ChatGPT

Safely Configuring and Replacing the Bootloader on Arch (Hyprland Setup)

Arch Linux (with the Hyprland compositor) can use any standard bootloader, but careful configuration is needed to avoid disrupting system functions. This guide covers choosing a bootloader, switching from GRUB to an alternative, tweaking kernel parameters to fix Bluetooth and reboot issues, and making the boot process unobtrusive (brief boot menu and no lingering splash screens). We include step-by-step instructions, configuration examples, and references to Arch Wiki and other authoritative sources for reliability and reproducibility.

Bootloader Options for Arch + Hyprland

Common Bootloaders: Arch Linux supports **GRUB**, **systemd-boot**, **rEFInd**, and others. All of these can boot Arch/Hyprland reliably, but they differ in complexity and features:

- **GRUB (GRand Unified Bootloader):** The traditional, widely-used bootloader on Linux. It supports both legacy BIOS and UEFI systems and can handle complex setups (multiple OSes, many filesystems, encryption unlocking, etc.). GRUB is the default on many distros and "just works" for most cases [1]. However, it requires generating a config (e.g. via `grub-mkconfig`) after kernel updates or config changes, and its menu and theming are more complex. Boot speed difference is negligible in practice [1]. GRUB is a safe choice if you dual-boot with non-UEFI OSes or need its advanced features.

- **systemd-boot (formerly "gummiboot"):** A simple UEFI boot manager that comes with systemd (and thus is included in Arch by default). It works only on UEFI systems with GPT. Configuration is extremely simple – you create a text file entry per kernel in `/boot/loader/entries/` and a minimal `/boot/loader/loader.conf` [2] [3]. There's no complex scripting; it directly boots the specified kernel image. Pros: it's lightweight and fast, and if you already use systemd it integrates well. It also works fine with Windows dual-boot on UEFI systems [4]. The Arch community notes that *"systemd-boot is simpler and you will realistically not notice any difference [compared to GRUB] when installed – both can do everything you need"* [1]. **Important:** systemd-boot relies on the UEFI to read kernel files, so your kernel and initramfs must be on the EFI System Partition (ESP) (usually by mounting the ESP at `/boot`). This may require adjusting your partition setup (discussed below). Secure Boot support requires manually signing kernels or using unified kernel images, since systemd-boot doesn't have a shim like GRUB.

- **rEFInd:** A graphical UEFI boot manager with auto-detection capabilities. rEFInd automatically scans for bootable OSes/kernels across ESP and other partitions, presenting them in a GUI menu with icons. It's great for dual- or multi-boot scenarios (Windows, macOS, multiple Linux distros) because it can detect many OSes without manual config. It includes drivers to read Linux filesystems (ext4, Btrfs, etc.), so it can load kernels from a Linux `/boot` partition even if not on the ESP [5]. rEFInd also supports theming and mouse/keyboard navigation. Compared to systemd-boot, it's more

feature-rich: for example, *rEFInd can display a graphical menu, load UEFI drivers (including filesystem drivers), boot legacy BIOS OSes on UEFI systems (e.g. Windows on Macs), and automatically scan for new boot entries* [5] . The downside is slightly more complexity and no built-in "hidden menu" timeout – rEFInd always shows a menu at boot unless you configure a very short timeout. It's an excellent choice if you want a polished GUI boot selector or need its auto-detection, but for a single-OS Arch + Hyprland system it might be overkill.

rEFInd's default GUI boot menu auto-detects installed OSes and shows each as an icon. In this screenshot, rEFInd found multiple boot options (Linux, Ubuntu, macOS, Windows). It provides a user-friendly interface by default, whereas systemd-boot shows a simple text list and GRUB's menu is text-based or basic gfx. [6]

Summary – which to choose? For a pure Arch Linux/Hyprland system on UEFI, many users prefer the simplicity of **systemd-boot** (fewer moving parts, straightforward config) [1] . If you already have GRUB set up and it's working, there's no urgent need to switch – GRUB is perfectly fine and stable [1] . But if you want a cleaner boot setup and faster configuration, systemd-boot is an elegant choice. Use **rEFInd** if you value a nice GUI menu or have a complex multi-boot; otherwise, its additional features might not be needed for a single Arch installation. All three support booting Arch with Hyprland equally well. The remainder of this guide will assume you are interested in switching from GRUB to systemd-boot (as an example), but the general principles (careful ESP configuration, kernel parameters, etc.) apply to rEFInd as well.

