rinsmiles’ Guide to the VOID

v2020.09

rinsmiles´ Guide to the Void is a guide for installing and setting up a Void operating system:

voidlinux.org

It mainly, and merely intends to *guide* the user through the steps, providing hints and examples along the way, and it is complemented by the official Void documentation.

The installation is performed *from the ground up,* by bootstrapping the new system with the XBPS package manager from a console. After following through with this guide, the user will have a fully configured, reliable, performant and fundamental system which they can then tune and extend to their requirements.

This guide assumes that the user is familiar with the use of a shell, core utilities and GNU/Linux operating systems, and that they will take a look at specific documentation of any tool and feature with which they are not.

Index

Installation

0. Boot a base live image of Void and login as root.

1. Set keyboard layout and desired font

The default keyboard layout is `us`, and the default small font is `default8x16`.

Available layouts are located in `/usr/share/kbd/keymaps/`. You can set one with **loadkeys**.
Console fonts reside in `/usr/share/kbd/consolefonts/`. You can set one with **setfont**.

2. Set up filesystems

You will minimally need to set up a root partition, and an EFI system partition (ESP) under UEFI.

a) Identify the block devices to manage—you can list them with **lsblk**.

- If you already have an ESP on the target device and want to preserve it (for example, when dual-booting with a different OS on the same drive) you could use it as Void's ESP as well.
- If you are re-purposing a whole drive, you can wipe all signatures from the device with the **wipefs** command.

b) Create, as needed, a new partition table and the required partitions with **fdisk**.

- If wishing to use hibernation, create a swap partition as well.
- The ESP should be at least 200MB in size to allow for multiple kernel installs. Make sure to set the partition type of the ESP accordingly (typically named "EFI System").
- `fdisk` handles partition alignment automatically.

c) Format the created partitions with **mkfs**.

- *mkfs.vfat*, *mkfs.ext4* and *mkfs.btrfs* are some filesystem specific frontends for this purpose.
- For most UEFI implementations the ESP must be formatted as FAT. The appropriate FAT type (FAT16, FAT32) is determined automatically by `mkfs.vfat`.

d) Finally, **mount** the created partitions—you can mount the root partition to `/mnt/`, create the directory `/mnt/boot`, and then mount the ESP to such directory.

3. Connect to the internet

If you are on a wired network, you will likely already be connected to the internet. Verify this with a tool like `ping`, for example (hitting `Ctrl+C` to interrupt it):

```
# ping voidlinux.org
```

If you are on a wireless network, the connection will have to be configured:

i) Make sure the interface is not hard or soft blocked. Do the latter using **rfkill**:

```
# rfkill unblock wlan
```

ii) Check that the wireless interface is listed with the **ip link** command, then set it up:

```
# ip link set ⟨wireless_interface⟩ up
```

iii) Configure the wireless network using **wpa_supplicant**:

```
# wpa_passphrase ‹ssid› ‹passphrase› >> \
    /etc/wpa_supplicant/wpa_supplicant.conf
# wpa_supplicant -B -i ‹wireless_interface› \
> -c /etc/wpa_supplicant/wpa_supplicant.conf
```

iv) Disable **dhcpcd**'s Link-local IPv4 addressing and issue an interface rebind:

```
# echo 'noipv4ll' >> /etc/dhcpcd.conf
# dhcpcd -n
```

4. Install Void on the mounted filesystem

Set up the parameters that **xbps-install** will use to install the new system:

a) Define your preferred main repository mirror from which to download the packages, e.g:

```
# REPO=https://alpha.us.repo.voidlinux.org/current
```

Define any other repositories you need—e.g. the `non-free` *repo,* for proprietary firmware:

```
# REPONF=https://alpha.us.repo.voidlinux.org/current/nonfree
```

b) Set the architecture and C standard library implementation flavor to install, e.g:

```
# ARCH=x86_64
```

c) Select the packages to install. It can be as simple as installing the `base-system` and `efibootmgr` packages, or something more selective, such as:

Basic components, including base files, XBPS, runit, shell, kernel, firmware and other core utilities and libraries:

```
# PACKS='base-minimal linux kbd file ncurses libgcc libstdc++
```

... essential services to manage input, device and time-based events:

```
> acpid eudev snooze
```

... along with our selected filesystems' tools:

```
> dosfstools e2fsprogs (btrfs-progs)
```

... utilities to query and configure boot entries and system bus devices:

```
> efibootmgr pciutils (usbutils)
```

... those you need to set up networking:

```
> iproute2 dhcpcd (iwd|wpa_supplicant)
```

... documentation and utilities to access it:

```
> mdocml man-pages void-docs
```

... any extra firmware you need, if using the non-free repo enabling it for the new install:

```
> (intel-ucode void-repo-nonfree)
```

... and any additional software you may want to initially install, like an application to easily edit text files and utilities to test network connectivity:

```
> nano mtr'
```

Note: AMD device firmware is already included in the `kernel` meta-package. Microcode updates are applied automatically once configured by XBPS.

d) Issue the installation of the packages to the new root partition mount (here `/mnt`):

```
# XBPS_ARCH=$ARCH xbps-install -S -r /mnt -R "$REPO" \
... optionally:
> -R "$REPONF" \
... then:
> $PACKS
```

