How do I debug this FS error on a flash device?

source : http://serverfault.com/questions/331779/how-do-i-debug-this-fs-error-on-a-flash-device

I have console access to an embedded linux device. This device has flash memory part of which is partitioned as a FAT filesystem.

Its running linux-2.6.31.

However I am seeing these errors on the console these days and the FAT file system becomes read only.

```
111109:154925 FAT: Filesystem error (dev loop0)
111109:154925 fat_get_cluster: invalid cluster chain (i_pos 0)
111109:154925 FAT: Filesystem error (dev loop0)
111109:154925 fat_get_cluster: invalid cluster chain (i_pos 0)
```

I cannot understand why this happened? What is the root cause? And what is the fix? I would appreciate answers that can point me how to investigate the possible root cause of this issue on the device.

---

is the flash memory corrupted? – The Unix Janitor Nov 16 '11 at 22:44

Have you performed a disk check? chkdsk in a windows for example. – The_aLiEn Nov 16 '11 at 22:52

Flash memory is not corrupted, however I see this error every now and then. I want to figure out why is it happening ? Because of the bug in the software or is it the flash memory part that has a problem. – abc Nov 16 '11 at 23:00

Probably not a software bug, FAT is very well tested. Maybe the flash is corrupted to to sudden powerloss. Make sure you have backup / dd image of that disk. – AndreasM Nov 17 '11 at 8:03

What has actually happened on a bits-and-bytes level is that 4 bytes (or more) of the file allocation table has been overwritten with `0x00` bytes.

I'll explain in brief how the file allocation table works. It may be looked at as an array whose values are indices of that same array. So if we know that the first cluster number of a file is `i`, then the next cluster number is `fat[i]`, and the next after that is `fat[fat[i]]`, and so on. (This is slightly simplified). To signal that the end of the chain is reached, a special EOC value is used instead of a valid cluster number.

To read a FAT file from disk, you need the cluster numbers where the file is stored, in order. The directory entry gives the first cluster number (`i`). The rest can be found following the chain fat[i], fat[fat[i]] etc. until the EOC value is encountered. Then it's a simple calculation to get the on-disk position of each cluster from the cluster numbers, read each cluster into memory and concatenate them.

The `fat_get_cluster: invalid cluster chain error` occurs when the value `0x00000000` is found following such a chain. This should not happen. It should either be a new valid cluster number or the EOC value. No more of the file can be read when this happens, since there's no way of following the chain further. (The `0x00000000` value is used to mark a cluster as free. Cluster 0 is never used to store data, so there's no ambiguity)

Your case may be a special case, since `i_pos` is given as `0`. When I got this message, it was a big number. Kernel source says:

```
    loff_t i_pos;           /* on-disk position of directory entry or 0 */
```

So `i_pos` is not a cluster number, but a position on the disk. What it means when it's zero, I don't know.

EDIT: As to what might have caused it, I can only speculate, but here are some possibilities:

1. A FAT driver bug.
2. Cosmic rays.
3. A virus or other malicious software.
4. Maybe if two programs/drivers were writing and reading to the same FAT simultaneously for some reason, they might trip over each other. Don't know if it's possible.
5. Hard power off at the wrong moment. Flash drives must zero out a block before writing changes, so theoretically a power off right after the erase would cause this result. There are failsafes to prevent this on most drives though.

6. User error or sabotage (e.g. `dd if=/dev/zero of=/dev/sda1 bs=512 count=1 seek=32` -- don't try this at home!)

FAT filesystem drivers actually keeps two FAT tables up-to-date for redundancy, the second lying just after the first. Checking to see if they're identical might give clues to what might have happened. If they differed in only the value that breaks the cluster chain, it would make direct tampering in some form more likely, I think, since at least 1 and 3 should be expected to do the job "properly".

It seems likely to me though, that most modern drivers would keep the entire FAT table in RAM and write the parts that change back to both on-drive copies. Thus, even if there were at one time a difference, it might have been quickly and silently "fixed" during normal usage. Note that this is only an educated guess.

In the end, it's very hard to know with any certainty without further information about the circumstances, and even then it's likely guesswork. Ideal circumstances would be if you could reliably recreate the problem. Then I'd compare the "before" and "after" FAT tables (and FAT headers) to see exactly what had been changed and to what, looking for hints in the placement and content of changes.

Your FAT32 has been corrupted for some reason. My usb stick often ends up with a corrupted FS because I am currently debugging a usb host issue on an ARM platform. After my tests, I issue following commands from my desktop machine (Ubuntu 11.04):

```
sudo fsck.msdos -aw /dev/sdb1
```

Ref: http://askubuntu.com/questions/31614/how-to-delete-edit-files-from-readonly-filesystem