Safely Replacing GRUB with systemd-boot

If you decide to replace GRUB with systemd-boot, you must be careful to avoid rendering your system unbootable during the transition. Below is a safe procedure to switch bootloaders reliably:

1. **Verify UEFI Mode:** systemd-boot only works in UEFI mode. Ensure your system was installed in UEFI mode (you should have an EFI System Partition, typically mounted at `/boot` or `/boot/efi`). For example, run `lsblk -f` and identify the FAT32 partition (ESP). It should be mounted. If your Arch is in BIOS/Legacy mode, stick with GRUB or convert to UEFI first.

2. **Mount the ESP at `/boot`:** For systemd-boot, the kernel and initramfs files should reside on the ESP. On Arch with GRUB, often the ESP is mounted at `/boot/efi` and `/boot` is part of the root filesystem (ext4). You'll need to mount the ESP to `/boot` instead. Do the following:

3. Edit `/etc/fstab`: change the mount point of the ESP from `/boot/efi` to `/boot` (or add a new entry mounting the ESP to `/boot`). Ensure the ESP partition has the type GUID "EFI System" (gdisk type `EF00`), otherwise `bootctl` will complain [7].
4. Reboot or temporarily mount the ESP to `/mnt` and copy your existing kernel files. *Tip:* Easiest is to do this from an Arch live USB to avoid conflicts. Alternatively, you can mount the ESP to a temp location, copy `vmlinuz-linux`, `initramfs-linux.img`, and any `*-ucode.img` to it, then unmount and remount it at `/boot`.

5. Update `/etc/mkinitcpio.conf` (if needed) so that future initramfs updates output to `/boot` (on Arch this is usually the default).

6. **Install systemd-boot:** Once the ESP is mounted at `/boot` and contains your kernel images, install the bootloader by running:

   ```
   # bootctl install
   ```

   This copies the systemd-boot EFI binaries to the ESP (`/boot/EFI/systemd/systemd-bootx64.efi` and sets up `/boot/loader/`). If `bootctl` reports *"Couldn't find EFI system partition"* or *"wrong partition type"*, it means the ESP isn't properly mounted or not labeled as an ESP [8] [7]. In that case, ensure the partition type is correct (use `gdisk` to set type EF00 if using GPT [7]) and try again.

7. **Create boot loader entries:** In `/boot/loader/entries/`, create a file (e.g. `arch.conf`) with your Arch Linux kernel details. For example:

   ```
   title   Arch Linux
   linux   /vmlinuz-linux
   initrd  /initramfs-linux.img
   initrd  /intel-ucode.img      # (if you have a CPU microcode update, e.g.
   Intel)
   options root=UUID=<YOUR-ROOT-PART-UUID> rw quiet splash
   ```

   Use the UUID of your root filesystem (find it with `blkid`). The `quiet splash` are optional kernel parameters (quiet suppresses verbose boot messages, and `splash` is used by Plymouth – include it if you use a splash screen). This entry file is the equivalent of GRUB's menuentry – it tells systemd-boot which kernel to boot and with what options. You should also edit `/boot/loader/loader.conf` to set a default entry and timeout (see next section for making the menu brief). For example:

```
default  arch.conf
timeout  3
console-mode max
```

A 3-second timeout is a safe starting point (we will adjust/hide it later). `console-mode max` ensures the text mode uses the highest resolution (optional).

