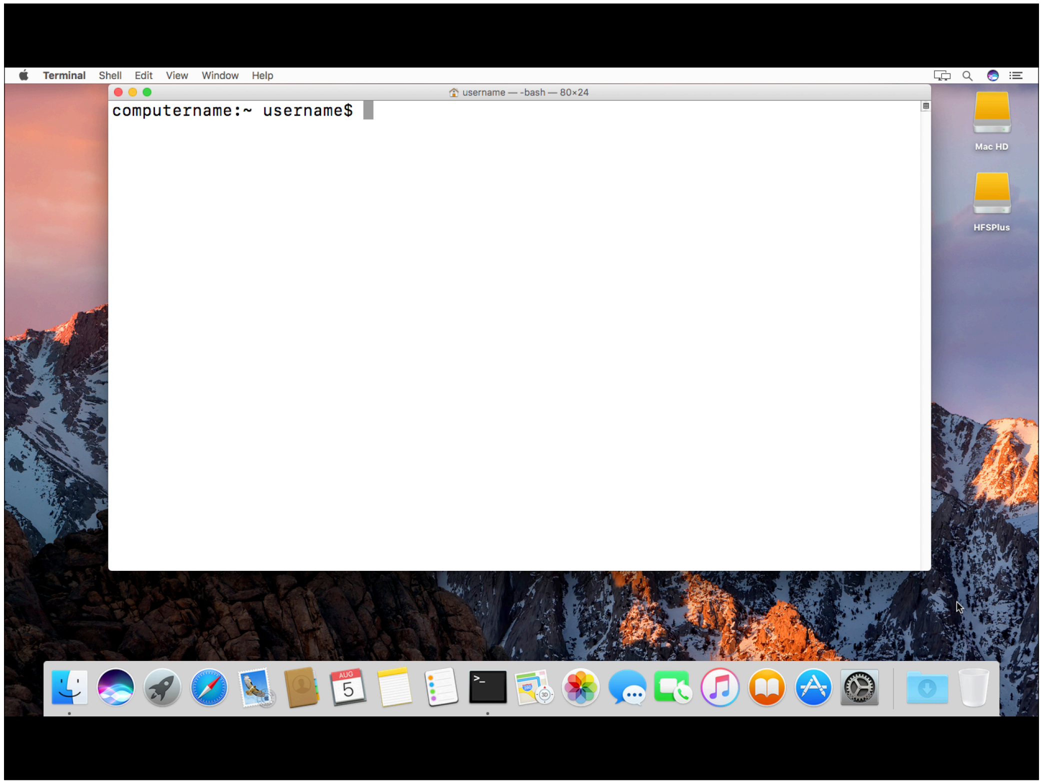
Apple File System



```
computername:~ username$ diskutil apfs create /dev/disk0s2 APFS
```

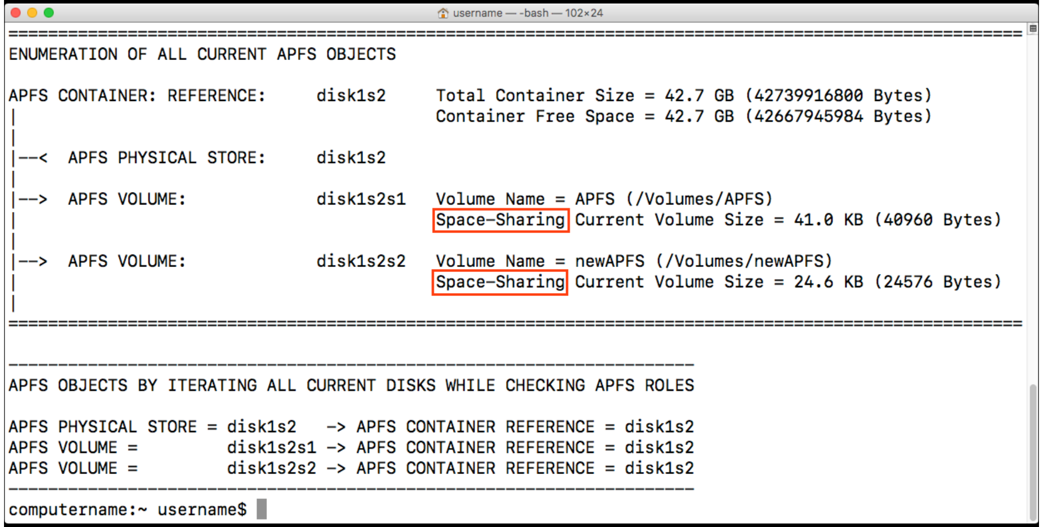
To create a new container on a physical device:
diskutil apfs create /dev/<devicename> new_name

The create verb is a convenient command that combines the functions of the createContainer and addVolume verbs. It converts an existing HFS Plus drive to an APFS drive with an empty container and then sets up one APFS volume in the container.



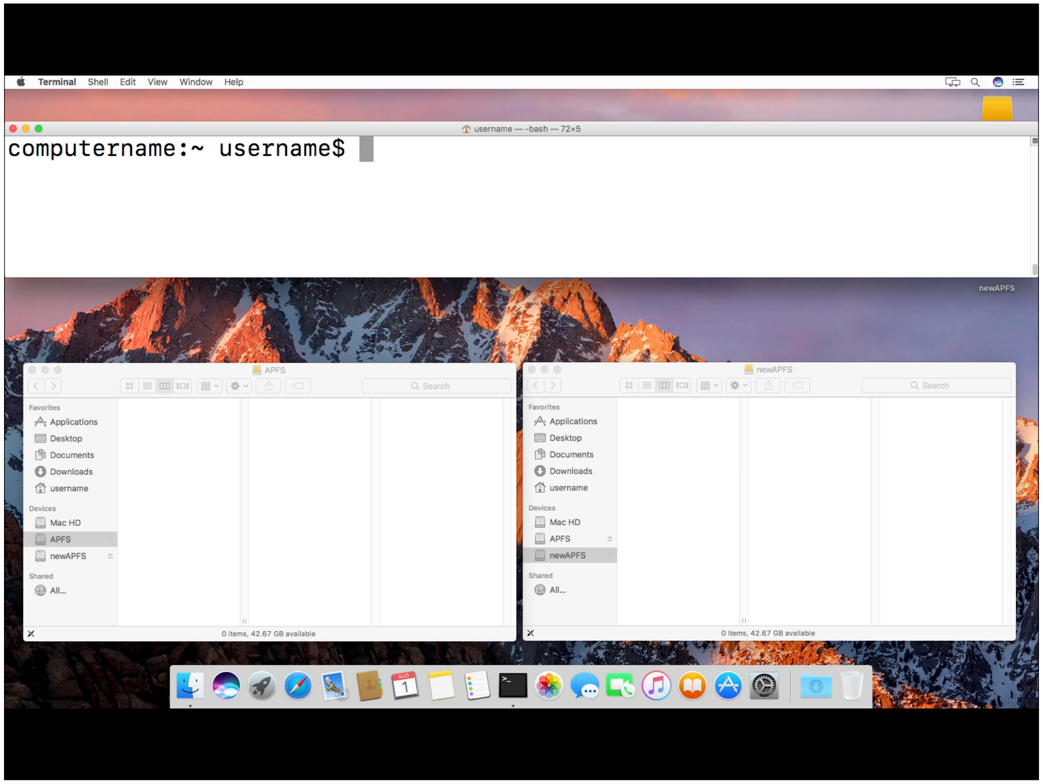
```
computername:~ username$ 
```

Apple File System



```
======================================================================================================
ENUMERATION OF ALL CURRENT APFS OBJECTS

APFS CONTAINER: REFERENCE:     disk1s2     Total Container Size = 42.7 GB (42739916800 Bytes)
|                                          Container Free Space = 42.7 GB (42667945984 Bytes)
|
|--<  APFS PHYSICAL STORE:     disk1s2
|
|-->  APFS VOLUME:             disk1s2s1   Volume Name = APFS (/Volumes/APFS)
|                                          Space-Sharing Current Volume Size = 41.0 KB (40960 Bytes)
|
|-->  APFS VOLUME:             disk1s2s2   Volume Name = newAPFS (/Volumes/newAPFS)
|                                          Space-Sharing Current Volume Size = 24.6 KB (24576 Bytes)
|
======================================================================================================


---------------------------------------------------------------------
APFS OBJECTS BY ITERATING ALL CURRENT DISKS WHILE CHECKING APFS ROLES

APFS PHYSICAL STORE = disk1s2   -> APFS CONTAINER REFERENCE = disk1s2
APFS VOLUME =         disk1s2s1 -> APFS CONTAINER REFERENCE = disk1s2
APFS VOLUME =         disk1s2s2 -> APFS CONTAINER REFERENCE = disk1s2
---------------------------------------------------------------------
computername:~ username$ 
```

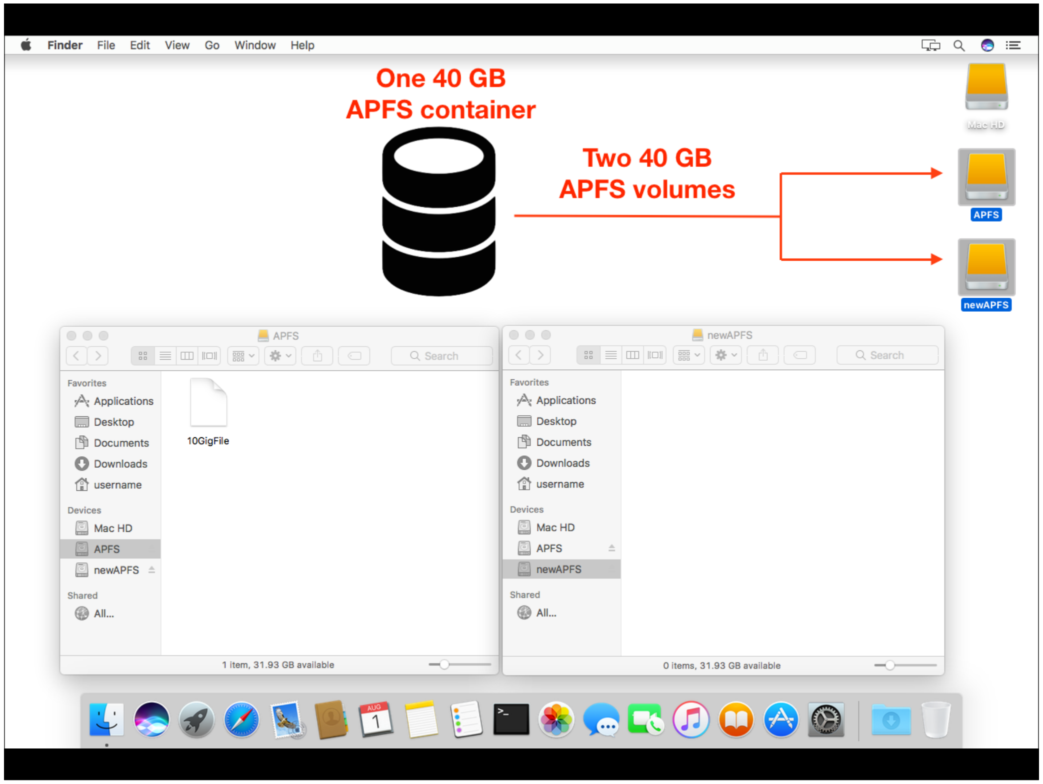
Something to be aware of is that volumes hosted on the same container will all display the same size and available free space of their parent APFS container. This behavior is referenced when you list the available APFS drives and look for the drives listed as Space-Sharing.



```
computername:~ username$
```