5. Chroot into the newly installed system

a) Mount the pseudo-filesystems needed for a chroot:

```
# mount --rbind /sys /mnt/sys   && mount --make-rslave /mnt/sys
# mount --rbind /dev /mnt/dev   && mount --make-rslave /mnt/dev
# mount --rbind /proc /mnt/proc && mount --make-rslave /mnt/proc
```

b) Copy the DNS configuration into the new root in case more packages are to be downloaded:

```
# cat /etc/resolv.conf > /mnt/etc/resolv.conf
```

c) And **chroot** to the new installation:

```
# chroot /mnt/
```

*Note: Having installed **void-docs** on the new system, you can now access Void's documentation directly using that command with the option "-r", which displays items as regular man pages.*

6. Configure the basics

a) Set the system's hostname:

```
# echo ‹system_hostname› > /etc/hostname
```

b) Edit `/etc/rc.conf` with your text editor of choice, setting the system's time settings and the console's default keyboard layout and fonts.

c) If installing a glibc distribution, uncomment desired locales in `/etc/default/libc-locales` and generate locale files by reconfiguring the `glibc-locales` package:

```
# xbps-reconfigure -f glibc-locales
```

You can then set specific locale options in `/etc/locale.conf`, as displayed by **locale**.

d) Set the root password:

```
# passwd
```

e) Configure `fstab`. This can be as simple as adding entries for the root and ESP partitions to the existing file, setting the appropriate parameters for each. You can look up current mount information in `/proc/mounts` or by using **findmnt**:

```
# findmnt -kl -o SOURCE,UUID,TARGET,FSTYPE,OPTIONS --real
```

For example, the file may result like so:

```
/etc/fstab
```

```
# <file system>      <dir>      <type>     <options>              <dump>   <pass>
# /dev/sda2
UUID=a1b2(...)       /          ext4       defaults                   0        1
# /dev/sda1
UUID=9Z8Y(...)       /boot      vfat       defaults,umask=022,utf8    0        2
# /tmp tmpfs
tmpfs                /tmp       tmpfs      defaults,nosuid,nodev      0        0
```

f) If you used another mirror in place of the default one for installation, add that to your XBPS configuration by overriding the corresponding system configuration file—e.g:

```
# echo 'repository=https://alpha.us.repo.voidlinux.org/current' > \
> /etc/xbps.d/00-repository-main.conf
```

The corresponding name of the `non-free` repo system file is `10-repository-nonfree.conf`.

g) If you had to unblock wireless interfaces with `rfkill` previously, add the command to `/etc/rc.local` to do this automatically on system initialization, before services are executed:

```
# echo 'rfkill unblock wlan' >> /etc/rc.local
```

7. Set up swap

Setting up swap is required if you will use hibernation, and necessary on systems with low to moderate amounts of physical memory (and otherwise simply prudent).

If you have created a swap partition, start in step d below replacing "`/swap`" with your swap partition (prefer UUID for the `fstab` entry).

Or else, set up a swap file going through the following steps:

a) If not using btrfs, jump to step b. For btrfs filesystems there is some prior setup to be done:

i) First create a zero length file:

```
# truncate -s 0 /swap
```

ii) Then disable COW (copy-on-write) on it:

```
# chattr +C /swap
```

iii) And make sure compression is disabled:

```
# btrfs property set /swap compression none
```

b) Allocate the swap file, replacing `‹count›` with the number of MB you wish swap to have:

```
# dd if=/dev/zero of=/swap bs=1MiB count=‹count›
```

c) Set the appropriate permissions to it—it should only be accessible to priviliged processes:

```
# chmod 600 /swap
```

d) Set up a Linux swap area on the file:

```
# mkswap /swap
```

e) Finally, add an entry for it in `fstab`:

```
# echo '/swap none swap sw 0 0' >> /etc/fstab
```

8. Make the new installation bootable

You can boot the kernel directly as an EFI executable or use a separate bootloader.

To set up the bootloader GRUB, consult the Void documentation with the **void-docs** command:

```
# void-docs -r chroot
```

Note: Many UEFI implementations allow choosing boot entries on startup, which may make using a separate bootloader/manager for multi-booting superfluous.

To load the kernel directly as an EFI executable, use **efibootmgr**:

i) Go through the options in `/etc/default/efibootmgr-kernel-hook`, enabling modification of EFI entries so that a boot entry is automatically generated on kernel package reconfiguration.

- `OPTIONS` needs to supply the root partition, in the form ”`root=UUID=a1b2(...)`“. You can use `blkid` or `lslbk` to add the UUID string to the file. Common kernel parameters to add here are ”`resume=‹swap_partition›`“ if using hibernation and ”`quiet`“ if wishing to see fewer log messages during system startup.
- `DISK` + `PART` point to the ESP. Due to a potential issue with block device name persistence (where a device `sdX` may later on surface as `sdY`) set the `DISK` parameter dynamically using `findmnt`, where `‹ESP_dir›` below is your ESP mount directory (e.g. `/boot`):

  ```
  DISK="$(findmnt -enrs -o SOURCE -M ‹ESP_dir› | cut -c -8)"
  ```

  Note that for NVMe drives, the device name will have to be formatted differently.

ii) Add an entry for the system's efivars to `fstab`, and mount them manually so that the boot entry can be generated in the next step:

```
# echo 'efivarfs /sys/firmware/efi/efivars efivarfs defaults 0 0' >> \
> /etc/fstab
# mount efivarfs
```

9. Reconfigure and reboot

Use **xbps-reconfigure** to ensure all installed packages are set up properly:

```
# xbps-reconfigure -fa
```

This also generates a boot entry for the new installation. Review the generated boot configuration with the `efibootmgr` command, as you may need to update the boot order on first install.

