

Troubleshooting Guide: Runaway CPU Usage After Video Encoding

This guide explains why your CPUs remained at 100% usage after completing your MP4 encoding tasks, even after running your Restore command, and provides solutions to monitor and fix this without rebooting.

1. Why the CPU Kept Racing

The Restore command changes system power settings (telling Linux it is allowed to sleep again); it does not force active or hung applications to stop running. When video encoding ends, processor usage should drop to near 0% within seconds. If it stays at 100%, it is usually due to:

- * Runaway Background Processes: Encoding engines (like FFmpeg or HandBrake workers) failing to shut down cleanly, trapping themselves in an infinite processing loop.
- * Zombie Threads: The main interface closes, but the operating system fails to "reap" the child threads, leaving them spinning infinitely in the Linux kernel.

2. Dynamic Monitoring Tools (Terminal-Based)

Instead of rebooting, use these terminal commands to see exactly what is running:

- * top : The native Linux tool to view system resource usage in real time.
- * htop : An interactive, color-coded, easier-to-read version of top (highly recommended).

3. Manual Intervention (How to Fix Without Rebooting)

Step 1: Identify the Culprit

Open your terminal and type "top" or "htop". Look at the very top lines to find the application consuming the highest %CPU. Note its exact name (e.g., ffmpeg, ghb, HandBrakeCLI).

Step 2: Kill the Runaway Process

Forcefully terminate the stuck process by passing its name to the killall command with the -9 (SIGKILL) flag:

```
killall -9 ffmpeg
```

(Replace "ffmpeg" with the exact process name you found in Step 1).

4. Automation: Updating Your .bashrc Function

To prevent this from happening manually in the future, you can embed an automatic cleanup sequence directly into your workstation_restore() function. This will automatically terminate stuck encoders the moment you run your restore command.

Open your .bashrc file:

```
xed ~/.bashrc
```

Update your function to match this version:

```
workstation_restore() {
    # 1. Turn Privacy Mode OFF (Restore Original Prompt)
    if [ "$color_prompt" = yes ]; then
        export PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
    else
        export PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
    fi
    export PRIVACY_MODE=off

    # 2. Restore Default Sleep Settings
    xset s on +dpms
```

```
gsettings set org.cinnamon.desktop.session idle-delay 900
gsettings set org.cinnamon.desktop.screensaver lock-enabled true

# 3. Optional Automation: Automatically clear runaway encoding processes
# Remove the "#" from the line below if you want it to auto-kill ffmpeg
# killall -9 ffmpeg 2>/dev/null

echo "Status: RESTORED (Privacy DISABLED & Default sleep active)"
}
```

After saving and closing Xed, reload your terminal profile to apply the changes:

```
source ~/.bashrc
```

Use code with caution.

Would you like to explore adding multiple program variants (like HandBrake and FFmpeg simultaneously) to the automated cleanup step, or should we look at creating a separate custom alias purely dedicated to system monitoring?

- - Medium
 - XED: The Lightweight Text Editor for Xcode | by Shashi Jagtap | XCBlog
 - Dec 16, 2018 – What is XED The xed is Xcode Text Editor Invocation tool which means we can use xed for opening, editing, creating files inside th...

I like the idea of a seaparate custom alias. I will note this for the futre, and will bring it up again. Thank you, you have been very hgelpful

You are very welcome! Saving that for a separate alias is a smart approach—it keeps your tools organized and prevents accidental file closures while you are still working.

When you are ready to set that up in the future, just let me know. We can build a custom shortcut (like `check_cpu` or `mon`) that instantly launches a filtered view of only your heaviest tasks, making it incredibly fast to spot and stop stuck encoders.

Have a great rest of your day, and happy video encoding!