0 items, 42.67 GB available

0 items, 42.67 GB available



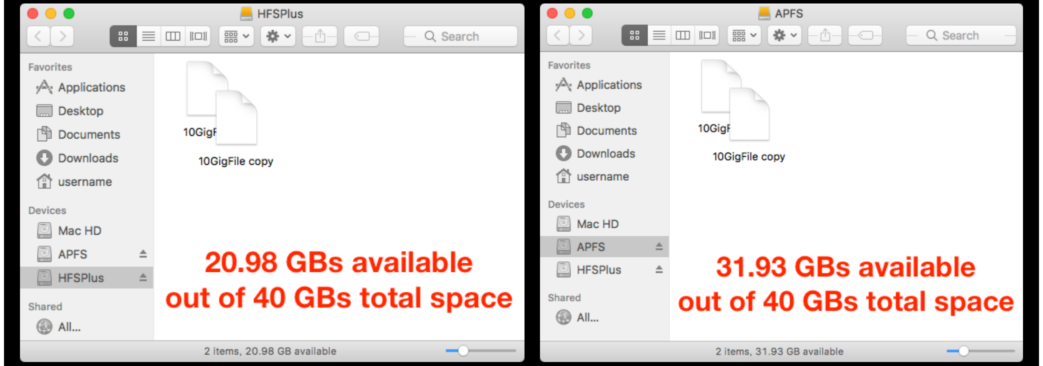
APFS: 1 item, 31.93 GB available

newAPFS: 0 items, 31.93 GB available

Because free space is being managed at the container level, you shouldn’t run into an issue where someone tries to save too much data to an APFS volume but this behavior is something to be aware of if you get asked why someone’s two drives apparently have X GBs of free space each, but you can only save a total of X GBs of data onto both drives. To help with this issue and to save on disk space overall, Apple is also introducing a new way to handle duplicate files as part of APFS.

Rather than save a complete duplicate of a file, APFS will instead make a reference to the original file when duplicating a file.

APFS Cloning



20.98 GBs available
out of 40 GBs total space

2 items, 20.98 GB available

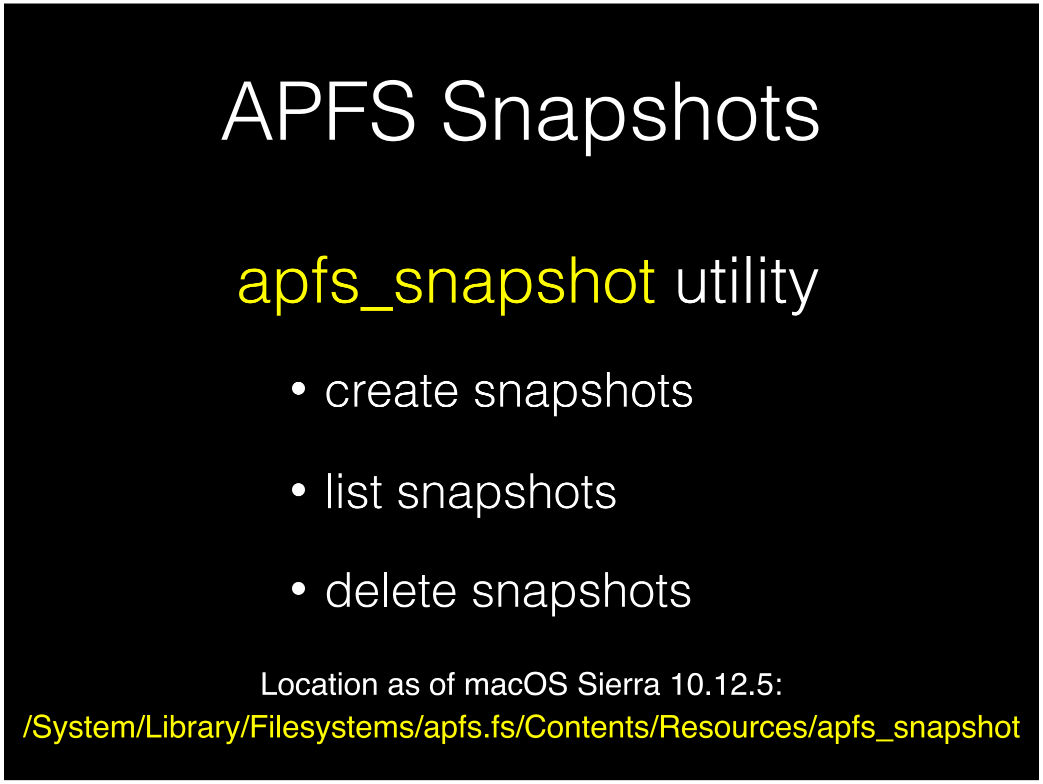
31.93 GBs available
out of 40 GBs total space

2 items, 31.93 GB available

Apple calls this new copy behavior cloning and is defining a clone in the APFS context as being a copy of a file or directory that occupies no additional space for file data and can be created nearly instantaneously.

When a cloned file is modified, only the modified blocks are written to disk. This allows the file system to do two things:

1. Store multiple revisions of the same document, by tracking which blocks were saved.
2. Use less file storage by only writing changes to disk and using the original file as a reference for the modifications.



APFS Snapshots

apfs_snapshot utility

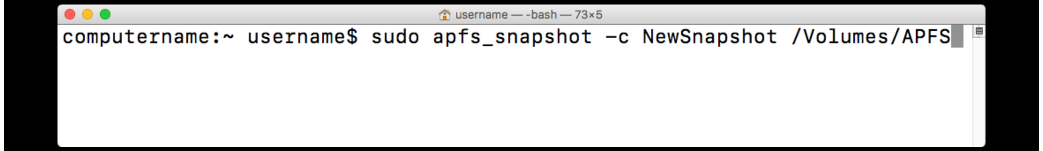
- create snapshots
- list snapshots
- delete snapshots

Location as of macOS Sierra 10.12.5:
/System/Library/Filesystems/apfs.fs/Contents/Resources/apfs_snapshot

As mentioned previously, APFS includes the ability to create snapshots of the APFS filesystem. For those not familiar with this, a snapshot is a read-only instance of a file system on a volume. Snapshots can be used to make backups work more efficiently and offer a way to revert changes to a given point in time.

In the APFS developer preview included with current releases of Sierra, the apfs_snapshot utility can be used to create, delete and list snapshots. Apple also notes in its APFS documentation that the apfs_snapshot utility will be disabled in a future release of macOS, so this tool will be superseded by another tool at some point.

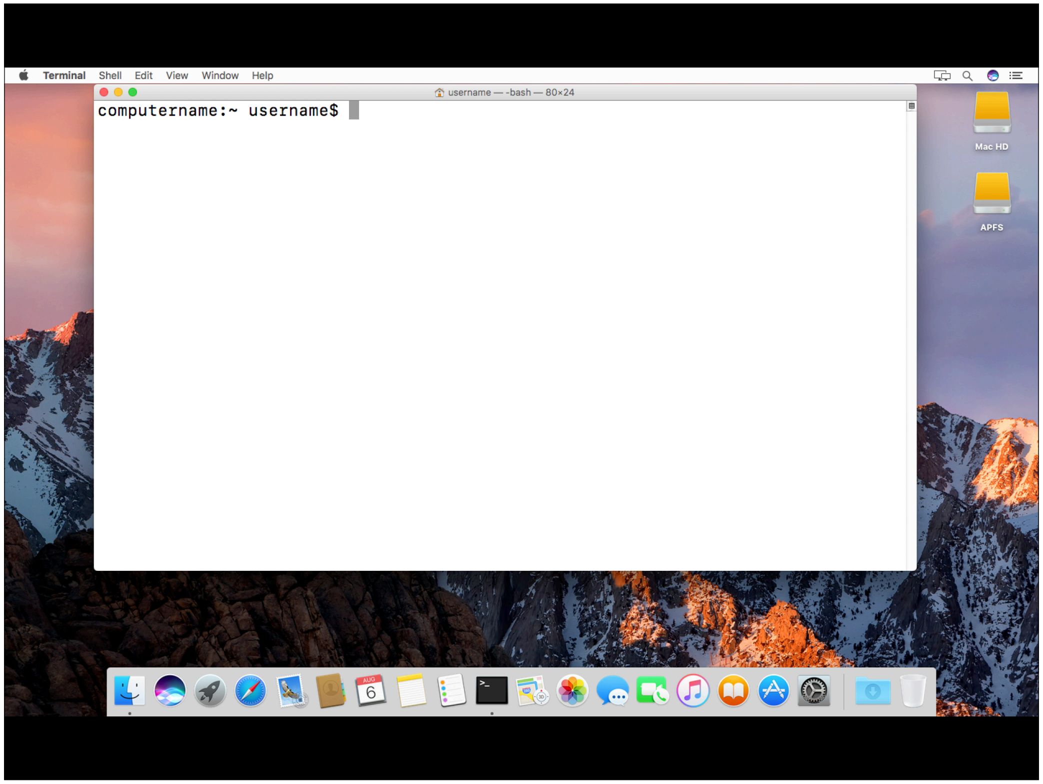
APFS Snapshots



```
computername:~ username$ sudo apfs_snapshot -c NewSnapshot /Volumes/APFS
```

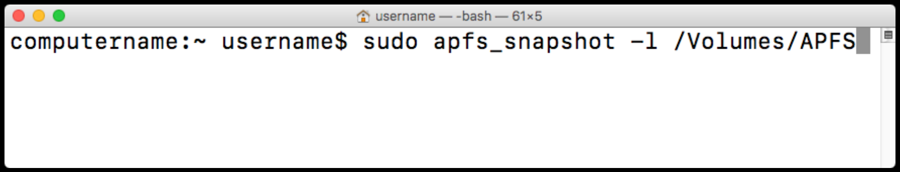
To create a new snapshot on an APFS volume:
apfs_snapshot -c SnapshotName /Volumes/<volumename>