At this point, the installation is complete. `exit` the chroot, unmount or `sync` the new filesystems manually to be sure no writes are pending, and reboot the computer with “`shutdown -r now`”.

System Components and Management

Network

Do any necessary configuration to connect to the internet and persist it across boots. If not using static IPs you will minimally need to set up a DHCP client, and a wireless daemon if using Wi-Fi:

For DHCP with **dhcpcd** or **ndhc**, you can simply copy and rename the service folder suffixing your network interface’s name, edit the `run` file to use such interface, and then enable the service.

For wireless connections, enable the (potentially interface specific) `wpa_supplicant` or `iwd` services and use **wpa_cli** or **iwctl** respectively to easily set up a new connection. Note that `iwd` can also be used as a DHCP client, and that it requires `dbus` to be running.

For Domain Name Resolution (DNS), you may use a separate application that provides security features and/or caching, like `stubby` or `unbound`, or use the nameservers already set up by your DHCP client, or simply add your preferred nameservers manually to `/etc/resolv.conf`.
Note that also, `firefox` has built-in resolution capabilities using DNS-over-HTTPS and name caching.

Firewall

You can manage the kernel firewall with **iptables**. For quick setup for IPv4 connections, apply the `simple_firewall.rules` present in `/etc/iptables/` with the `iptables-restore ‹rules›` command. See the Void docs for more details:

```
# void-docs -r firewall
```

Tip: If using `iptables`, you should follow the “Simple stateful firewall” guide in the ArchWiki to generate a more elaborate rule set, and which has a description of what each rule does.

Users

a) Privilege elevation: Once the system is fully set up, it is crucial to log in as a regular user for normal operation. To grant such regular users per-command privilege elevation you can use the tool **doas**, provided by the `opendoas` package. Simply install the package and set it up to allow users of the group *wheel* to execute commands as *root* when issued with it:

```
# echo 'permit :wheel as root' > /etc/doas.conf
```

A very common, yet more internally complex alternative is using `sudo` for this purpose.

b) User defaults: Default user creation parameters are set in `/etc/default/useradd`. Modify the file, and those in `/etc/skel/`, to provide the desired initial environment for new users.

c) Adding users: Add the aforementioned regular user (plus any others you may need) and then set their password. In addition to the *wheel* group, this user should be part of the *audio*, *input* and *video* groups to access audio and input devices and video hardware acceleration, among others:

```
# useradd -m -G audio,video,input,wheel ‹user_name›
# passwd ‹user_name›
```

Audio

To set up audio, install the `alsa-utils` package and use the `alsamixer` tool to unmute channels and set the volume. If you see that your desired sound card is not set as default, find the card's index number in `/proc/asound/cards` and set it as the default in `/etc/asound.conf`:

`/etc/asound.conf`

```
# Set default sound card
defaults.ctl.card ‹card_index›
defaults.pcm.card ‹card_index›
defaults.hwdep.card ‹card_index›
defaults.timer.card ‹card_index›
```

Notes:

- You may also, or instead, need to specify the default ALSA *device*, in the same way.
- If you use applications that require Pulseaudio, install it or the `apulse` package.
- See the "Input and ACPI events" section for details on adjusting volume with hotkeys.

Display server

This section focuses on installing and configuring the Xorg display server with a window manager. You can follow the installation below and simply add or replace any drivers and components you need—for information on installing proprietary or vendor specific graphics drivers, access Void's documentation with the **void-docs** command, e.g:

```
# void-docs -r nvidia
```

Installation

Select the packages needed for a graphical session, including the X server, startup scripts, input and graphics drivers and the message bus system:

```
# PACKS='xorg-minimal mesa-dri dbus-x11
```

… *add bitmap and outline fonts (potentially some with extensive unicode support):*

```
> font-misc-misc dejavu-fonts-ttf (font-unifont-bdf noto-fonts-ttf)
```

… *and your preferred window manager, a tool to launch applications (which may be included with the former) and a terminal emulator—for example:*

```
> spectrwm dmenu st'
```

Install the selected packages, making sure the repository index is synchronized and that the system is up-to-date:

```
# xbps-install -Su $PACKS
```

Configuration

Keyboard layout: To set the default keyboard layout(s) for Xorg, add an X configuration file that defines an input class setting the "XkbLayout" option to them:

```
/etc/X11/xorg.conf.d/91-keyboard-user.conf
```

```
# Set keyboard layout
Section "InputClass"
     Identifier           "default keyboard layout"
     MatchDriver          "libinput"
     MatchIsKeyboard      "on"
     # Either set one layout:
     Option "XkbLayout"   "‹layout›"
     # Or define more layouts to switch between, e.g.
     # by pressing both Ctrl keys simultaneously:
     #Option "XkbLayout"  "‹layout›,‹layout_alt›"
     #Option "XkbOptions" "grp:ctrls_toggle"
EndSection
```

To temporarily set other keyboard layouts on Xorg use the **setxkbmap** command, provided by the homonymous package.

