

# Post Installation

From Void Linux Wiki

This article provides a list of basic functions to perform after finishing the installation of Void.

(If you cannot login after doing an install from a GUI Live ISO, you may need to create an account from a virtual console first. `CTRL + Alt + F2` should get you to one where you will be able to create the account as documented below. After that you should see that user account listed on the login page.)

## Contents

- 1 Updates
- 2 Setting a keymap
  - 2.1 Xorg keymap (Optional)
- 3 Services
- 4 Network
  - 4.1 Wireless-only access
- 5 User Account
- 6 Graphical User Interface
  - 6.1 Install Xorg
  - 6.2 Install a Desktop Environment
    - 6.2.1 ConsoleKit and ck-launch-session
- 7 Display Manager
- 8 Adjust screen brightness
  - 8.1 Manually, when X11 is not present
  - 8.2 With a script, when X11 is not present
  - 8.3 When X11 is used
- 9 Audio
- 10 Setting up Polkit
- 11 Allow users to shutdown without a password
  - 11.1 via ConsoleKit/Polkit
  - 11.2 via sudo
- 12 Regional Repository (Suggested)
- 13 Troubleshooting
  - 13.1 My hardware isn't showing up!
- 14 What's next?

## Updates

After installation, the first thing you should do on your new void system is update your remote repository information and update your installed packages. This is especially important if you used the **Local** source during installation – it is highly likely that at least some of your packages will be out of date. To update all packages to the most recent version, run:

```
# xbps-install -Suv
```

**Repeat this command until there are no more updates.** After updating your system, you may want to reboot depending on the updates installed (such as kernel updates).

In the event that there is no network connection e.g. need to set up wireless, proceed to the network section below and return.

## Setting a keymap

**Note:** The following information may be irrelevant/unnecessary for you if your Void installation includes a DE with a suitable keyboard config editor. The MATE flavour of Void Linux comes with such an editor, whereby you can easily pick your keymap with a GUI based application instead of editing config files directly.

To see all available keymaps on your system, run:

```
$ find /usr/share/kbd/keymaps/ -type f
```

To configure the **console** keymap, modify `/etc/rc.conf` with your keymap code. Using the results from the above command, remove the `.map.gz` suffix and assign it like this (for example, using the "Español" keymap):

```
# sed -i -e "s|#\?KEYMAP=.*|KEYMAP=us|" /etc/rc.conf
```

which becomes active after the next reboot. If you just want to change the keymap ad hoc, the **loadkeys** utility can be used:

```
$ loadkeys us
```

## Xorg keymap (Optional)

If you install Xorg, then that has its own settings for keyboard layout.

While the default keyboard layout can be overridden by running **setxkbmap** in a user's `.xinitrc`, or during session startup (depending on DE/WM used), this can be globally set by adding configuration snippets to `/etc/X11/xorg.conf.d/` (thus should work regardless of whether one uses a login manager or starts X from a logged-in console).

```
# cat /etc/X11/xorg.conf.d/10-keyboard.conf
Section "InputClass"
    Identifier "any value works here"
    MatchIsKeyboard "on"      # Limit these settings to keyboards
    #
    Driver "libinput"
    # Keyboard layout - comma-separated list for multiple layouts
    Option "XkbLayout" "es"
    # Keyboard variant - comma-separated list matching XkbLayout line
    #
    Option "XkbVariant" "dvorak"
EndSection
```

**A note on the *Driver* option:** In recent Xorg versions, it seems that this is not necessary. If a driver *is* listed, then Xorg will use it if it exists, otherwise it will use whatever module is available for keyboards. The layout settings will still be applied. (Previously, incorrectly selecting the driver could leave you without keyboard input in Xorg).

Also note that if multiple keyboard layouts are to be switched between, the line

```
Option "XkbOptions" "grp:shift_toggle"
```

will allow you to switch between multiple layouts by pressing both `Left Shift` and `Right Shift` keys simultaneously.

Keyboard layouts are defined in `/usr/share/X11/xkb/symbols/` and multiple variants of a language/layout are defined in one file.