8. **Do *not* remove GRUB yet:** At this stage, you have both GRUB and systemd-boot installed. Reboot the system and enter your firmware's boot menu (often by pressing a key like F11/F12/ESC at power-on) to choose the "Linux Boot Manager" (systemd-boot) if it's not already default. If everything is set up correctly, you should see systemd-boot's menu and be able to boot Arch via the new entry. If it fails, you can always reboot and select the old GRUB entry from UEFI (or re-enable GRUB from BIOS) to get back in.

9. **Uninstall or cleanup GRUB (optional):** Once systemd-boot is confirmed working, you can remove GRUB. This involves uninstalling the `grub` package and deleting GRUB's files from the ESP. For example:

```
# pacman -Rns grub
# rm -rf /boot/EFI/GRUB
```

Also use `efibootmgr` to remove the GRUB boot entry from NVRAM if desired (this is optional – leaving it won't harm anything). **Caution:** Only remove GRUB after you have a working alternative. Many users have made the mistake of deleting GRUB or reformatting the ESP before installing the new bootloader, resulting in an unbootable system [9] [10]. Always keep a live USB handy for recovery. In case of trouble, you can boot a live USB, mount your partitions, and reinstall a bootloader in chroot to recover.

Following the above steps, you'll transition to systemd-boot without breaking your system. One Arch user who switched from GRUB to systemd-boot noted that the solution was to mark the partition correctly as an ESP, install systemd-boot, **and reinstall the Linux kernel package to restore missing kernel files on the ESP (since they had reformatted it)** [9]. In summary, ensure the ESP is properly set up and contains the needed boot files before rebooting into the new bootloader.

Bootloader Menu: Brief and Hidden by Default

Once you have your bootloader of choice configured, you likely want the boot menu to be as unobtrusive as possible – **show it only briefly (or not at all) during startup**, and definitely **not during shutdown/reboot** (the bootloader shouldn't appear on shutdown in any case, unless something is misconfigured). Here's how to configure each bootloader for a minimal, accessible menu:

- **systemd-boot (Hide menu unless needed):** In systemd-boot's `loader.conf`, you can set `timeout 0` to effectively hide the menu. A timeout of 0 **disables** the countdown – the default entry boots immediately [11]. If the menu is disabled, you can still access it by pressing and holding a key

right after you power on (before the OS starts). The systemd-boot manual notes: *"If the timeout is disabled (set to 0), the default entry will be booted immediately. The menu can be shown by pressing and holding a key before systemd-boot is launched."* [11] . In practice, this means if you ever need the menu (to select another kernel or recovery option), you can hold **Space** (or basically any key) during boot to force the menu to appear. This gives a very clean startup – no menu is shown by default at all. If you prefer a small delay instead of zero, you could set `timeout 1` (one second). **Note:** systemd-boot can remember a previously set timeout via an EFI variable. If you find the menu still showing for 3s even after setting 0, you may need to adjust it manually in the menu (press `-` to decrease the timeout to 0) or clear the saved EFI boot timeout variable [12] [13] .

- **GRUB (Hidden menu with optional keypress):** GRUB can be configured to not display the menu unless a key is pressed, by using **hidden timeout** mode. In `/etc/default/grub`, set:

```
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=1
```

This will make GRUB wait 1 second *before* showing the menu. If during that 1 second you press **ESC** (or **SHIFT**, or F4 depending on firmware) [14] , the menu will be shown – otherwise, if no key is pressed, it will boot the default entry immediately after the timeout expires without ever showing the menu [14] . A one-second hidden timeout is usually enough to catch with a key if you're quick; you could set it to 2–3 seconds if you want a larger window (or 0 for no delay at all, though 0 means you essentially *cannot* interrupt it easily). Make sure `GRUB_DEFAULT` is set to your desired default OS (0 for first entry typically) [15] . After changing `/etc/default/grub`, run `grub-mkconfig -o /boot/grub/grub.cfg` to apply changes. For example, a minimal config for hidden menu:

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=1
```

With this, you will **not** see the GRUB menu on a normal boot. If you need to access it (say to choose an older kernel or another OS), reboot and **hold SHIFT** (for BIOS grub), or press ESC/F4 for UEFI grub, during the boot process to invoke the menu [14] . GRUB's manual confirms: when set to `hidden`, GRUB will *"wait for the timeout before displaying the menu. If ESC or F4 are pressed, or SHIFT is held down during that time, it will display the menu and wait for input..."* [14] . This achieves an "unobtrusive but accessible" menu.

- **rEFInd (brief display):** rEFInd doesn't support a truly hidden menu on keypress – it will always paint some menu. The best you can do is set a very short timeout so it auto-boots quickly. In `refind.conf`, the `timeout` parameter controls how long (in seconds) the menu waits. Setting `timeout 1` will show the rEFInd menu but only for ~1 second before it boots the default entry [16] . You can't fully hide the GUI, but you *can* make it less noticeable (for example, by using text mode or a minimal theme). rEFInd's author states: *"There's no option to hide rEFInd [completely]. You could set rEFInd's timeout to 1, which will pause for 1 second and then boot the default entry. If you don't want to see the rEFInd screen, you could use a custom theme that hides almost everything or use text mode, so you'd just see a text display for 1 second. In any case, a 1-second display is a small price to pay."* [16] [17] .

In practice, many users set `timeout 2` or `timeout 5` in rEFInd for dual-boot scenarios so you have a moment to pick another OS, but you can use 1 for near-instant boot. Additionally, rEFInd offers an option `shutdown_after_timeout` – if set to `0` (false, the default) it will reboot immediately after timeout. Ensure that is off (so it doesn't "linger" at shutdown – though rEFInd is only in memory during boot, not at shutdown).

Bootloader on Shutdown/Reboot: The bootloader should **not** appear during shutdown or reboot sequences. If you ever see a bootloader menu or text while shutting down, that indicates an unusual issue. One possibility is if the system is *restarting* instead of fully powering off (in which case on "shutdown" it comes right back to the bootloader). We'll address that in the reboot fixes below. But normally, with the configurations above, the boot manager is only invoked at system startup.

Kernel Parameters to Fix Bluetooth and Reboot Issues

Beyond the bootloader menu behavior, certain **kernel parameters** can solve the specific problems the user noted:

1. Preventing Bluetooth Instability (Power Management Fix)

If you experience **Bluetooth instability** (e.g. Bluetooth adapter randomly disconnecting or turning off, especially on a laptop) after switching bootloaders or updating kernels, it might be due to aggressive power-saving on the Bluetooth module. A known solution is to **disable USB autosuspend for Bluetooth**. The Linux Bluetooth driver (`btusb`) has a module option to control this. You can add the kernel parameter `btusb.enable_autosuspend=n` to **turn off autosuspend** for the Bluetooth USB device [18]. This prevents the Bluetooth chip from being powered down to save power (which on some hardware causes instability or even sudden power-off of the device).

How to add this parameter: - **GRUB:** edit `/etc/default/grub` and append `btusb.enable_autosuspend=n` to `GRUB_CMDLINE_LINUX_DEFAULT` (or `GRUB_CMDLINE_LINUX`). For example:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash btusb.enable_autosuspend=n"
```

Then run `sudo grub-mkconfig -o /boot/grub/grub.cfg`. Reboot for it to take effect. - **systemd-boot:** edit your `arch.conf` (entry file) and add `btusb.enable_autosuspend=n` inside the `options` line, just like other kernel options (space-separated). - **rEFInd:** if you have a manual stanzas in `refind.conf`, add `btusb.enable_autosuspend=n` to the kernel options. If rEFInd is just auto-detecting your vmlinuz, you can edit the kernel's embedded command line or use a drop-in conf to add options.

The Arch Wiki notes that Bluetooth issues *"might be caused by power saving measures, in which case adding the kernel parameter* `btusb.enable_autosuspend=n` *is a potential solution."* [18] This is a safe tweak – it just disables one USB power-saving feature for the BT module. Alternatively, some users achieve the same by creating a modprobe config to disable autosuspend or using tools like TLP to blacklist the BT device from

autosuspend. But the kernel param is the quickest fix since it applies early. After adding it and rebooting, your Bluetooth should be more stable (no sudden drop-outs when idle).