Touchpad tapping: To enable tapping on touchpad devices, add an X configuration file that defines an input class setting the "Tapping" option to them:

```
/etc/X11/xorg.conf.d/91-touchpad-user.conf
```

```
# Enable tapping
Section "InputClass"
     Identifier           "enable touchpad tapping"
     MatchDriver          "libinput"
     MatchIsTouchpad      "on"
     Option "Tapping"     "on"
EndSection
```

Graphics driver: To specify the driver to use with a graphics device and its configuration, first find the BusID of the device with **lspci** (the first column) and format it accordingly—usually just prefixing it with "`PCI:`" and replacing the dot with a colon—then use it in an X configuration file specifying the desired driver along with any options, in the following format:

```
/etc/X11/xorg.conf.d/91-integrated-graphics.conf
```

```
# DDX Intel Graphics driver configuration (from xf86-video-intel package)
Section "Device"
     # Some identifier
     Identifier "integrated graphics device"
     # Driver to use with the device
     Driver     "intel"
     # BusID of the device in our PC
     BusID      "PCI:00:02:0"
     # Driver settings (optional)
     # This one is driver specific, to prevent tearing without compositing
     Option     "TearFree" "on"
EndSection
```

Session timeouts: To set the screen locker and DPMS timeouts, set the BlankTime and OffTime options respectively in a server flags X configuration file:

/etc/X11/xorg.conf.d/91-server-flags.conf

```
# Xorg server configuration
Section "ServerFlags"
     Option "BlankTime"   "‹minutes_blank›"
     Option "StandbyTime" "0"
     Option "SuspendTime" "0"
     Option "OffTime"     "‹minutes_off›"
EndSection
```

Notes:

- The distinction in Standby, Suspend and Off times is only meaningful for CRT monitors. Furthermore, screen blanking often triggers DPMS activation in modern monitors.
- Use the tool **xss-lock** to run a screen locker (like *slock*) triggered by the Blank timeout.

Starting the X server

X is commonly started using the **startx** script, provided by the `xinit` package, which handles the more complex aspects of setting up the server. To configure it, modify the `xinitrc` script in `/etc/X11/xinit/` to launch your desired application (like a window manager) when X starts, after doing any necessary setup - for example:

/etc/X11/xinit/xinitrc

```
#!/bin/sh
# start some nice programs - some applications put scripts there
if [ -d /etc/X11/xinit/xinitrc.d ] ; then
 for f in /etc/X11/xinit/xinitrc.d/?*.sh ; do
  [ -x "$f" ] && . "$f"
 done
 unset f
fi
# apply per-user wallpaper - this uses ImageMagick
display -background gray12 -backdrop -window root $HOME/Pictures/.wallpaper.*g
# start locker daemon
xss-lock slock &
# start session
dbus-launch --exit-with-x11 spectrwm
```

The last line above sets up a session message bus and launches the window manager `spectrwm`. When the window manager quits, X will in turn exit which will cause the dbus daemon to notify clients of this (which ideally would clean up and quit in response) and then exit itself.

To start X automatically when logging in from a login shell, execute `startx` in `/etc/profile`, which is read and executed by the login shell at such point, conditioning such execution on any parameters you require:

/etc/profile

```
(...)
# startx automatically when logging in on tty1 as a regular user
if [ ! $DISPLAY ] && [ "$(tty)" = '/dev/tty1' ] && [ $(id -u) -ne 0 ]; then
  exec startx
fi
```

Once you have tested your configuration to work, consider modifying the `Xwrapper.config` file in `/etc/X11/` to allow for auto-detection of required privileges to run the server, so that X itself may be run as a regular user rather than root (see Xorg.wrap(1)). This is a security improvement, but its availability depends on the graphics devices and drivers you use.

Note: The "exec" command above is desirable, as when the X server exits, the user will be logged out. This denies the ability of killing (or crashing) the X server to bypass locking or authentication mechanisms, and it does not prevent running Xorg root-less.

Input and ACPI events

To manage input and ACPI events you use **acpid**, a daemon which links such events to an action. It can be used to set volume and backlight levels on key presses, to react to power or battery related events, and much more. Generically, to trigger an action in response to an event:

i) Use the tool **acpi_listen** to identify the event by triggering it.

ii) Add an entry for the event to a file in `/etc/acpi/events/`, and its associated handler script (if such is required) in `/etc/acpi/handlers/`.

iii) Restart the `acpid` service.

The script `/etc/acpi/handler.sh` has preset actions for some ACPI events. Since its dynamic power management functionality will have to be implemented differently for one’s machine, and to maximize the modularity of the configuration, it is advisable to limit such script to handling system power states and then manage others with dedicated files:

/etc/acpi/events/anything

```
# anything, that is one of these
event=button/(power|sleep|lid)
action=/etc/acpi/handler.sh %e
```

Reboot, shutdown and suspend

To reboot, simply log out and hit `Ctrl+Alt+Del` on the login shell, which is set to send the appropriate signal to the OS.

To shutdown or suspend the system, use `acpid` and **zzz**. Simply enable the `acpid` service if you have not yet done so, and the following will already be configured for you:

- Shutting down by pressing (but not holding) the power button on your PC. Make sure you have logged out of all sessions first, being dropped out to a login shell.
- Suspending by pressing the sleep button on your PC, or by closing a laptop’s lid.

Hibernation

After setting up an environment that allows for hibernation (with an appropriately sized swap partition and the corresponding kernel parameter), you can trigger it by issuing the "`zzz -Z`" command as a response to an event of your choosing through `acpid`.

Volume and backlight

Files to manage volume and backlight related input events through `acpid` are provided in the Appendix.