## Services

Now is a good time to customize the running services. Void uses Runit as an init and service supervisor. Since the settings were most likely copied from the live system, there are possibly a few services running that you do not really need. Check the contents of `/var/service` and consider if you need each service. If you do not know or are unsure about a particular service, better leave it in place. These are relatively safe bets:

- You probably won't need 6 TTYs, so you can safely delete some of the higher-numbered `agetty-ttyX` entries. Just make sure you leave at least `agetty-tty1`
- You only need `mdadm` if you are using software RAID

- If you do not plan to connect to this system via SSH from another computer, you can remove *sshd*
- If you are using a static network configuration you can (and should) remove *dhcpcd*

**Note:** You can always re-enable removed services later should you need them.

A selection of possible firewall services and associated packages to add is discussed at Firewall Configuration.

## Network

If your network provides configuration via DHCP, the default setup should already work fine. If not you may need to enable the *dhcpcd* service first:

```
# ln -s /etc/sv/dhcpcd /var/service/dhcpcd
```

If you need a static configuration, first get the necessary data:

- Network interface name. We'll be using "enp0s3" in this example. The name can be obtained from the output of `ip link` on your system.
- IP-Address, eg "192.168.1.2"
- Network mask, eg "255.255.255.0"
- Broadcast Address, eg "192.168.1.255"
- Gateway address, eg "192.168.1.1"

When you have the data, put these lines into `/etc/rc.local` and adapt them to your settings:

```
# ip link set dev enp0s3 up
# ip addr add 192.168.1.2/24 brd + dev enp0s3
# ip route add default via 192.168.1.1
```

Now, after you reboot, you should have a working network connection.

## Wireless-only access

If you only have access to wireless, you will have to connect through *wpa\_supplicant*.

First you need to find the wireless interface name. It is most likely *wlp2s0*, and that name will be used for the following examples. However you can check the interface name with:

```
$ ip addr
```

Next you need to find the ssid (the name) of the available wireless networks:

```
$ sudo iw dev wlp2s0 scan | grep -i ssid
```

After you have picked the network you want, we must generate a configuration for it and add that to `/etc/wpa_supplicant/wpa_supplicant.conf`.

To generate a new configuration you must run:

```
$ wpa_passphrase {ssid} {password} | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf
```

where `{ssid}` is replaced by the actual network name, and `{password}` is replaced by the real password.

Finally, to actually connect you must run:


```
$ sudo wpa_supplicant -B -D wext -i wlp2s0 -c /etc/wpa_supplicant/wpa_supplicant.conf  
$ sudo iw wlp2s0 link
```

You should now have a working wireless connection.

See also: *Network Configuration*

## User Account

The next step, if only the root account exists, is to create an additional user account, put it into all the appropriate groups and set a password for it.


First, create a user using the `useradd` command. The below example creates a user named *voiduser*, sets the user's default shell to  bash ([https://en.wikipedia.org/wiki/Bash\\_\(Unix\\_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell))), and adds the user to the *wheel*, *users*, *audio*, *video*, *cdrom*, and *input* groups:

```
# useradd -m -s /bin/bash -U -G wheel,users,audio,video,cdrom,input voiduser
```

**Note:** The 'wheel' user group allows the user to escalate to root if they have the root password. It is also generally used to allow `sudo` privileges (see below).

Next, set the user's password using `passwd`:

```
# passwd voiduser
```

**Note:** Use your own shell instead of bash if you want, but you may need to install it first since only bash and  dash ([https://en.wikipedia.org/wiki/Almquist\\_shell](https://en.wikipedia.org/wiki/Almquist_shell)) are installed by default.

To allow this user (and other administrators) system access via the `sudo` command, you need to configure that behaviour. Edit `/etc/sudoers` by running

```
# visudo
```

and uncomment this line:

```
# %wheel ALL=(ALL) ALL
```

## Graphical User Interface

At this point you have a fully functional Void Linux environment and a regular user account to use it. Unless you installed Void with a DE already on top (in which case you can skip this and probably also the next section altogether), the next step is generally to setup a GUI. Void uses Xorg to provide a graphical environment. Please see the Xorg page for more information on installing and configuring Xorg.

The following steps will help you get a basic XFCE setup running. You can choose any desktop environment or window manager you want; the steps are essentially the same for each one. The following commands assume you are logged in as root.

