## NAME

**sync** — flush kernel caches

## SYNOPSIS

**sync** [**-d**|**-f**] [*file*]...

## DESCRIPTION

Instruct the kernel to commit cached writes to *file*s to their backing storage. If none specified, flush all caches.

## OPTIONS

**-d**, **--data** Don't flush metadata.
**-f**, **--file-system** Flush entire filesystem containing each *file*. Only availabe under Linux.

## SEE ALSO

fdatasync(2), fsync(2), sync(2), syncfs(2)

## STANDARDS

Compatible with Version 4 AT&T UNIX; arguments are a GNU extension.

## HISTORY

sync(II) appears in the second edition of the UNIX Programmer's Manual as

```
sync ....................... assure synchronisation
```

in the index and

```
NAME        sync -- update super-block
```

on its page, described as causing "the super block for all file systems to be written out", and to only be necessary on systems incorporating "hardware protection facilities", as those may "delay this writing for a long time". It also notes that it "should be used by programs which examine a file system", as those read the underlying block devices directly.

Version 4 AT&T UNIX broadens the description to causing "all information in core memory that should be on disk to be written out", listing "modified super blocks" and i-nodes, and "delayed block I/O". as well as noting it to "be mandatory before a boot", with the new sync(VIII) and update(VIII) in **SEE ALSO**.

sync(VIII) ("update the super block") is described simply as

> executes the *sync* system primitive.

whereas update(VIII) ("periodically update the super block")

> executes the *sync* primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash.

nevertheless, its **BUGS** section states, that

> There is a system bug which, it is suspected, may be aggravated by this program. Until further notice, *update* should not be run.

Version 5 AT&T UNIX elucidates the synchronisation/insurance relationship in sync(VIII) and expands update(VIII) **BUGS**:

> With *update* running, if the CPU is halted just as the *sync* is executed, a file system can be damaged. This is partially due to DEC hardware that writes zeros when NPR requests fail. A fix would be to have *sync* temporarily increment the system time by at least 30 seconds to trigger the execution of *update*. This would give 30 seconds grace to halt the CPU.

Version 7 AT&T UNIX sync(2) notes in its **BUGS** section that

> The writing, although scheduled, is not necessarily complete upon return from *sync*.

sync(VIII) moves to sync(1M) (though sync(2) refers to sync(1)) with no further changes.

update(8) disappears from Programmer's Workbench (PWB/UNIX). Further, AT&T System III UNIX moves the Version 7 AT&T UNIX sync(2) **BUGS** line into the description.

4.1BSD moves `sync`(1M) back to `sync`(8), despite the executable remaining squarely in the global `/bin`.

The BSD sees a steady evolution of `update`(8), and, with 4.4BSD, shortening the **BUGS** to

> It is possible on some systems that a `sync` occuring simultaneously with a crash may cause file system damage. See `fsck`(8).

(with no useful notes in `fsck`(8)) and being mentioned in `sync`(2) as a requirement for a healthy system.

This is not necessarily entirely unlike the user-space flushing daemons (`bdflush`(8)**/bdflushd/updated/**`bdflush`(2)), as found in Linux 1.1.3 through 2.5.12, though with a different scope (`bdflush`(2) is in some configurations an entire kernel-side flushing daemon) and for a different reason (Linux being single-threaded).