Power and battery

Battery: Some laptops send the appropriate signals the OS as a battery discharges, which you can handle with `acpid`. For others, you can query battery status and load with the **acpi** command, provided by the homonymous package, whenever another event occurs or on a timer (see next section).

Power: See the "Power profiles" subsection of the "Power saving and performance" section further below, and the relevant files in the Appendix.

Scheduled actions

To schedule the execution of commands you can use **snooze**, which has services already set up to run commands on an hourly, daily, weekly and monthly basis that you can enable. You can also use `snooze` to run a command at a particular time and date, or on different periods.

The simplest way to use it is to enable a service for a particular run frequency, and drop executable scripts in its corresponding folder with the name "`chron.<frequency>`" in `/etc/` (which may have to be created).

Among other things, scheduling can be used to automate system maintenance, updating backups, querying for information over the web, send notifications, etc. Common periodic actions most users will require are described below.

Notes:

- *You should minimally enable the `snooze-daily` service, as daily scheduling is used for some system maintenance tasks.*
- *Consider installing the package `PopCorn`, which provides a service that uses `snooze` as well and supplies the Void project with anonymous package statistics.*

Network Time Synchronization

There are many ways to synchronize the system's clock with that of a time server. A simple and effective one is using the SNTP Perl script from the site kloth.net:

i) Install `perl` if you have not already. Apart from being handy, Perl is a dependency of a large amount of packages so it would likely be pulled as such at some point.

ii) Copy the script on the website to a file, commenting out the invocation of the "output_ntp_data" function (as suggested) near the bottom, to make it less verbose.

iii) Set is as executable, and drop it in `/usr/local/bin/`.

iv) Add an entry in `chron.weekly` so that `snooze` may trigger its execution regularly:

```
/etc/chron.weekly/sntp-update
```

```
#!/bin/sh
/usr/bin/perl /usr/local/bin/sntp -u pool.ntp.org
```

Notes:

- By running the script without the "-u" option on a terminal, you can see the network time and its delta compared to your system's time. Small deltas, <1s, are expected, but even deltas of several seconds are usually not a problem.
- If accuracy is critical, or if your hardware clock tends to deviate substantially from the actual time, consider using an NTP daemon instead.

Filesystem TRIM

Filesystems on SSD drives should be subject to *trimming* periodically to maintain drive performance and health. For this, use the tool `fstrim` provided by the `util-linux` package, itself a dependency of the `base-minimal` meta-package.
To trim all supported filesystems listed in `fstab` weekly, use the following:

```
/etc/chron.weekly/fstrim-fstab
```

```
#!/bin/sh
/usr/bin/fstrim -A
```

Tip: That above is periodic trim. Do not use continuous *trim -the "discard" mount option or flag- unless you have a solid understanding of its purpose and how it would affect your drive.*

Power saving and performance

Configuration in this section is presented to be applied *statically,* not dependent on the system's power source or a global 'power mode', while the "Power profiles" subsection describes how it can be applied dynamically and files for such purpose are provided in the Appendix. Each item is succinctly described—the user is expected to read further about them elsewhere if required.

An excellent aid in the configuration of power saving is the tool **powertop**, which can generate system power reports and show real time power draw data. While being primarily a diagnostics tool, it also has an "auto-tune" option which can automatically set a good deal of the tunables below—however, note that such automatic configuration is not persistent and not meant to replace a custom-tailored one.

Indeed widely, I suggest avoiding tools that do intend to provide automatic power configuration. Understanding and tailoring the configuration for one's hardware and its use will result in a more reliable, efficient and better performing system—and it can be quite simple, as today most power management aspects are already handled smartly by your firmware, drivers, kernel and distro.

Laptop mode

If you are using a laptop and running the OS from a hard drive (i.e, not an SSD, etc), you can use the "laptop mode" kernel feature as a power saving technique, which aims to keep the OS's hard drive in suspension as much as possible. An implementation of this feature can be found in the Appendix, which requires that you disable *access timestamps* as described below, and that you do not enable *block layer rpm* on your main hard drive, as it uses `hdparm` for drive suspension.

Lockup detectors

Turning off the kernel's lockup detectors can decrease power consumption, as these are active high-priority tasks and, particularly, the hardlockup detector may generate a high number of interrupts on some systems. These detectors are used for debugging, and are generally dispensable to PC users. To do this, add the "nowatchdog" option to your kernel parameters in your `efibootmgr` hook or bootloader configuration.

PCI and SCSI bus devices

PCI RPM: PCI devices can be put in a low-power state when idle through automatic Runtime Power Management to save power. The caveat is that some devices may not become active again after entering such state. To handle this, identify the Vendor and Device IDs of any device that fails to wake up with "`lspci -nn`" and add them to a blacklist, so that those remain active throughout.

Block Layer RPM: Similarly, block devices may use runtime power management to access low-power states. This time a timeout needs to be set, at which point the device will flush the cache and go into suspension. Note that using it with frequently accessed hard drives could result in high latencies and component wear-down if the power state transitions are over-frequent. A whitelist may thus be preferred, where drives can be identified by their World Wide Identifier (see `/sys/bus/scsi/devices/*:*:*:*/wwid`) to be selected and assigned specific timeouts.