### Install Xorg

The **xorg** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/xorg>) package is a metapackage that installs pretty much everything closely related to Xorg. This will give you a quick start but will also clobber your system with many unneeded packages. `xorg-minimal` (<https://github.com/voidlinux/void-packages/blob/master/srcpkgs/xorg-minimal/template>) is another metapackage with fewer dependencies, e.g. without any video drivers.

Quick start with *Xorg*:

```
# xbps-install -S xorg xterm
```

Or, you can install *xorg-minimal* instead if you own e.g. an Intel-based notebook.

This will suffice to continue with installation of a desktop environment:

```
# xbps-install -S xorg-minimal xorg-fonts xf86-video-intel
```

## Install a Desktop Environment

**Note:** The heavyweight desktop environments like Gnome and KDE increasingly rely on systemd, so they are likely outdated.

In this example, we will install xfce4. Other options include LXQT, Mate or even just a window manager like bspwm. See desktop environments.

As root (or sudo):

```
# xbps-install -S xfce4
```

It is recommended you only run a DE/WM as a regular user, and not as root. Many DEs will warn you of the danger, but will also allow you to do so at your own risk, if you so insist.

To configure a global xkbmap, see the above section Xorg keymap.

If you want to override the default X keymap on a per-user basis, create and edit the file `.xinitrc` in the user's home directory, and put these lines at the end:

```
setxkbmap de /* Adapt for your own keyboard layout – this selects the German QWERTZ layout */
exec startxfce4 /* Adapt for your chosen DE/WM */
```

Lastly, run `startx` to start an Xfce4 session. If you did not override the global X keymap, then you may simply run `startxfce4`. Some DEs will work this way, some will not (for instance, LXQT should be launched via your `.xinitrc` if not launching from a GUI login manager).

## ConsoleKit and ck-launch-session

If you want to use dbus and ConsoleKit2, in `.xinitrc` replace

```
exec i3
```

with

```
exec ck-launch-session dbus-launch --sh-syntax --exit-with-x11 i3
```

if you do not do that, programs utilizing **dbus** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/dbus>) and **ConsoleKit2** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/ConsoleKit2>) will not work, like the *Browse Network* feature of **Thunar** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/Thunar>)

## Display Manager

**Note:** Single user autologin without a display manager is described in the *Automatic Login to Graphical Environment* article.

Now that you have such a pretty GUI, you might find it tedious or ugly to login on the console and run `startx` every time. In that case, you can install a display manager that takes care of that much more fashionably. In this example we will use LightDM. For this to work you also need to enable the *dbus* service:

```
# xbps-install -S lightdm lightdm-gtk3-greeter
# ln -s /etc/sv/dbus /var/service/dbus
# ln -s /etc/sv/lightdm /var/service/lightdm
```

Below is a sample `.xinitrc` file you can tweak and put in your home directory (`~`) to help configure **xorg** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/xorg>), automatically start a desktop environment, etc.:

```
#!/bin/sh

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/etc/X11/xinit/.Xresources
sysmodmap=/etc/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdp -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f "$userresources" ]; then
    xrdp -merge "$userresources"
fi

if [ -f "$usermodmap" ]; then
    xmodmap "$usermodmap"
fi
```



```
[ -f ~/.Xdefaults ] && xrdp -merge ~/.Xdefaults

# start some nice programs

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

if [ "$(command -v startxfce4)" >/dev/null 2>&1 ];
then
exec startxfce4 --with-ck-launch
fi

if [ "$(command -v startkde)" >/dev/null 2>&1 ];
then
exec startkde
fi

if [ "$(command -v i3)" >/dev/null 2>&1 ];
then
exec i3;
elif [ "$(command -v bspwm)" >/dev/null 2>&1 ];
then
sxhkd & exec bspwm
else
printf "Install a DE/WM\n"
fi

if [ "$(command -v xset)" >/dev/null 2>&1 ];
then
#xset s off          #Disable screen saver blanking
#xset s 3600 3600    #Change blank time to 1 hour
#xset -dpms          #Turn off DPMS
xset s off -dpms     #Disable DPMS and prevent screen from blanking
#xset dpms force off  #Turn off screen immediately
#xset dpms force standby #Standby screen
#xset dpms force suspend #Suspend screen
fi
```

**Note:** In case you have problems logging in you may need to reboot, forcibly if necessary.

## Adjust screen brightness

If you just installed Void on a desktop computer, chances are that screen brightness is too high. This is especially noticeable on web browsers, where white backgrounds look too bright and mild greys are virtually white too.