Troubleshooting: To verify it's applied, you can run `cat /proc/cmdline` to see the active kernel command line – it should list `btusb.enable_autosuspend=n`. Also, if you modprobe the module manually you can check `modinfo btusb` to see the default of `enable_autosuspend` (usually default `Y` or `1`, which we override to `n`). With the parameter set to `n`, the Bluetooth adapter will remain powered during runtime. If instability persists, ensure your BlueZ daemon is up to date and not blocked by another issue. In rare cases, different chipsets have their own quirks – e.g. some Intel Bluetooth devices had firmware issues requiring updates, but that's beyond bootloader scope.

2. Ensuring Reboot Actually Reboots (No Accidental Power-off)

The issue where `reboot` **commands power off the machine** (or otherwise don't reboot correctly) usually comes down to ACPI or firmware quirks. Normally, `systemctl reboot` should do a full reboot. If instead your system shuts down (powers off) when you attempt to reboot, the Linux kernel's *reboot method* might need adjustment. The kernel provides a parameter `reboot=` to control how the reboot is signaled to hardware. Common values are: `reboot=acpi` (use ACPI reboot), `reboot=bios` (legacy BIOS int19h reboot), `reboot=efi` (use UEFI reboot call), etc [19] [20]. By default, the kernel tries a sequence of methods; on most modern PCs ACPI or UEFI works. But if your PC is powering off instead, it might be invoking the wrong method.

Fix options: - First, **make sure no erroneous reboot parameter is set**. Check your kernel cmdline (`/proc/cmdline`) for any `reboot=` option. If some guide had you add `reboot=power_off` or similar, remove it – that explicitly makes the reboot act as poweroff. It's uncommon to use that parameter unless intentionally wanted. - If nothing obvious, try explicitly setting a different reboot method. A safe bet is to add `reboot=acpi` (to force ACPI reboot) or `reboot=efi` (to use UEFI runtime services reboot). For example, one forum suggests: *"Will your system reboot properly if you use either of these kernel parameters:* `reboot=bios` *or* `reboot=acpi`*?"* [19] – in many cases one of those can resolve reboot issues on certain laptops. Another source notes that adding `reboot=efi` fixed "soft reboot" problems on some systems [20]. You can try one at a time (add to your bootloader kernel options similar to above, then reboot and test). - A related ACPI quirk: If you dual-boot with Windows, **disable Windows Fast Startup** in Windows. Fast Startup can leave devices in a state that confuse Linux on reboot. In one case, an HP laptop had reboot problems that were solved by disabling Windows fast startup and blacklisting a buggy driver, not strictly by boot param alone [21] [22]. So ensure that's off if applicable.

For Arch with systemd, typically `systemctl reboot` should "just work." If it doesn't, and the kernel param doesn't help, observe the console on reboot: does it say "reboot: Power down" (meaning it's doing a shutdown) or does it hang? If you see **"reboot: Power down"** message during an attempted reboot, it implies the kernel thought it should shut off. That could be triggered by `reboot=poweroff` or by systemd misinterpreting the action. Ensure you are actually using the correct command (`systemctl reboot` or the GNOME/KDE reboot which call it). Some desktop environments had old issues where reboot would call the wrong action – but on a modern Arch/Hyprland setup that's unlikely.

In summary, adding a kernel parameter like `reboot=acpi` (or `reboot=bios` on some hardware) often *"fixes 'soft reboot' issues"*, causing the system to truly restart instead of halt [20] [19]. You can leave the chosen

param in place permanently if it works. It doesn't negatively affect anything else; it just picks a specific reboot mechanism. Example: to set via GRUB, add `reboot=acpi` to the GRUB CMDLINE and regen config; via systemd-boot, put it in the options line.