To create a new snapshot on an APFS volume, run the command shown on the screen with root privileges.



```
computername:~ username$ 
```

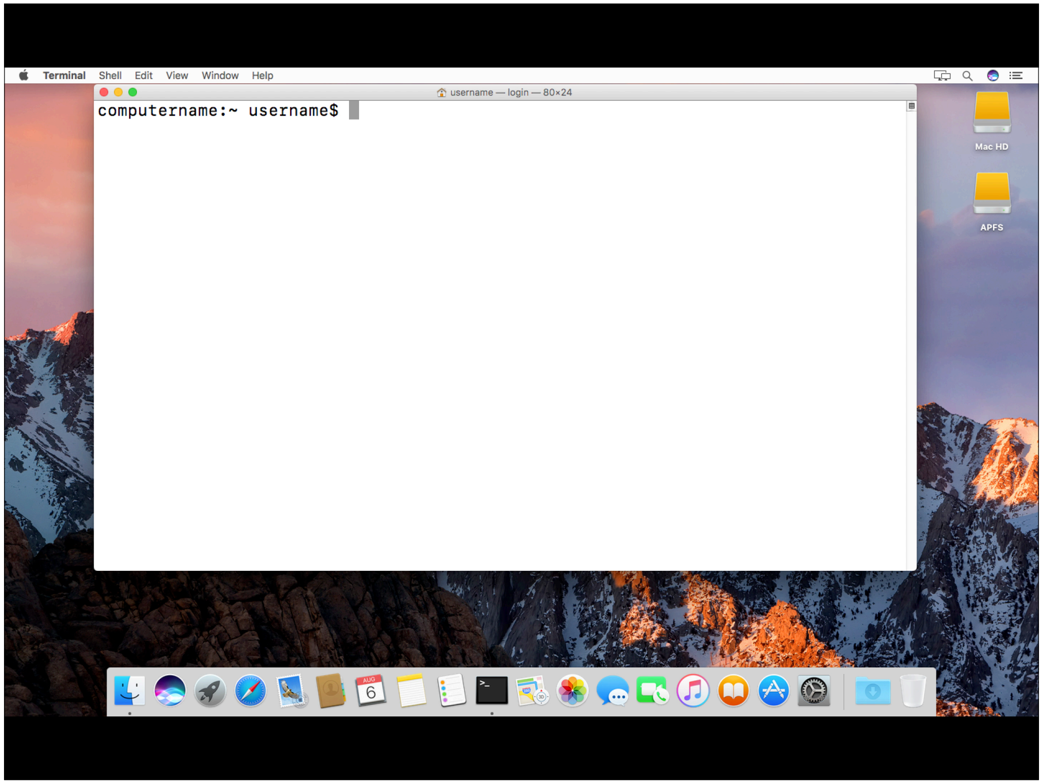
APFS Snapshots



```
computername:~ username$ sudo apfs_snapshot -l /Volumes/APFS
```

To list snapshots on an APFS volume:
apfs_snapshot -l /Volumes/<volumename>

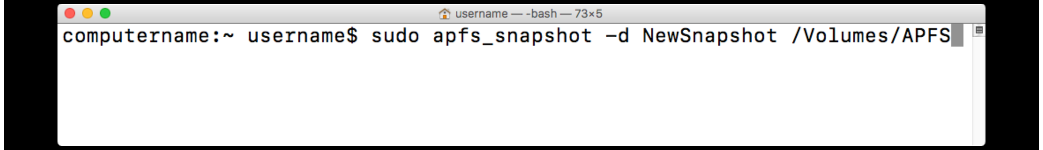
To list the available snapshots on an APFS volume, run the command shown on the screen with root privileges.



```
computername:~ username$
```

To list the available snapshots on an APFS volume, run the command shown on the screen with root privileges.

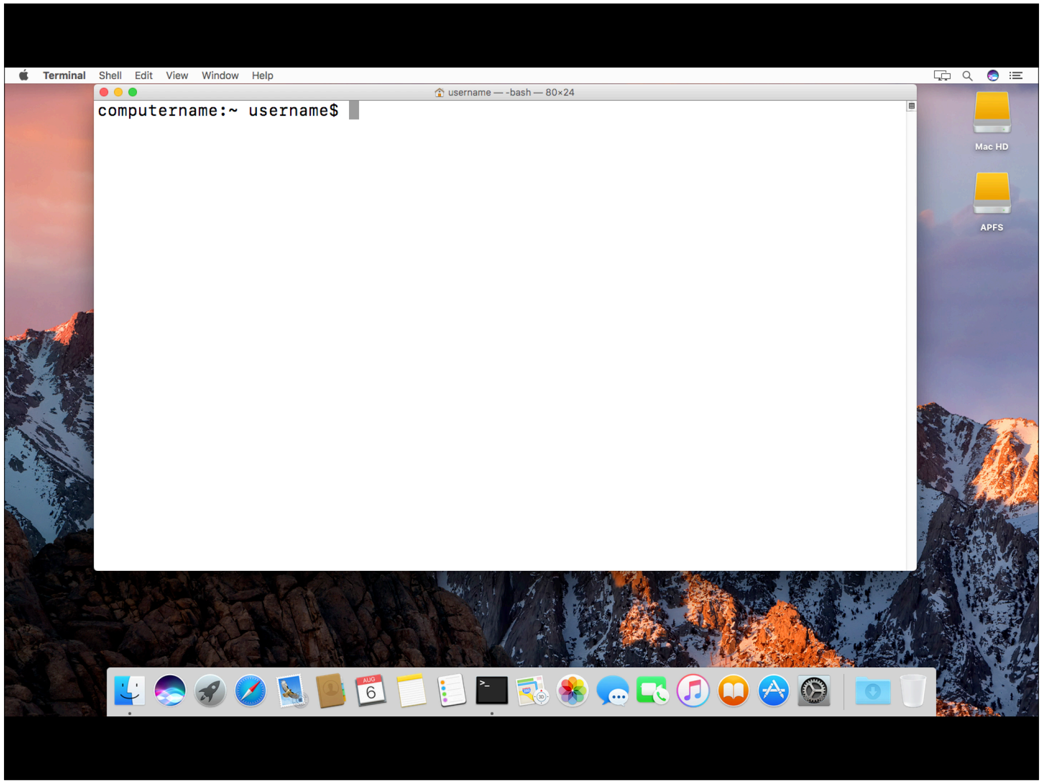
APFS Snapshots



```
computername:~ username$ sudo apfs_snapshot -d NewSnapshot /Volumes/APFS
```

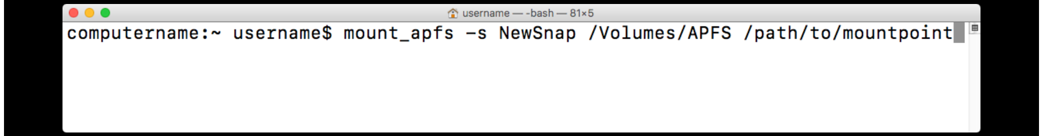
To delete a new snapshot on an APFS volume:
apfs_snapshot -d SnapshotName /Volumes/<volumename>

To delete a snapshot on an APFS volume, run the command shown on the screen with root privileges.



```
computername:~ username$
```

APFS Snapshots

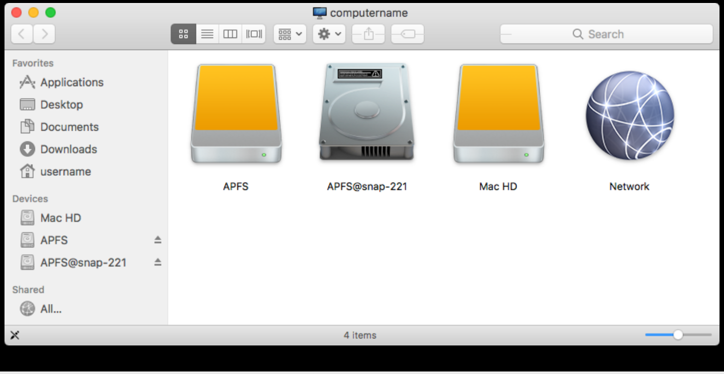


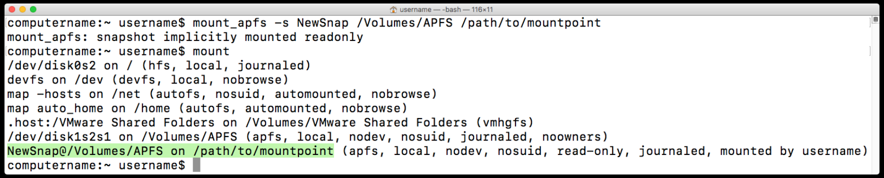
```
computername:~ username$ mount_apfs -s NewSnap /Volumes/APFS /path/to/mountpoint
```

To mount a snapshot as a read-only APFS volume:
mount_apfs -s SnapshotName <volumename> /path/to/mountpoint

To mount an existing snapshot as a read-only volume, you'll need to use the mount_apfs command's dash s option. As part of using the command, you'll also need to specify a directory for the snapshot to mount to.