### Manually, when X11 is not present

To change the brightness of the screen manually, edit `/sys/class/backlight/*/brightness`, for example, using `vi(1)`

(<https://man.voidlinux.org/vi.1>):

```
$ sudo vi /sys/class/backlight/*/brightness
```

The content of `/sys/class/backlight/*/brightness` should be an integer  $\geq 0$   $\leq 255$ .

## With a script, when X11 is not present

If a POSIX-compliant shell is being used, save the following script, for instance, as `~/brightness.sh`:

```
#!/bin/bash
brightness_file='/sys/class/backlight/*/brightness'

# check if one or more arguments were given
if [ $# -eq 0 ] # no arguments given
then
    echo Set brightness to:
    read input
    brightness=$input
else
    brightness=$1 # sets brightness variable as the first argument given
fi

case $brightness in
    *[!0-9]*)
        echo $brightness is not a positive integer.
        ;;
    *)
        sudo truncate -s 0 $brightness_file
        echo "$brightness" | sudo tee -a $brightness_file
        ;;
esac
```

Or if `ion` shell is being used, save the following script instead:

```
#!/usr/bin/env ion

let brightness_file = "/sys/class/backlight/*/brightness"
let brightness = 0

# ask for input if no arguments given
if test $len(@args) -eq 1
then
    echo "Set brightness to:"
    read input
    let brightness = $input
# otherwise use first argument
else
    let brightness = @args[1]
end

if matches $brightness '^0*([1-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$'
then
    sudo truncate -s 0 $brightness_file
    echo "$brightness" | sudo tee -a $brightness_file
fi
```

```
else
    echo "Input '$brightness' is not a positive integer >= 1 <= 255."
end
```

Run it in CLI mode using:

```
$ sh ~/brightness.sh
```

Or by passing an integer argument to it. For example, to change the brightness to 10:

```
$ sh ~/brightness.sh 10
```

## When X11 is used

If you use an X11-based window manager or desktop environment, you can address this matter with the `xrandr(1)` (<https://man.voidlinux.org/xrandr.1>) command. First off, run it without arguments to check and identify the available displays:

```
$ xrandr
```

On a dual monitor setup this might return something like

```
Screen 0: minimum 320 x 200, current 1920 x 1080, maximum 16384 x 16384
DisplayPort-0 disconnected (normal left inverted right x axis y axis)
HDMI-0 connected primary 1920x1080+0+0 (normal left inverted right x axis y axis) 698mm x 392mm
  1920x1080    50.00 +  60.00*   59.94   24.00   23.98
  1920x1080i    60.00   50.00   59.94
  1280x720     60.00   50.00   59.94
  720x576      50.00
  720x576i     50.00
  720x480      60.00   59.94
  720x480i     60.00   59.94
  640x480      60.00   59.94
DVI-0 disconnected (normal left inverted right x axis y axis)
DVI-1 connected (normal left inverted right x axis y axis)
  1680x1050    59.95 +
  1600x1200    60.00
  1280x1024    75.02   60.02
  1440x900     59.89
  1280x960     60.00
  1152x864     75.00
  1024x768     75.03   70.07   60.00
  832x624      74.55
  800x600      72.19   75.00   60.32   56.25
  640x480      75.00   72.81   66.67   59.94
  720x400      70.08
```

Following this example and assuming we want to reduce brightness on our HDMI

monitor to 80% of the nominal (max) value, we might do this:

```
$ xrandr --output HDMI-0 --brightness 0.8
```

This is a temporary change and will be lost upon the next reboot. To have the change automatically applied on startup, you should add the command as a startup application. The procedure varies depending on the DE you are using:

<b>Cinnamon</b>	Start menu → System Settings → Preferences → Startup Applications
<b>LXDE</b>	Applications and Settings → Preferences → Default applications for LXSession → Autostart (left sidebar)
<b>MATE</b>	System → Control Center → Startup Applications
<b>XFCE</b>	Start menu → Session and Startup → Application Autostart

If you are not using a desktop environment, the command should be specified in `$HOME/.xinitrc`.