SATA LPM: Further power savings can be achieved by using a medium power SATA Link Power Management setting, allowing devices to enter lower power states. The recommended option for modern systems is “med_power_with_dipm”, which works together with the devices’ own power configuration, otherwise being regular “medium_power”. The default is “max_performance”.
Note: While reportedly rare, using medium LPM settings (with some older drives) could result in data corruption. There is a “min_power” mode as well, but using it is strongly discouraged due to the risk of data loss, and using Device Initiated PM already results in similar power savings.

Readahead: Increasing file data readahead for hard drive devices can substantially increase I/O throughput with minimal latency penalties, and it can also benefit solid state drives to a more moderate degree. The default (software) readahead is 128kB, which can be safely doubled on modern systems, while more memory or I/O constrained systems should keep the default value.

Scheduler: I/O performance can be increased by using the appropriate I/O scheduler for your hardware and workloads. Generally, you can improve performance and reduce latency by using the "mq-deadline" scheduler on SSDs and the (default low latency) "bfq" scheduler on HDDs. However, NVMe drive performance benefits from setting no scheduler at all, i.e. “none”.

To apply these settings use a set of **udev** rules applying desired values, for example:

`/etc/udev/rules.d/91-power-and-performance.rules`

```
# schedulers
SUBSYSTEM=="block", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0", \
     ATTR{queue/scheduler}="mq-deadline"
SUBSYSTEM=="block", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1", \
     ATTR{queue/scheduler}="bfq"
# readahead
SUBSYSTEM=="block", KERNEL=="sd[a-z]", ATTR{queue/read_ahead_kb}="256"
# SATA LPM
SUBSYSTEM=="scsi_host", KERNEL=="host*", \
     ATTR{link_power_management_policy}="med_power_with_dipm"
# Block Layer RPM - using whitelist
SUBSYSTEM=="scsi", ATTR{wwid}=="exact very long world wide identifier", \
     ATTR{power/autosuspend_delay_ms}="300000", ATTR{power/control}="auto"
# PCI RPM - using blacklist
SUBSYSTEM=="pci", ATTR{vendor}=="0x0a1b", ATTR{device}=="0x2c3d", \
     GOTO="pci_rpm_end"
SUBSYSTEM=="pci", ATTR{power/control}="auto"
LABEL="pci_rpm_end"
```

HDD Advanced Power Management

Among other parameters, you can set a hard drive’s Advanced Power Management (APM) feature with the tool **hdparm**. Generally, it is best to set high APM levels to avoid excessive head parking and undesired spindown -which can cause heightened component wear-down when too frequent- while also not compromising the -much needed- drive’s performance. Power savings and wear reduction can be achieved by manually managing drive suspension, which may be done in different ways—e.g. the above-mentioned *Block Layer RPM* or tools like `hdparm`.

Note that many drives *do* allow spindown with APM levels above 127, while they may ignore any manually set timeout with levels below 128. If unsure, try first levels 192, 128, 127, in that order, and add `hdparm` commands to `/etc/rc.local` to apply them on system startup and to a script in `/etc/zzz.d/resume/` to retain them after a system suspend.

Filesystems and virtual memory

Access timestamps: Using the `noatime` mount option disables inode access time updates on a filesystem, increasing performance and reducing drive access. You can set this in `/etc/fstab` to your root and ESP partitions, and any other mount point you judge it pertinent.

Data flushing: To further reduce drive access you can increase the intervals after which the kernel will write old data to disk. This is by default done every 5 seconds for both data and journaling (on journaling filesystems like ext4) — you can tune it with the following:

i) Journaling: Add the “`commit=⟨seconds⟩`” option to your (ext4 or btrfs) filesystem's mount options in `/etc/fstab`. E.g, 15 seconds.
ii) Data: Add the “`vm.dirty_writeback_centisecs = ⟨centiseconds⟩`” setting in a `.conf` file located in `/etc/sysctl.d/` (see further below). E.g, 1500 centiseconds.

```
/etc/fstab
```

```
# <file system>  <dir>    <type>    <options>                        <dump> <pass>
# /dev/sda2
UUID=a1b2(...)   /        ext4      defaults,noatime,commit=15          0     1
# /dev/sda1
UUID=9Z8Y(...)   /boot    vfat      defaults,noatime,umask=022,utf8     0     2
(...)
```

Paging and cached data: There are two main tuning knobs to consider here, *swappiness* and *virtual filesystem cache pressure*. The first one defines the rough relative I/O cost of swapping and filesystem paging, the other controls the tendency of the kernel to reclaim the memory which is used for caching of directory and inode objects.
The default value of swappiness is 60—decreasing the value will cause the kernel to disprefer swapping. The default value of cache pressure is 100—decreasing the value will increase its preference to conserve cached data.
On systems with high physical memory, you can use lower swappiness and cache pressure values, e.g. 15 and 65 respectively, by setting “`vm.swappiness`” and “`vm.vfs_cache_pressure`” in a `.conf` file located in `/etc/sysctl.d/`:

```
/etc/sysctl.d/91-df-vm.conf
```

```
vm.dirty_writeback_centisecs = 1500
vm.swappiness = 15
vm.vfs_cache_pressure = 65
```

Audio power save

For audio devices using the "snd_hda_intel" or "snd_ac97_codec" drivers, these can be set to turn off the codec power when the device has not been in use after a certain amount of time.