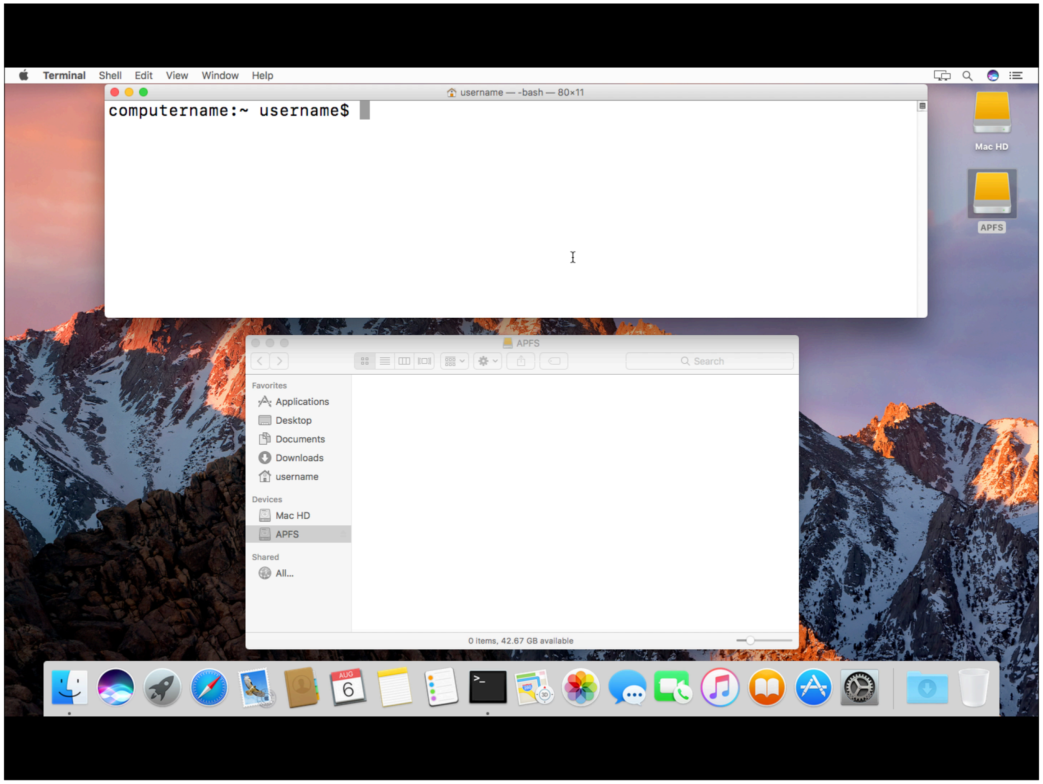
APFS Snapshots





```
computername:~ username$ mount_apfs -s NewSnap /Volumes/APFS /path/to/mountpoint
mount_apfs: snapshot implicitly mounted readonly
computername:~ username$ mount
/dev/disk0s2 on / (hfs, local, journaled)
devfs on /dev (devfs, local, nobrowse)
map -hosts on /net (autofs, nosuid, automounted, nobrowse)
map auto_home on /home (autofs, automounted, nobrowse)
.host:/VMware Shared Folders on /Volumes/VMware Shared Folders (vmhgfs)
/dev/disk1s2s1 on /Volumes/APFS (apfs, local, nodev, nosuid, journaled, noowners)
NewSnap@/Volumes/APFS on /path/to/mountpoint (apfs, local, nodev, nosuid, read-only, journaled, mounted by username)
computername:~ username$ 
```

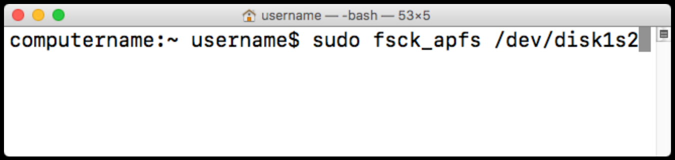
Once the drive is mounted, you should see it appear as a read-only drive in the Finder. You can also find information about how the snapshot is mounted using the mount command.



```
computername:~ username$
```

0 items, 42.67 GB available

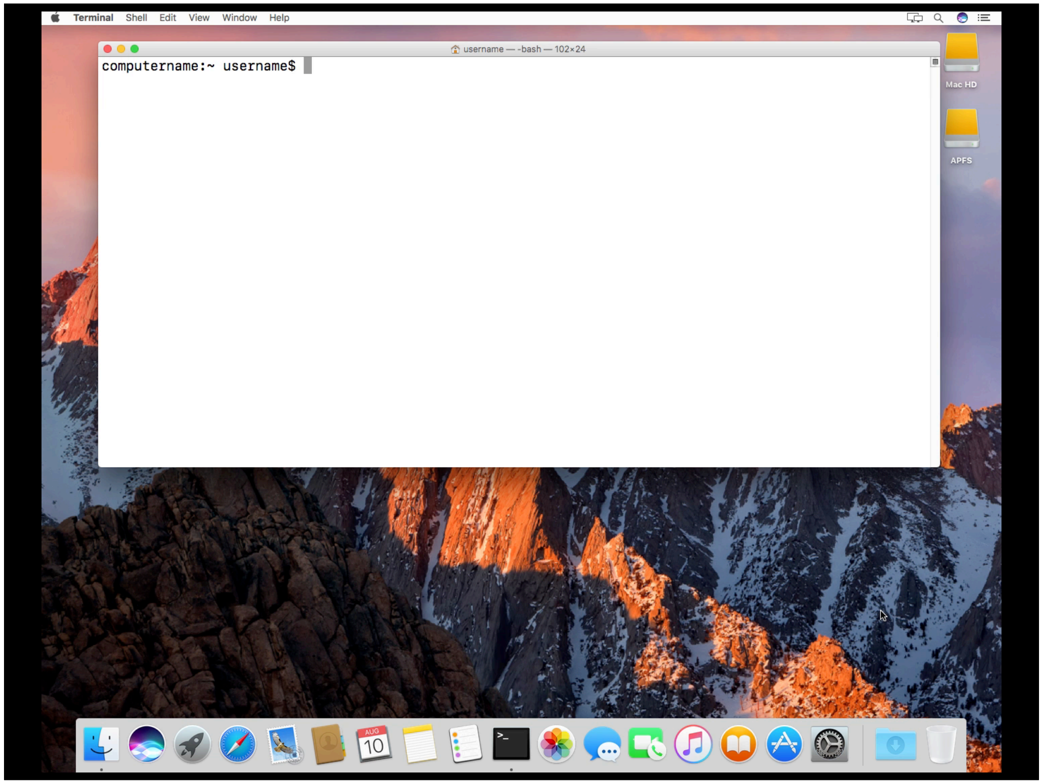
Apple File System



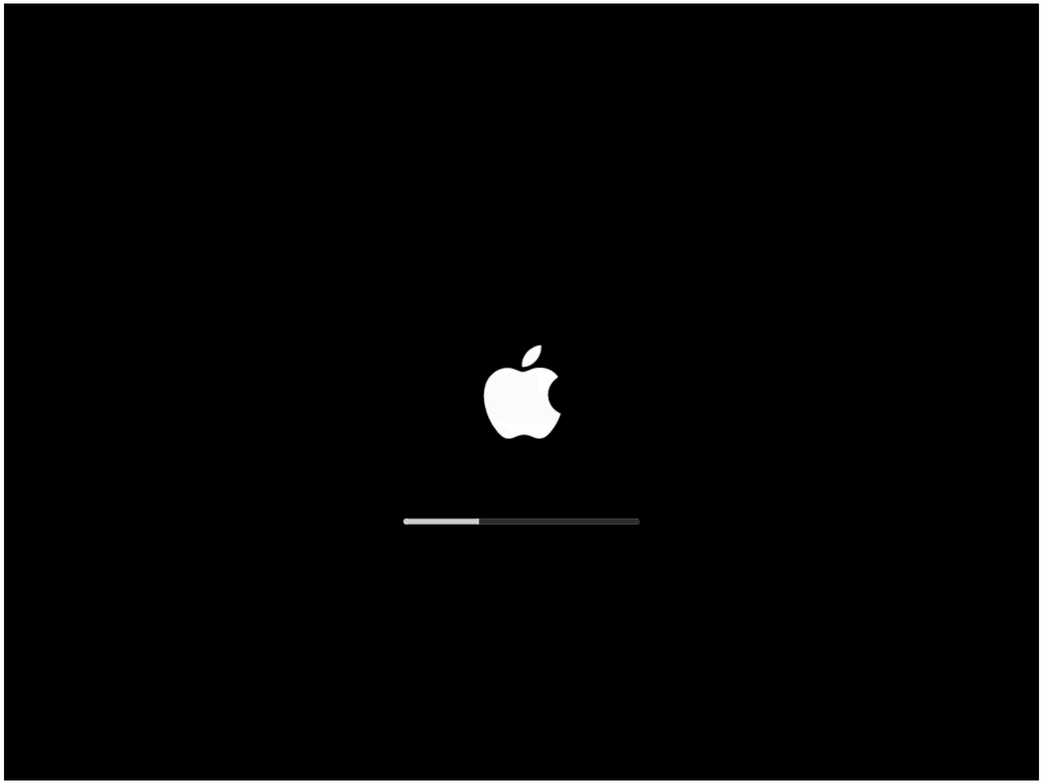
```
computername:~ username$ sudo fsck_apfs /dev/disk1s2
```

To create a new container on a physical device:
fsck_apfs /dev/<devicename>

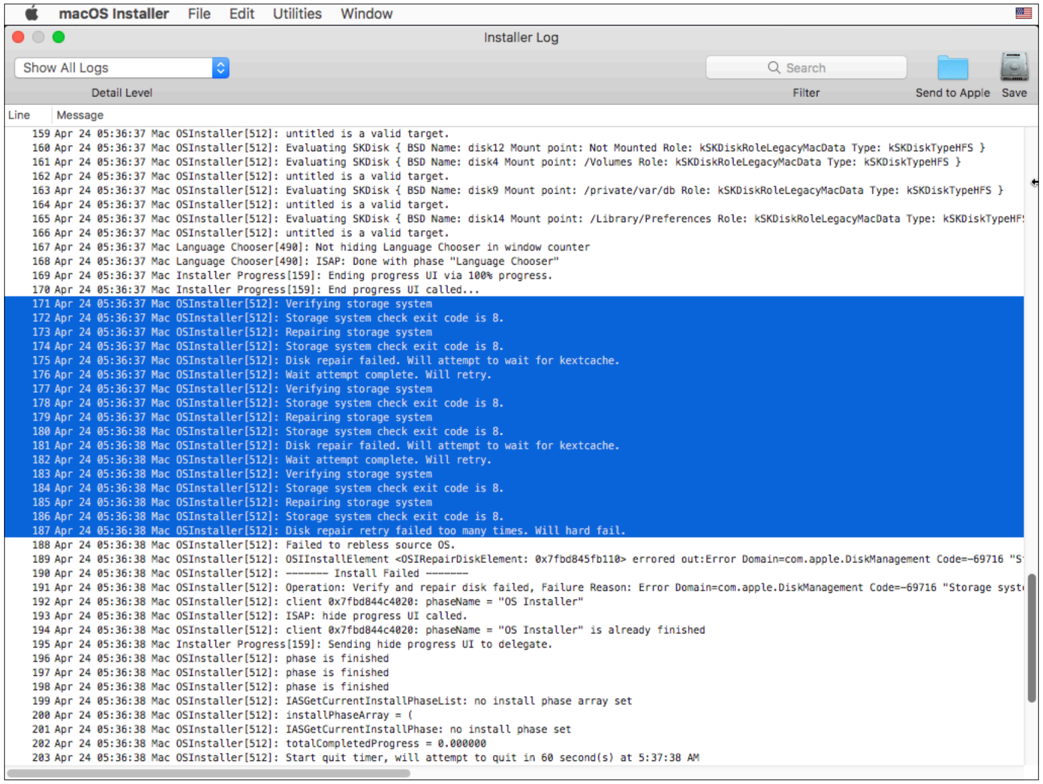
Also, because bad things can happen to good drives, if you want to fix a malfunctioning APFS drive, you can run the fsck apfs command with root privileges



```
computername:~ username$ 
```