## Audio

ALSA channels have to be unmuted manually. First install ***alsa-utils*** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/alsa-utils>):

```
# xbps-install -S alsa-utils
```

Unmute the desired channels with alsamixer:

```
$ alsamixer
```

Use the arrow keys to select a channel and the `m` key to unmute it. Muted channels are indicated by a `MM` label.

## Setting up Polkit

```
# xbps-install polkit polkit-gnome(optional)
# ln -sv /etc/sv/polkitd/ /var/service
# ln -sv /etc/sv/dbus/ /var/service (skip if already done)
```

## Allow users to shutdown without a password

## via ConsoleKit/Polkit

If you are using a Desktop Environment/Session Manager that makes use of ConsoleKit -such as XFCE, Cinnamon, or LXQT- then you have the ability to shutdown by using those environments' built-in logout/shutdown interfaces.

The default Polkit rules for ConsoleKit actions **already allow any user to perform shutdown/suspend** actions without any extra privileges on single-user systems (when other users are not logged in). On systems with multiple users logged in, the default is that only root can do this (and thus end all users' login sessions). These defaults can be overridden by rules added to </etc/polkit-1/rules.d/>. See Polkit documentation for rules syntax at FDO (<https://www.freedesktop.org/software/polkit/docs/latest/polkit.8.html>) or check `polkit(8)` (<https://man.voidlinux.org/polkit.8>).

**If using a login manager** such as `lightdm` or `lxdm`, the login manager will usually launch your chosen DE with ConsoleKit support, and things should "just work" out of the box.

However, **if starting your DE from a console login without a DM**, you may need special setup or command options for your DE's session to connect to ConsoleKit. *This will vary between DEs.*

For instance, XFCE4 should be launched with the command `startxfce4 --with-ck-launch` or by having `XFCE4_SESSION_WITH_CK` in your environment when you run **startxfce4**. LXQT can be launched by adding `exec ck-launch-session startlxqt` to the end of your `.xinitrc` and running **startx**.

**Note:** This functionality requires that the `dbus` service be enabled and running (either as a system service, or launched by a `ConsoleKit2` session).

## via sudo

This is the simplest method to give any group of users the ability to shutdown/restart without entering a password and can be useful for simpler WMs that don't talk to ConsoleKit2, but instead allow users to customize menu actions, such as OpenBox. This requires commands be prepended with **sudo**.

**Warning:** Never edit `/etc/sudoers` directly! Always use the **visudo** command, which will run vital sanity checks before exiting.

To allow members of the group "wheel" to shutdown without a password, use this line:

```
%wheel ALL=(ALL) NOPASSWD: /usr/bin/halt, /usr/bin/poweroff, /usr/bin/reboot, \  
/usr/bin/shutdown, /usr/bin/zzz, /usr/bin/ZZZ
```

**Note:** See the `sudo(8)` (<https://man.voidlinux.org/sudo.8>) documentation for more information on the `/etc/sudoers` file.

**Note:** It is also possible, if you *really* wanted to, to use Consolekit in these instances as well by using less-friendly (and less memorable) `dbus-send` commands for poweroff actions instead of `sudo` commands (not covered here).

## Regional Repository (Suggested)

Switching from the system default repository in Germany to another official repository nearer to you (see listing) will help relieve the workload on the main server and may speed your system updates. See this guide.

## Troubleshooting

### My hardware isn't showing up!

If you have issues with hardware and it doesn't appear with **dmesg**, you may need to install the **linux-firmware** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/linux-firmware>) and/or **linux-firmware-network** (<https://github.com/void-linux/void-packages/tree/master/srcpkgs/linux-firmware-network>) packages:

```
# xbps-install linux-firmware
```

```
# xbps-install linux-firmware-network
```

## What's next?

To further customize your new Void system, you should have a look at our Guides.

Retrieved from "[https://wiki.voidlinux.org/index.php?title=Post\\_Installation&oldid=11507](https://wiki.voidlinux.org/index.php?title=Post_Installation&oldid=11507)"

Category: Installation

---

■ This page was last edited on 18 January 2019, at 20:46.

- Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.