Check the drivers your audio devices are using with “`lspci -k`”, and set the “`power_save`” option with a timeout in seconds for such drivers in a `.conf` file located in `/etc/modprobe.d`:

`/etc/modprobe.d/91-audio-power-save.conf`

```
#options snd_ac97_codec power_save=5
options snd_hda_intel power_save=5
```

CPU frequency scaling

CPU frequencies are managed by the active *governor* for each core. Scaling governors set frequencies dynamically, providing great performance while not sacrificing efficiency. Situationally, however, non-scaling governorns may be preferred, like (`cpufreq`'s) ‘powersave’ governor which sets frequencies statically to their lowest point for maximum power savings.
Within `cpufreq`, the “schedutil” scaling governor is gradually turning to be the default, “best-of-all-worlds” one for general use, being closely integrated with the kernel. To set it -or any other governor you wish- at startup, add the following to your `rc.local` file:

`/etc/rc.local`

```
(...)
for d in /sys/devices/system/cpu/cpufreq/policy[0-9]* ; do
     echo 'schedutil' > "$d/scaling_governor"
done
(...)
```

Note that, on modern Intel CPUs, you can use the Intel P-State driver which provides its own scaling governors when active, called ‘performance’ and ‘powersave’ (like cpufreq's *non*-scaling governors). Broadly, they still tend to perform better than “schedutil”, especially when raw performance is prioritized over power draw. If you wish to use them, set the governors’ `status` in `/sys/devices/system/cpu/intel_pstate/` to “`active`” and then set a governor as above.

Initial images

You can generate smaller initramfs boot images by directing **dracut** to install only what is needed for booting the local host, which can speed up startup times. This is usually safe, but you should always test it first conserving a generic boot image in case some module results to be missing. Additionally, you can use a different compressor or no compression at all for further tuning. E.g:

`/etc/dracut.conf.d/90-boot-image-tuning.conf`

```
# generate host-specific boot image
hostonly="yes"
# we set kernel parameters elsewhere
hostonly_cmdline="no"
# having installed lz4, we can use it as a fast (de)compression alternative
#compress="lz4"
```

Power profiles

Some power and performance configuration needs to be dynamic: Laptop users will require that the system enter a power saving mode when running on battery, and desktop users may wish to trigger a (high power) minimal latency mode for timing-critical tasks, among other scenarios. Such cases can be easily managed with `acpid`, by creating an event file that matches the event, like a power source transition or some sort of *turbo button* press, to a script that will set the desired configuration.

When changing the configuration dynamically, kernel parameters should be set with the **sysctl** command, and parameters set above through `udev` are (typically) accessible as writable files under the `/sys` folder.

Laptop users should set up power profiles dependent on the computer's power source, usually involving HDD power management if running the OS from one, CPU frequency scaling, and any relevant GPU settings that their driver provides. All but the last one are tackled in the Appendix files, and the latter can easily be added to them. Do not forget you can check `powertop`'s live data while setting it up, to identify any further configuration that may be needed.

What about...?

Wakeup-on-Lan?

WOL appears to be disabled by default on Void. If you wish to set it, use the commands that `powertop` provides to toggle it, or use a `udev` rule to -for example- match to the "net" subsystem and change the `wakeup` power setting on the devices (enabled/disabled).

USB autosuspend?

I find that its unreliability outweighs the meager benefit. Furthermore, a power-sensitive system like a laptop would be set to suspend before USB autosuspend could make an appreciable difference, as any connected USB devices will likely *be used* when the computer is active.

AHCI host controller PM?

Quite bug-prone on *both* my laptop and desktop, so I omitted it out of caution. Perhaps in a future version of this guide, if I can solve the problems, it will be featured. If you want to try it, use the commands that `powertop` provides and test, checking `dmesg` logs, after having saved and synced your data.

Active State Power Management?

This is normally handled by your BIOS.

APPENDIX

Files in the Appendix should simply be copied to their corresponding system directories. Make sure that all files are readable, and additionally that all scripts are executable.

The scripts are made for dash (Debian Almquist shell), so they should be quite compatible.

WARNING: The scripts in the Power Management section below need testing. They should be safe to try, but take a look at them prior to use.

POWER MANAGEMENT

/etc/acpi/events/ac_adapter

```
# Change power mode on power source transition
event=ac_adapter
action=/etc/acpi/handlers/power-ctl %e
```

/etc/acpi/handlers/power-ctl

```
#!/bin/sh
# Power mode handler script - part of: rinsmiles guide to the Void.
# Use HDD APM and spindown features, and allow laptop-mode.
manage_hdd=1
# Hdparm's APM and spindown options.
hdparm_options_powersave='-qB127 -qS40'
hdparm_options_performance='-qB192 -qS0'
# Use laptop mode for further power savings. Overrides the above.
use_laptop_mode=1
# Force using specified root HDD. Values: {no|/dev/disk/by-id/<hdd-id>}
force_hdd='no'
# Allow setting CPU governors and driver state.
manage_cpu=1
# Set CPU frequency governors.
governor_powersave='powersave'
governor_performance='ondemand'
# Use Intel P-State driver for scaling if available. Overrides the above.
use_pstate_scaling=1
#
# Setup: HDD
if    [ "$force_hdd" != 'no' ] ; then
      rd_id="$force_hdd"
else # get the device containing the root mount, as long as it is compatible
      rd_id=$(findmnt -enr -o SOURCE -t ext4,btrfs -M / | cut -c 6-8)
      rd_id_pass=$(cat /sys/class/block/$rd_id/queue/rotational || echo 0)
      rd_id="/dev/$rd_id"
      [ "$rd_id_pass" != '1' ] && manage_hdd=0
fi
[ ! -x /etc/acpi/resources/laptop_mode ] && use_laptop_mode=0
#
```