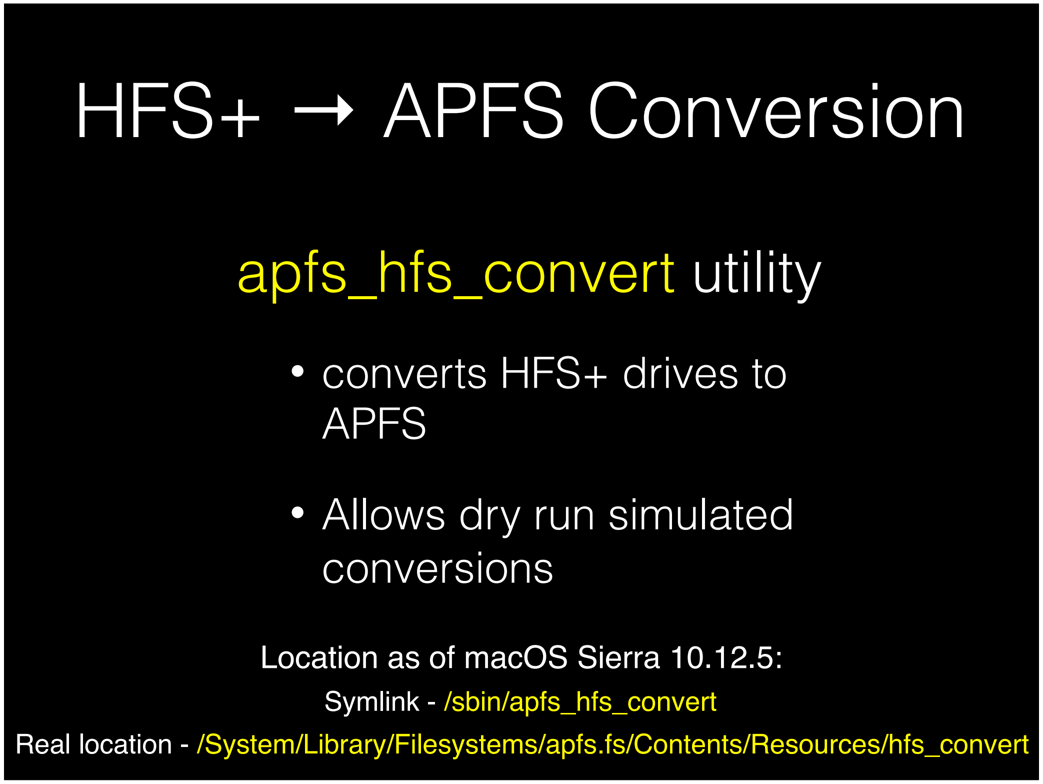


One area affected by this is that Sierra’s OS installer fails right away when you try to install onto an APFS-formatted drive.



```
Apr 24 05:36:37 Mac OSInstaller[512]: untitled is a valid target.
Apr 24 05:36:37 Mac OSInstaller[512]: Evaluating SKDisk { BSD Name: disk12 Mount point: Not Mounted Role: kSKDiskRoleLegacyMacData Type: kSKDiskTypeHFS }
Apr 24 05:36:37 Mac OSInstaller[512]: Evaluating SKDisk { BSD Name: disk4 Mount point: /Volumes Role: kSKDiskRoleLegacyMacData Type: kSKDiskTypeHFS }
Apr 24 05:36:37 Mac OSInstaller[512]: untitled is a valid target.
Apr 24 05:36:37 Mac OSInstaller[512]: Evaluating SKDisk { BSD Name: disk9 Mount point: /private/var/db Role: kSKDiskRoleLegacyMacData Type: kSKDiskTypeHFS }
Apr 24 05:36:37 Mac OSInstaller[512]: untitled is a valid target.
Apr 24 05:36:37 Mac OSInstaller[512]: Evaluating SKDisk { BSD Name: disk14 Mount point: /Library/Preferences Role: kSKDiskRoleLegacyMacData Type: kSKDiskTypeHF
Apr 24 05:36:37 Mac OSInstaller[512]: untitled is a valid target.
Apr 24 05:36:37 Mac Language Chooser[490]: Not hiding Language Chooser in window counter
Apr 24 05:36:37 Mac Language Chooser[490]: ISAP: Done with phase "Language Chooser"
Apr 24 05:36:37 Mac Installer Progress[159]: Ending progress UI via 100% progress.
Apr 24 05:36:37 Mac Installer Progress[159]: End progress UI called...
Apr 24 05:36:37 Mac OSInstaller[512]: Verifying storage system
Apr 24 05:36:37 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:37 Mac OSInstaller[512]: Repairing storage system
Apr 24 05:36:37 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:37 Mac OSInstaller[512]: Disk repair failed. Will attempt to wait for kextcache.
Apr 24 05:36:37 Mac OSInstaller[512]: Wait attempt complete. Will retry.
Apr 24 05:36:37 Mac OSInstaller[512]: Verifying storage system
Apr 24 05:36:37 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:37 Mac OSInstaller[512]: Repairing storage system
Apr 24 05:36:38 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:38 Mac OSInstaller[512]: Disk repair failed. Will attempt to wait for kextcache.
Apr 24 05:36:38 Mac OSInstaller[512]: Wait attempt complete. Will retry.
Apr 24 05:36:38 Mac OSInstaller[512]: Verifying storage system
Apr 24 05:36:38 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:38 Mac OSInstaller[512]: Repairing storage system
Apr 24 05:36:38 Mac OSInstaller[512]: Storage system check exit code is 8.
Apr 24 05:36:38 Mac OSInstaller[512]: Disk repair retry failed too many times. Will hard fail.
Apr 24 05:36:38 Mac OSInstaller[512]: Failed to rebless source OS.
Apr 24 05:36:38 Mac OSInstaller[512]: OSIInstallElement <OSIRepairDiskElement: 0x7fbd845fb110> errored out:Error Domain=com.apple.DiskManagement Code=-69716 "S
Apr 24 05:36:38 Mac OSInstaller[512]: -------- Install Failed --------
Apr 24 05:36:38 Mac OSInstaller[512]: Operation: Verify and repair disk failed, Failure Reason: Error Domain=com.apple.DiskManagement Code=-69716 "Storage syst
Apr 24 05:36:38 Mac OSInstaller[512]: client 0x7fbd844c4020: phaseName = "OS Installer"
Apr 24 05:36:38 Mac OSInstaller[512]: ISAP: hide progress UI called.
Apr 24 05:36:38 Mac OSInstaller[512]: client 0x7fbd844c4020: phaseName = "OS Installer" is already finished
Apr 24 05:36:38 Mac Installer Progress[159]: Sending hide progress UI to delegate.
Apr 24 05:36:38 Mac OSInstaller[512]: phase is finished
Apr 24 05:36:38 Mac OSInstaller[512]: phase is finished
Apr 24 05:36:38 Mac OSInstaller[512]: phase is finished
Apr 24 05:36:38 Mac OSInstaller[512]: IASGetCurrentInstallPhaseList: no install phase array set
Apr 24 05:36:38 Mac OSInstaller[512]: installPhaseArray = (
Apr 24 05:36:38 Mac OSInstaller[512]: IASGetCurrentInstallPhase: no install phase set
Apr 24 05:36:38 Mac OSInstaller[512]: totalCompletedProgress = 0.000000
Apr 24 05:36:38 Mac OSInstaller[512]: Start quit timer, will attempt to quit in 60 second(s) at 5:37:38 AM
```

The reason for the failure is that the disk check in the OS installer is not expecting an APFS formatted drive and causes the install process to hard fail when the disk check fails.



HFS+ ➞ APFS Conversion

apfs_hfs_convert utility

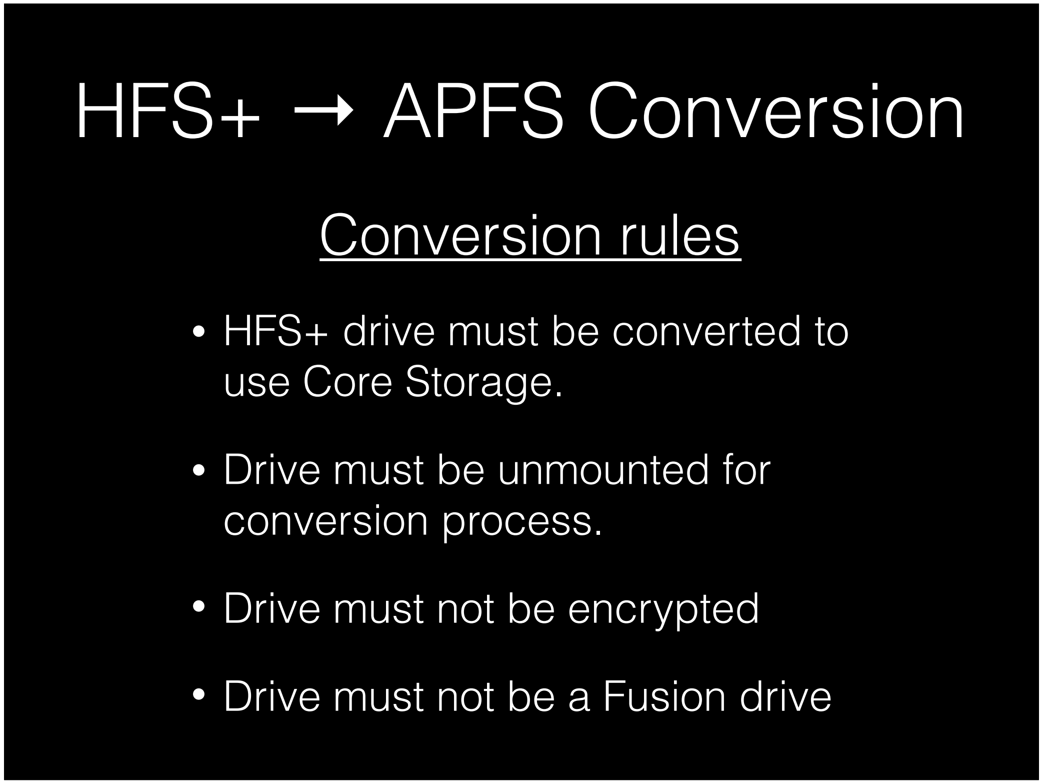
- converts HFS+ drives to APFS
- Allows dry run simulated conversions

Location as of macOS Sierra 10.12.5:

Symlink - /sbin/apfs_hfs_convert

Real location - /System/Library/Filesystems/apfs.fs/Contents/Resources/hfs_convert

One of the items Apple has been promising as part of the move from HFS Plus to APFS is a smooth conversion process. The tool Apple will be using for the conversion is called apfs hfs convert and it is available in Sierra.



