Ist es wahr, dass Linux Filesysteme (ETX3, XFS, ZFS, BTFS) gekippte Bits erkennen und reparieren können? Bei Hardwareeinsatz, zusätzliche Prüfbit(s) bei Speicherriegel, kann ich das eher verstehen als bei Filesystemen.

Es ist teilweise wahr: Während moderne Dateisysteme wie Z**FS** und **Btrfs** gekippte Bits (Bit Rot) sowohl erkennen als auch reparieren können, verfügen ältere Systeme wie **EXT3** und **XFS** nur über eingeschränkte oder gar keine Mechanismen für die Datenebene.

Hier ist der Vergleich der Dateisysteme (Stand 2026):

Dateisystem	Erkennung von Bit-Fehlern	Reparatur (Self-Healing)	Mechanismus
ZFS	Ja (Daten & Metadaten)	Ja (bei Redundanz)	Prüfsummen für jeden Datenblock; automatischer Abgleich beim Lesen.
Btrfs	Ja (Daten & Metadaten)	Ja (bei Redundanz)	Prüfsummen (Hashing) pro Block; Reparatur via RAID-Kopie oder `dup`-Modus.
XFS	Nur Metadaten	Nein (für Daten)	Prüfsummen sichern seit Version 3.2.3 primär die Struktur (Metadaten), nicht den Dateiinhalt.
EXT3 / EXT4	Nein	Nein	Bietet keine Ende-zu-Ende-Prüfsummen für Dateninhalte.

Funktionsweise im Vergleich zur Hardware

Der Vergleich mit **ECC-Speicher (RAM)** ist passend, aber die Ebene ist eine andere:

- **ECC-RAM:** Schützt Daten während der Verarbeitung im flüchtigen Speicher durch zusätzliche Prüfbits auf dem Riegel selbst.
- **ZFS/Btrfs:** Schützen Daten während der langfristigen Lagerung auf der Festplatte. Sie speichern zu jedem Datenblock eine kryptografische Prüfsumme (Hash) an einer anderen Stelle auf dem Datenträger.

Warum Redundanz für die Reparatur nötig ist

Ein Dateisystem kann ein gekipptes Bit zwar mittels Prüfsumme **erkennen**, es aber nur **reparieren**, wenn es eine unbeschädigte Kopie der Daten gibt.

- **Ohne Redundanz:** Das Dateisystem meldet lediglich einen E/A-Fehler (Input/Output Error), anstatt korrupte Daten kommentarlos auszuliefern.
- **Mit Redundanz (z.B. Mirror/RAID):** Das System erkennt die Diskrepanz, holt die korrekten Daten von der Spiegelplatte und überschreibt den defekten Block automatisch ("Self-Healing").