Continues on the next page

Continued: /etc/acpi/handlers/power-ctl

```
#
# Setup: CPU
cpu_dir='/sys/devices/system/cpu'
if [ $use_pstate_scaling -eq 1 ] && [ -w $cpu_dir/intel_pstate/status ] ; then
     governor_powersave='powersave'
     governor_performance='performance'
else use_pstate_scaling=0
fi
#
set_hdd_mode() {
     if [ $use_laptop_mode -eq 1 ] ; then
          /etc/acpi/resources/laptop_mode $lmode "$rd_id"
     else hdparm $hdparam "$rd_id"
     fi
}
set_cpu_mode() {
     [ $use_pstate_scaling -eq 1 ] && \
          echo 'active' > $cpu_dir/intel_pstate/status
     for d in $cpu_dir/cpufreq/policy[0-9]* ; do
          echo $governor > "$d/scaling_governor"
     done
}
#
case "$2" in
     AC*|ADP0)
          case "$4" in
               00000000) # battery mode configuration
                    lmode='enable'
                    hdparam=$hdparm_options_powersave
                    governor=$governor_powersave
                    [ $manage_hdd -eq 1 ] && set_hdd_mode
                    [ $manage_cpu -eq 1 ] && set_cpu_mode
                    # Signal the user that backlight should be lowered:
                    [ -x /etc/acpi/handlers/backlight-ctl ] && \
                         /etc/acpi/handlers/backlight-ctl down
               ;;
               00000001) # ac_power mode configuration
                    lmode='disable'
                    hdparam=$hdparm_options_performance
                    governor=$governor_performance
                    [ $manage_hdd -eq 1 ] && set_hdd_mode
                    [ $manage_cpu -eq 1 ] && set_cpu_mode
               ;;
          esac
     ;;
esac
```

/etc/acpi/resources/laptop_mode

```
#!/bin/sh
# Laptop mode setup script - part of: rinsmiles guide to the Void.
# Usage: laptop_mode {enable|disable} <oshdd>
# Requires noatime mnt opt be set on oshdd's filesystems, and hdparm. See:
# https://www.kernel.org/doc/html/latest/admin-guide/laptops/laptop-mode.html
#
hdparm -qf "$2" || exit
case "$1" in
	enable)
		# enable laptop mode with its default values for the vm subsystem
		sysctl -qw vm.laptop_mode=5               \
			vm.dirty_writeback_centisecs=60000   \
			vm.dirty_expire_centisecs=60000      \
			vm.dirty_ratio=40                    \
			vm.dirty_background_ratio=5
		mount --all -o remount,commit=600 -t ext4,btrfs
		hdparm -qa2048 -qB127 -qS6 "$2"
	;;
	disable)
		# disable laptop mode, setting parameters to their default values
		# where commit, writeback and readahead's are suggested in the guide
		sysctl -qw vm.laptop_mode=0               \
			vm.dirty_writeback_centisecs=1500    \
			vm.dirty_expire_centisecs=3000       \
			vm.dirty_ratio=20                    \
			vm.dirty_background_ratio=10
		mount --all -o remount,commit=15 -t ext4,btrfs
		hdparm -qa512 -qB192 -qS0 "$2"
	;;
esac
```

VOLUME AND BACKLIGHT

/etc/acpi/events/video-brightnessup

```
# Increase screen brightness through dedicated script
event=video/brightnessup
action=/etc/acpi/handlers/backlight-ctl.sh up
```

/etc/acpi/events/video-brightnessdown

```
# Decrease screen brightness through dedicated script
event=video/brightnessdown
action=/etc/acpi/handlers/backlight-ctl.sh down
```

/etc/acpi/handlers/backlight-ctl.sh

```
#!/bin/sh
# Screen backlight handler script - part of: rinsmiles guide to the Void.
#
# Look for an interface
bl_dev='/sys/class/backlight/acpi_video0'
[ ! -d $bl_dev ] && bl_dev='/sys/class/backlight/intel_backlight'
# Get max and current values, and derive increment/decrement
bl=$(cat $bl_dev/max_brightness) || exit
bl_cur=$(cat $bl_dev/brightness)
bl_mod=$(($bl / 15))
#
case $1 in
     up)       bl=$(( $bl_cur + $bl_mod )) ;;
     down)     bl_clamp=$(($bl / 3))
               bl=$(( $bl_cur - $bl_mod ))
               [ $bl -lt $bl_clamp ] && bl=$bl_clamp ;;
     max)      ;;
     dim)      bl=$(( $bl / 2 )) ;;
esac
echo $bl > $bl_dev/brightness
```

/etc/acpi/events/button-volumeup

```
# Raise Master volume of default ALSA device
event=button/volumeup
action=/usr/bin/amixer set Master 5%%+
```

/etc/acpi/events/button-volumedown

```
# Lower Master volume of default ALSA device
event=button/volumedown
action=/usr/bin/amixer set Master 5%%-
```

/etc/acpi/events/button-volumemute

```
# Toggle Master volume of default ALSA device
event=button/mute
action=/usr/bin/amixer set Master toggle
```

Hope you've found it useful! Feedback is always welcome, find me on Reddit as:

rinsmiles