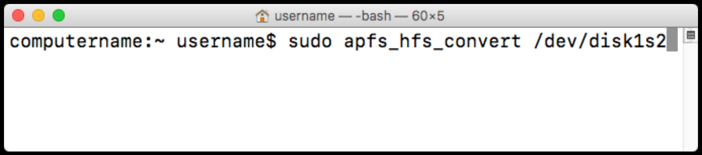
HFS+ → APFS Conversion

Conversion rules

- HFS+ drive must be converted to use Core Storage.
- Drive must be unmounted for conversion process.
- Drive must not be encrypted
- Drive must not be a Fusion drive

I’ve tested the conversion process in Sierra and it works, as long as the conditions listed on the screen apply. These rules may not apply to the conversion process in future OS versions, this is just for Sierra.

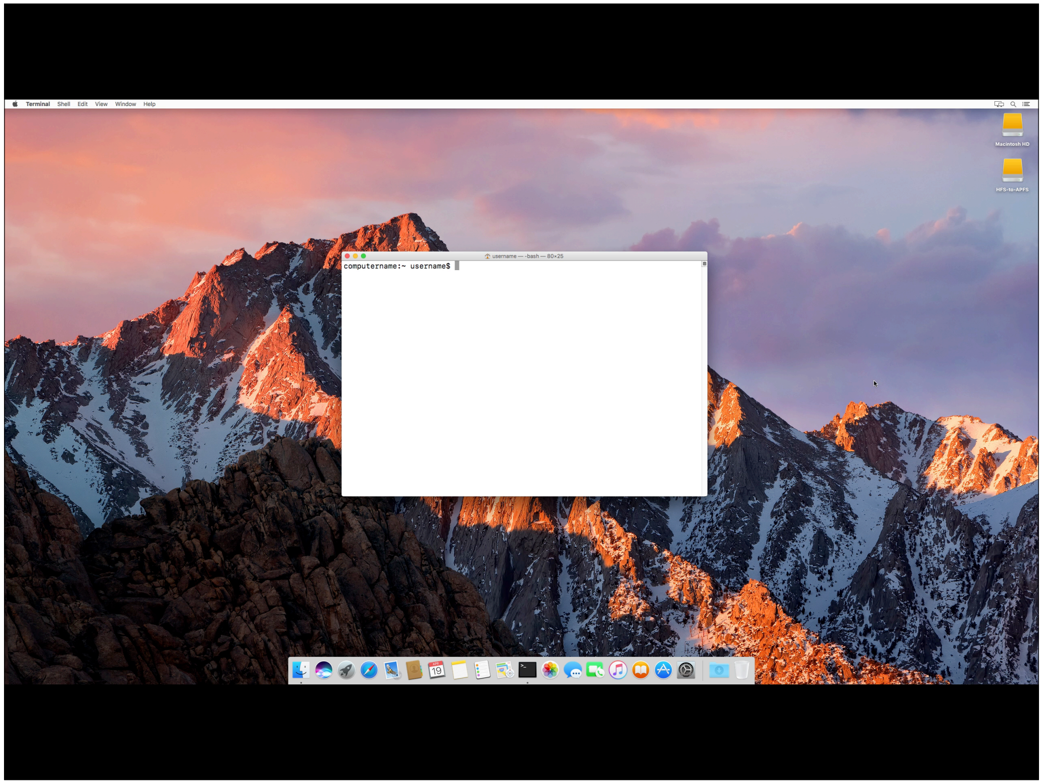
HFS+ → APFS Conversion



```
computername:~ username$ sudo apfs_hfs_convert /dev/disk1s2
```

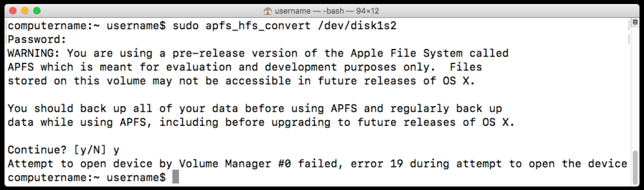
To convert a drive from HFS+ to APFS:
apfs_hfs_convert /dev/disk_id_goes_here

To run the conversion process, first make sure all conversion conditions have been met. Once they have been, run the command shown on the screen with root privileges.

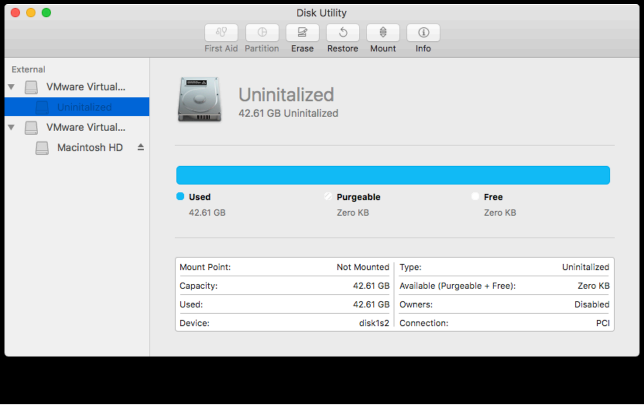


```
computername:~ username$ 
```

HFS+ → APFS Conversion



```
computername:~ username$ sudo apfs_hfs_convert /dev/disk1s2
Password:
WARNING: You are using a pre-release version of the Apple File System called
APFS which is meant for evaluation and development purposes only.  Files
stored on this volume may not be accessible in future releases of OS X.

You should back up all of your data before using APFS and regularly back up
data while using APFS, including before upgrading to future releases of OS X.

Continue? [y/N] y
Attempt to open device by Volume Manager #0 failed, error 19 during attempt to open the device
computername:~ username$ 
```



Uninitalized

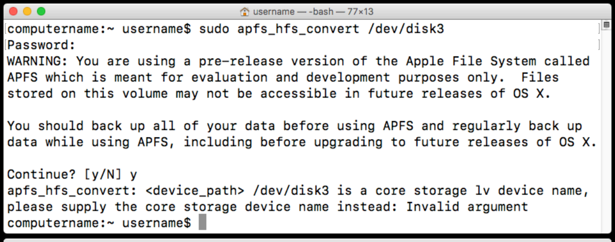
42.61 GB Uninitalized

Used	Purgeable	Free
42.61 GB	Zero KB	Zero KB

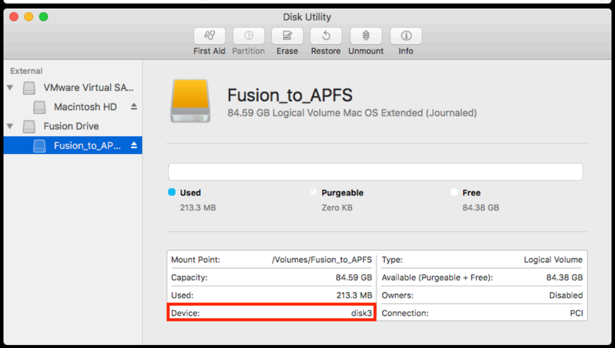
Mount Point:	Not Mounted	Type:	Uninitalized
Capacity:	42.61 GB	Available (Purgeable + Free):	Zero KB
Used:	42.61 GB	Owners:	Disabled
Device:	disk1s2	Connection:	PCI

When I’ve tested converting encrypted drives in Sierra, I get results that look like this. The conversion process fails with an error, the drive winds up in a state which Disk Utility labels as uninitialized and it’s not possible to mount the drive.

HFS+ → APFS Conversion



```
computername:~ username$ sudo apfs_hfs_convert /dev/disk3
Password:
WARNING: You are using a pre-release version of the Apple File System called
APFS which is meant for evaluation and development purposes only.  Files
stored on this volume may not be accessible in future releases of OS X.

You should back up all of your data before using APFS and regularly back up
data while using APFS, including before upgrading to future releases of OS X.

Continue? [y/N] y
apfs_hfs_convert: <device_path> /dev/disk3 is a core storage lv device name,
please supply the core storage device name instead: Invalid argument
computername:~ username$ 
```



Fusion_to_APFS

84.59 GB Logical Volume Mac OS Extended (Journaled)

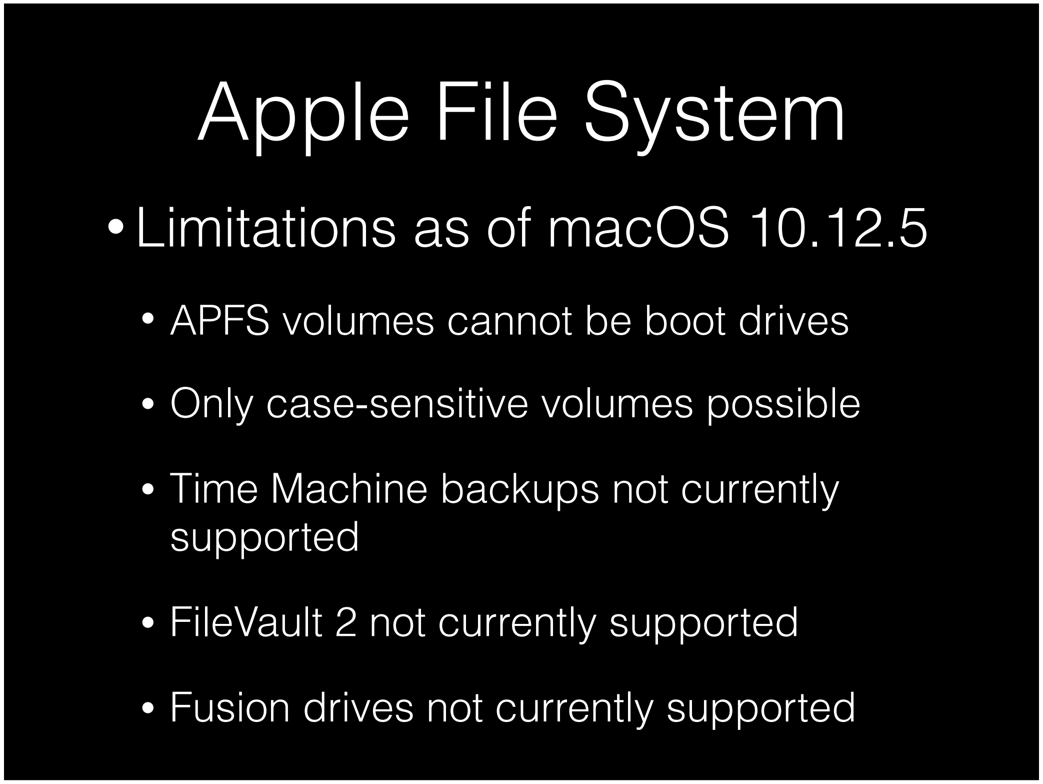
● Used 213.3 MB

Purgeable Zero KB

Free 84.38 GB

Mount Point:	/Volumes/Fusion_to_APFS	Type:	Logical Volume
Capacity:	84.59 GB	Available (Purgeable + Free):	84.38 GB
Used:	213.3 MB	Owners:	Disabled
Device:	disk3	Connection:	PCI

When I’ve tested converting Fusion drives in Sierra, I get results that look like this. apfs hfs convert gives me an message that I’ve targeted a Core Storage logical volume and the process halts at that point. No conversion takes place.



