🔥 Burning Raspberry Pi Images
(Without Losing Your Mind)

Writing SD cards safely with `dd`
…and understanding why it *looks* broken when it isn't

🎯 Goal

Reliably write Raspberry Pi OS images to SD cards:

- No corruption
- No panic reboots
- No yanking cards early
- No superstition

⚡ TL;DR (Muscle Memory)

```bash
xz -dc image.img.xz | pv | sudo dd of=/dev/sdX bs=4M
sync
```

🧠 If `/proc/diskstats` is moving → **walk away**

🚨 Absolute Rules
(Do Not Skip)

❌ Rule 1
Never write a **compressed** image directly

✅ Rule 2
Always write to the **whole device**

- `/dev/sdX` ✔️
- `/dev/sdX1` ✖

🦃 Rule 3
Unmount before writing

❌ Rule 4
Do **NOT** use `conv=fsync`

🔁 Rule 5
Always run `sync`

👀 Rule 6
Disk activity beats screen output

📦 Why You Can't `dd` a `.xz`

Compressed images are **not block images**.

You must:
- Decompress to a file **or**
- Stream-decompress

✅ Correct Patterns

```bash
xz -dk image.img.xz
dd if=image.img of=/dev/sdX
```

or

```bash
xz -dc image.img.xz | dd of=/dev/sdX
```

🤯 "`dd` Hung!" — No, It Didn't

Flash media has **two phases**:

1. 📊 Bulk write (visible progress)
2. 🧹 Cache flush (no output)

During phase 2:
- CPU mostly idle
- Disk LED may stop
- Terminal frozen
- Ctrl-C ignored

⏱️ This can take **minutes**

🕰️ The Sane Workflow

````
---

## 🔍 Step 1
### Identify the Device (CRITICAL)

```bash
lsblk -o NAME,SIZE,MODEL,TYPE,MOUNTPOINT
```

Example:

```
sdi   29G  SD32G  disk
├─sdi1
└─sdi2
```

🎯 Target: **`/dev/sdi`**

---

## 🦅 Step 2
### Unmount Everything

```bash
sudo umount /dev/sdi*
```

Unmounting prevents:
- Silent corruption
- Kernel write conflicts
- “Why won’t this boot?”

---

## 🚀 Step 3
### Write with Feedback

```bash
xz -dc 2025-12-04-raspios-trixie-armhf.img.xz \
| pv \
| sudo dd of=/dev/sdi bs=4M
```

Why this is ideal:
- No temp `.img`
- `pv` shows progress + rate
- Clean exit when done

---

## 🧹 Step 4
### Flush (This Is the Long Wait)

```bash
sync
```
````

🔴 Silence is **expected**
🔴 Cursor not returning is **normal**

👀 How to Tell It's Still Working

📈 Watch Disk I/O (Best Signal)

```bash
watch -n 1 "grep sdi /proc/diskstats"
```

If numbers change → **still alive**

🧠 Watch Dirty Buffers

```bash
watch -n 1 "grep -E 'Dirty|Writeback' /proc/meminfo"
```

Dirty ↓ → flushing in progress

📝 Watch Kernel Messages

```bash
sudo dmesg -w
```

```
Synchronizing SCSI cache
```

This is the "don't touch it" phase.

❌ Things That Break Cards

- ❌ Ctrl-C during flush
- ❌ Pulling the card early
- ❌ Writing to `/dev/sdX1`
- ❌ Using `conv=fsync`
- ❌ Assuming silence == failure

🔣 Why `conv=fsync` Is a Trap

It:

- Forces flush *inside* `dd`
- Hides progress
- Blocks signals
- Feels “hung” forever

✅ Better Pattern

```bash
dd ... && sync
```

Same safety, better behavior.

🧰 Optional Tools (Highly Recommended)

```bash
sudo zypper install pv sysstat
```

- `pv` → progress & ETA
- `iostat` → deep device insight

✅ Safe Removal Check

```bash
sync && echo "Safe to remove"
```

If that prints — you’re done.

🧠 Rule to Remember Forever

> **If `/proc/diskstats` is moving, nothing is wrong.**

🏁 Canonical Command (Final)

```bash
xz -dc image.img.xz | pv | sudo dd of=/dev/sdX bs=4M
sync
```

🔥 Write it
🕯️ Trust it
🚶 Walk away