3. Other Kernel Option Tips and Module Tweaks

While the question's focus is Bluetooth and reboot, a couple of additional tips:

- **Validate your kernel parameters:** A common mistake when editing bootloader configs is typos or forgetting to regenerate GRUB's config. Always verify by checking `/proc/cmdline` on the running system to see all active parameters. If a parameter is not recognized by the kernel, you'll typically see a message in `dmesg` like "unknown kernel parameter ignored". Running `dmesg | grep -i kernel` after boot can show if any parameter was rejected. This helps catch mistakes.

- **Bluetooth module power tweaks:** Aside from `btusb.enable_autosuspend=n`, if you still have BT trouble, check `powertop` or TLP settings. Ensure the Bluetooth device isn't being autosuspended by another power-management tool. You can also try the experimental `btusb.disable_autosuspend` (older name) or even disabling USB autosuspend globally (via kernel `usbcore.autosuspend=-1`, though that's overkill). There are also chipset-specific options (e.g. some Realtek BT drivers have parameters). The Arch Wiki's Bluetooth page and forums are good resources if a particular adapter (e.g. Intel AX210) has known issues – sometimes a firmware update via `linux-firmware` package is the cure.

- **Graphics and Plymouth:** If you use a graphical bootsplash (Plymouth), ensure your kernel options include the appropriate driver and `splash`. For instance, Hyprland users often use modesetting; the Plymouth hook might require `i915.modeset=1` (for Intel) or similar if not auto-detected. In one case, Plymouth only showed on shutdown and not on boot until the Intel graphics module was added to initramfs (module `i915`) [23] [24]. Just keep this in mind if you tinker with splash.

(Optional) Suppressing Unwanted Boot-Time Visuals (Plymouth Splash)

If you have Plymouth (a bootsplash) set up and notice that a **shutdown splash screen lingers**, or you get a blank screen that hangs for a few seconds at power-off, there are a couple of considerations:

- Plymouth is normally configured via the initramfs for the boot splash, and also integrates with systemd for shutdown. If the splash is *"lingering too long"* at shutdown, it could be that the system is actually halted and waiting for power-off, leaving the last Plymouth image on screen. By default, systemd will switch to a text console on shutdown; Plymouth, however, tries to display a "shutdown splash" until the very last moment. On a very fast shutdown, you might briefly see the splash or a flicker of it. To avoid this, you can *disable the Plymouth shutdown screen*. On Arch, one way is to mask the `plymouth-poweroff.service` or similar. For example:

```
systemctl mask plymouth-poweroff.service plymouth-reboot.service
```

This ensures Plymouth doesn't try to hijack the console on shutdown/reboot. The result is you'll see either a blank screen or the usual systemd shutdown text instead of the splash.

- Another approach is to adjust Plymouth's `ShowDelay` or `DeviceTimeout` so that it doesn't bother showing a splash if the shutdown is very quick. But masking the services is a more straightforward "don't show at shutdown at all" solution.

- **Halt vs Poweroff:** Recall that if you use `systemctl halt`, the system will stop OS operation but not actually cut power – this *will* leave whatever was last on the screen (possibly Plymouth or console text) frozen on screen until you manually power-off. So always use `poweroff` or `shutdown -h now` (which calls poweroff) for a full shutdown. The Arch ARM forum discussion shows a user trying to have an image remain on screen on halt [25] [26] – generally on a PC you want the opposite (no image, just power off). Make sure you aren't inadvertently halting instead of powering off.

- **Flicker-free boot:** If your goal is a completely clean boot with no ugly visuals, you might have Plymouth at boot with a nice theme and want it *gone instantly* at shutdown. In addition to masking Plymouth's shutdown services, you can compile your initramfs without the Plymouth shutdown hook (depending on how Plymouth is integrated on Arch – the AUR package may include a shutdown handler). Also, use the `quiet` kernel param to suppress kernel messages, so you don't see text after Plymouth quits.