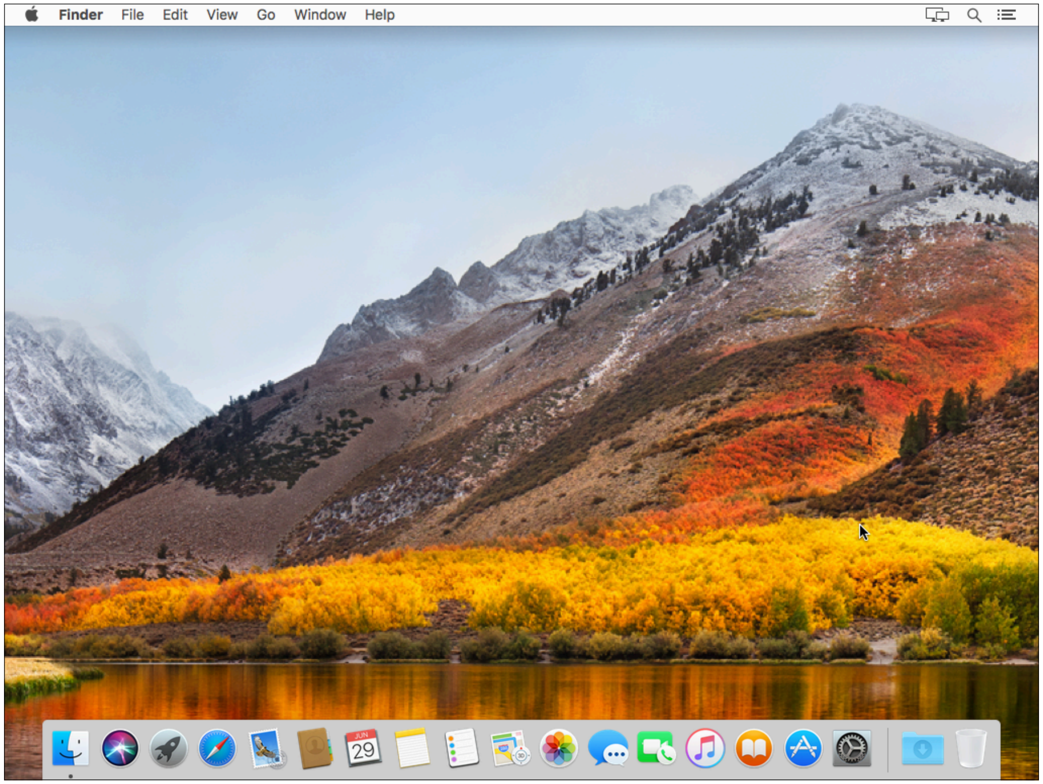
Apple File System

- Limitations as of macOS 10.12.5
  - APFS volumes cannot be boot drives
  - Only case-sensitive volumes possible
  - Time Machine backups not currently supported
  - FileVault 2 not currently supported
  - Fusion drives not currently supported

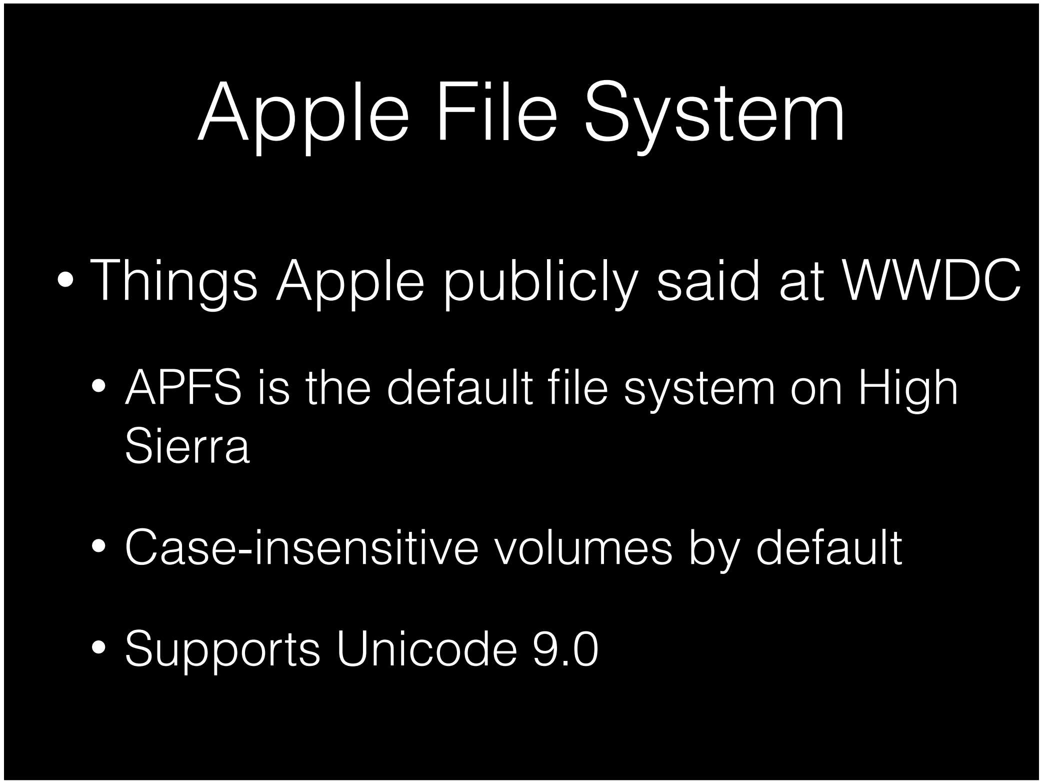
There are currently several limitations of APFS to be aware of:

Startup Disk: An APFS volume cannot currently be used as a startup disk.
Case Sensitivity: Volumes can currently only be case-sensitive.
Time Machine: Time Machine backups are not currently supported.
FileVault: APFS volumes cannot currently be encrypted using FileVault.
Fusion Drive: Fusion Drives cannot currently use APFS.

So that’s the state of APFS on Sierra.



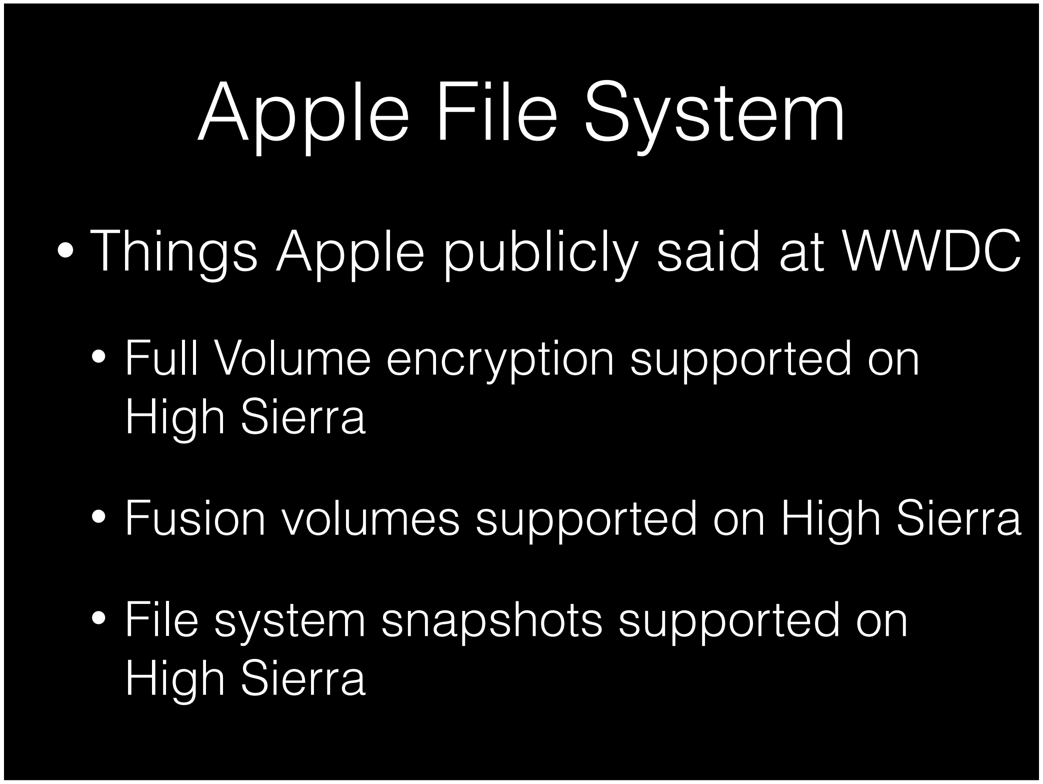
But what about High Sierra?



Apple File System

- Things Apple publicly said at WWDC
  - APFS is the default file system on High Sierra
  - Case-insensitive volumes by default
  - Supports Unicode 9.0

Apple made some public statements about APFS at WWDC, which means that I can also talk about them. Among those statements was that APFS is the new default filesystem.

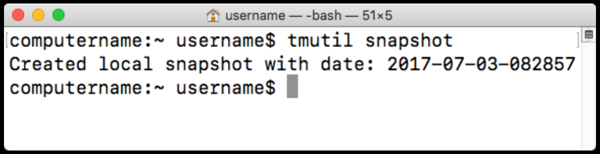


Apple File System

- Things Apple publicly said at WWDC
  - Full Volume encryption supported on High Sierra
  - Fusion volumes supported on High Sierra
  - File system snapshots supported on High Sierra

They also said that snapshots are supported on High Sierra.