If Plymouth still shows something on shutdown that bothers you, you can always remove Plymouth entirely. It's not essential – it's purely cosmetic. Hyprland itself doesn't care about Plymouth. Removing it (and the `splash` param) will give you simple text messages on boot/shutdown (or a blank screen if quiet). This is a matter of preference.

Finally, note that a fast NVMe-based system might shut down so quickly that any Plymouth splash might only appear for a split second. Some users have reported the splash appears *only* on shutdown but not on boot (often due to a misconfiguration that prevented it on boot) [27] [28]. The fix was adding the GPU module to initramfs as mentioned. So if you *want* the splash on boot, ensure it's correctly set; if you **don't** want it on shutdown, use the methods above to disable it there.

Troubleshooting Summary: Always keep a live USB around when making bootloader changes. Check the Arch Wiki for pages on Arch boot process (which has a feature comparison of bootloaders) and the specific bootloader's wiki page (e.g. GRUB, systemd-boot, rEFInd). Official docs and forums can provide additional tips if your hardware is quirky. The steps and parameters given above have been tested by the community and should cover the common issues. By carefully configuring the bootloader and kernel options, you can achieve a smooth experience on Arch + Hyprland – with a fast, silent boot, stable Bluetooth, and reliable reboot/shutdown behavior.

Sources:

- Arch Linux Reddit – discussion on GRUB vs systemd-boot (user experiences on simplicity and dual-boot) [1] [3]
- Arch Linux Forums – switching from GRUB to systemd-boot (step-by-step and pitfalls) [8] [9]

- Arch Wiki (Bluetooth) – autosuspend causing Bluetooth issues [18]
- Arch Wiki (kernel parameters) and forum – using `reboot=acpi` or others to fix reboot problems [19] [20]
- Arch Wiki / GRUB Manual – configuring hidden GRUB menu with key reveal [14] [15]
- rEFInd author's notes – no true hide option, use short timeout [16] and feature comparison with systemd-boot [5].
- Reddit Arch discussion – Plymouth behavior and fixes (ensuring it starts on boot, not just shutdown) [23] [24].

---

[1] [2] [3] [4] I am moving from ubuntu to arch with hyprland, some youtube tutorials use grub bootloader and some systemd, so which should i pick? : r/archlinux
https://www.reddit.com/r/archlinux/comments/1d0vsq2/i_am_moving_from_ubuntu_to_arch_with_hyprland/

[5] [6] [16] [17] [Workaround] Hide rEFInd boot manager? / Newbie Corner / Arch Linux Forums
https://bbs.archlinux.org/viewtopic.php?id=151440

[7] [8] [9] [10] [SOLVED] I can't switch from grub to systemd-boot / Newbie Corner / Arch Linux Forums
https://bbs.archlinux.org/viewtopic.php?id=295584

[11] [12] [13] Show/hide systemd-boot menu - Unix & Linux Stack Exchange
https://unix.stackexchange.com/questions/542438/show-hide-systemd-boot-menu

[14] [15] [SOLVED] Skip grub menu / System Administration / Arch Linux Forums
https://bbs.archlinux.org/viewtopic.php?id=273592

[18] Bluetooth - ArchWiki
https://wiki.archlinux.org/title/Bluetooth

[19] [21] [22] [SOLVED]Reboot/shutdown doesn't work (HP laptop) / Newbie Corner / Arch Linux Forums
https://bbs.archlinux.org/viewtopic.php?id=197674

[20] Fail to reboot after HAOS updates - homeassistant - Reddit
https://www.reddit.com/r/homeassistant/comments/1mpiitg/fail_to_reboot_after_haos_updates/

[23] [24] [27] [28] Plymouth not working correctly on Arch - shows up when shutting down, but not when booting up : r/archlinux
https://www.reddit.com/r/archlinux/comments/i7ag9a/plymouth_not_working_correctly_on_arch_shows_up/

[25] [26] Arch Linux ARM • View topic - Power off image
https://archlinuxarm.org/forum/viewtopic.php?f=57&t=7149