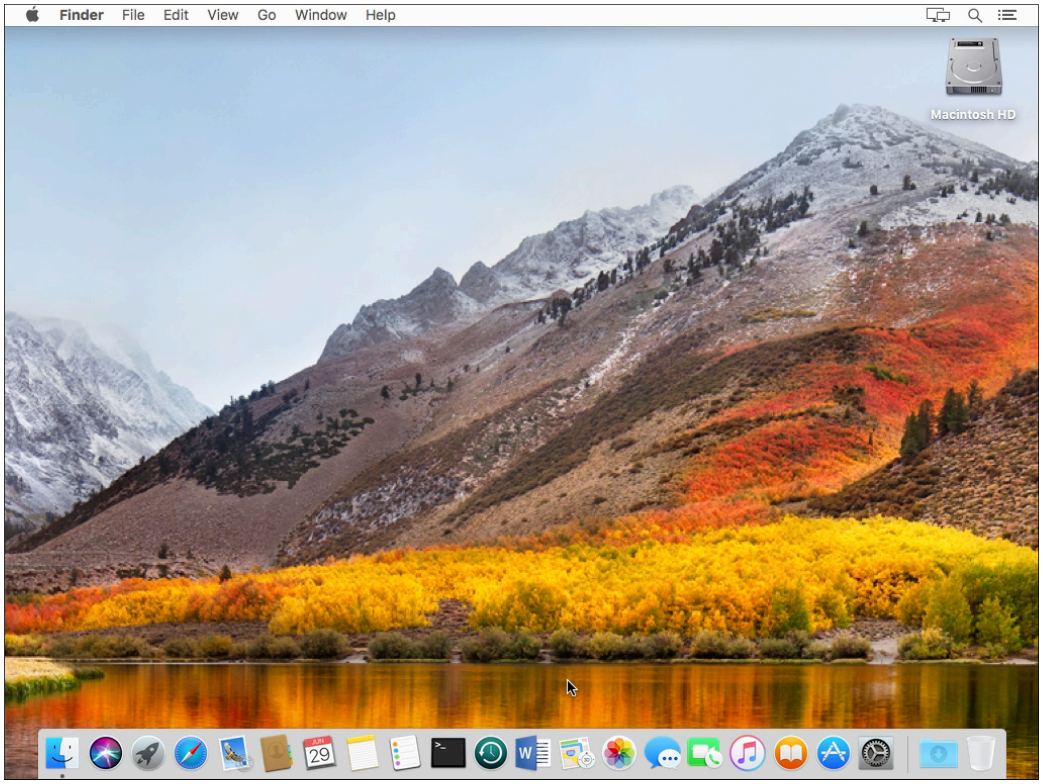
APFS Snapshots

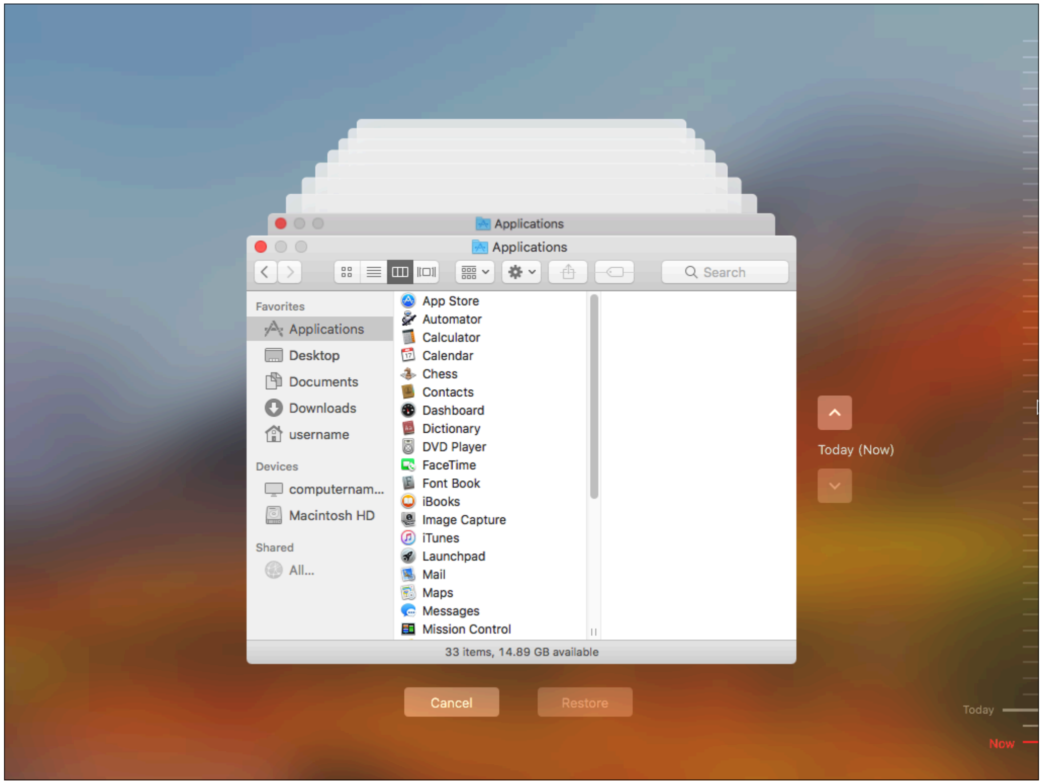


```
computername:~ username$ tmutil snapshot
Created local snapshot with date: 2017-07-03-082857
computername:~ username$
```

To create a new snapshot on an APFS boot volume:
tmutil snapshot

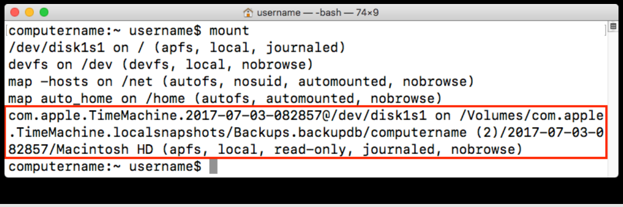
To create a snapshot on an APFS boot volume, run the tmutil command shown on the screen.


Finder
File
Edit
View
Go
Window
Help
Macintosh HD

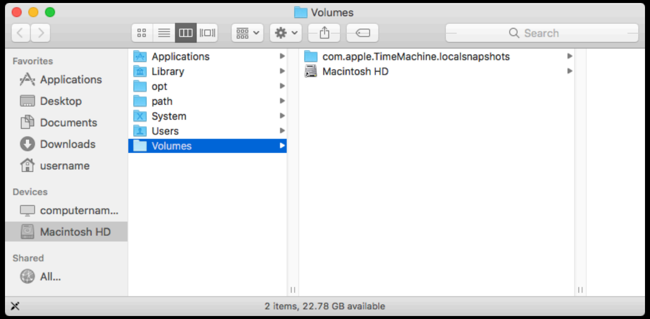


The interface for accessing the snapshot’s contents is the Time Machine interface and associated command line tools like tmutil.

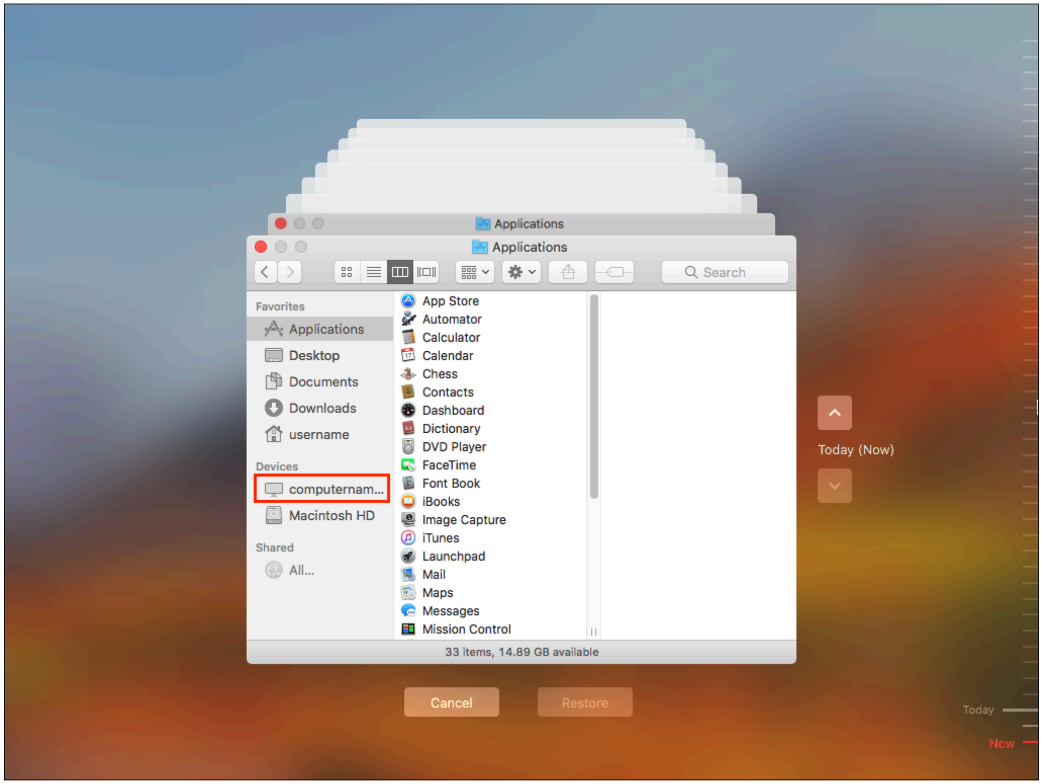
APFS Snapshots



```
computername:~ username$ mount
/dev/disk1s1 on / (apfs, local, journaled)
devfs on /dev (devfs, local, nobrowse)
map -hosts on /net (autofs, nosuid, automounted, nobrowse)
map auto_home on /home (autofs, automounted, nobrowse)
com.apple.TimeMachine.2017-07-03-082857@/dev/disk1s1 on /Volumes/com.apple
.TimeMachine.localsnapshots/Backups.backupdb/computername (2)/2017-07-03-0
82857/Macintosh HD (apfs, local, read-only, journaled, nobrowse)
computername:~ username$
```



When Time Machine is using a snapshot, you can find information about how the snapshot is mounted using the mount command. The snapshot should be mounting by default into slash Volumes.



If your Finder sidebar settings allow the display of external mounted drives, you should also see the mounted snapshot appear when you enter Time Machine.

Finder
File
Edit
View
Go
Window
Help
Macintosh HD
29

Useful Links

- Apple File System Guide: https://developer.apple.com/library/prerelease/content/documentation/FileManagement/Conceptual/APFS_Guide
- APFS: https://en.wikipedia.org/wiki/Apple_File_System
- Apple File System (Russ Bishop) - http://www.russbishop.net/apple-file-system
- Digging into the dev documentation for APFS, Apple's new file system: http://arstechnica.com/apple/2016/06/digging-into-the-dev-documentation-for-apfs-apples-new-file-system/

Useful Links

- FileVault 2 and the rise of Apple File System: https://derflounder.wordpress.com/2016/10/07/filevault-2-and-the-rise-of-apple-file-system/
- macOS High Sierra tech preview - A quick look at the stuff you can’t see: https://arstechnica.com/apple/2017/06/macos-high-sierra-tech-preview-a-quick-look-at-the-stuff-you-cant-see/
- macOS Sierra Includes a New Apple File System, APFS: http://blog.fosketts.net/2016/06/13/macos-sierra-includes-new-apple-file-system-apfs/

Downloads

PDF available from the following link:

http://tinyurl.com/psumac2017pdf

Keynote slides available from the following link:

http://tinyurl.com/psumac2017key