DustArch

DustVoice's Arch Linux from scratch

David Holland

Version 4.2, 2019-12-28

Table Of Contents

Inside the `archiso`

This section is aimed at providing help with the general installation of the customized Arch Linux from within official Arch Linux image.

Sync up `pacman`

First of all we need to sync up `pacman` in order to be able to install packages

```
root@archiso ~ # pacman -Sy
```

Formatting the drive

First you have to list all the available drives by issuing

```
root@archiso ~ # fdisk -l
```

The output of `fdisk -l` is dependent on your system configuration.

In my case, the partition I want to install the root file system on is `/dev/sdb2`. `/dev/sdb3` will be my `swap` partition.

A `swap` size **twice the size of your RAM** is recommended by a lot of people. You should make the `swap` size **at least your RAM size** though.

If you haven't yet partitioned your disk, please refer to the general partitioning tutorial in the arch-wiki.

Now we need to format the partitions accordingly

```
root@archiso ~ # mkfs.ext4 /dev/sdb2
root@archiso ~ # mkswap /dev/sdb3
```

After doing that, we can turn on the `swap` and `mount` the root partition.

```
root@archiso ~ # swapon /dev/sdb3
root@archiso ~ # mount /dev/sdb2 /mnt
```

If you have an additional `EFI system partition`, because of a *UEFI* - *GPT* setup or e.g. an existing Windows installation, which we will assume to be located under `/dev/sda2` (`/dev/sda` is the disk of my Windows install), you'll have to `mount` this partition to the new systems `/boot` folder

```
root@archiso ~ # mkdir /mnt/boot
root@archiso ~ # mount /dev/sda2 /mnt/boot
```

Preparing the `chroot` environment

First it might make sense to edit `/etc/pacman.d/mirrorlist` to move the mirror(s) geographically closest to you to the top.

After that we can either install the **bare minimum packages** needed

```
root@archiso ~ # pacstrap /mnt base linux linux-firmware
```

or install **all packages present** on the archiso, which makes sense in our case

```
root@archiso ~ # pacstrap /mnt base base-devel linux linux-firmware $(pacman -Qq | tr
'\n' ' ')
```

This could take quite some time depending on your Internet connection speed.

After that generate a `fstab` using `genfstab`

```
root@archiso ~ # genfstab -U /mnt >> /mnt/etc/fstab
```

and you're ready to enter the `chroot` environment.

Entering the `chroot`

```
root@archiso ~ # arch-chroot /mnt
```

Et Voila! You successfully `chroot`ed inside your new system, greeted by a `bash` prompt.

Installing additional packages

First off you'll probably need an editor. I'll use `neovim`

```
[root@archiso /]# pacman -S neovim
```

After that we'll make sure we get ourselves some basic utilities and enable the `NetworkManager.service` service, in order for the Internet connection to work upon booting into our fresh system later on.

```
[root@archiso /]# pacman -S sudo iputils dhcpcd dhclient grub dosfstools os-prober
mtools networkmanager networkmanager-openvpn networkmanager-openconnect
[root@archiso /]# systemctl enable NetworkManager.service
```

Furthermore you'll also need to make sure `polkit` is installed

```
[root@archiso /]# pacman -S polkit
```

and then create a file `/etc/polkit-1/rules.de/50-org.freedesktop.NetworkManager.rules` to enable users of the `network` group to add new networks without `sudo`.

/etc/polkit-1/rules.de/50-org.freedesktop.NetworkManager.rules

```
polkit.addRule(function(action, subject) {
    if (action.id.indexOf("org.freedesktop.NetworkManager.") == 0 &&
subject.isInGroup("network")) {
        return polkit.Result.YES;
    }
});
```

If you use `UEFI`, you'll also need

```
[root@archiso /]# pacman -S efibootmgr
```

Master of time

After that you have to set your timezone and update the system clock.

Generally speaking, you can find all the different timezones under `/usr/share/zoneinfo`. For me it is `/usr/share/zoneinfo/Europe/Berlin`. Now I would have to issue

```
[root@archiso /]# ln -s /usr/share/zoneinfo/Europe/Berlin /etc/localtime
[root@archiso /]# hwclock --systohc --utc
```

Now you can also enable time synchronization over network and check that everything is alright

```
[root@archiso /]# timedatectl set-timezone Europe/Berlin
[root@archiso /]# timedatectl set-ntp true
[root@archiso /]# timedatectl status
```

Master of locales

Now you have to generate your locale information.

For that you have to edit `/etc/locale.gen` and uncomment the `locale` lines you want to enable.

I recommend to always uncomment `en_US.UTF-8 UTF8` for development purposes, even if you want to use another language primarily.

In my case I only uncommented the `en_US.UTF-8 UTF8` line

/etc/locale.gen

```
en_US.UTF-8 UTF8
```

After that you still have to actually generate the locales by issuing

```
[root@archiso /]# locale-gen
```

and set the locale

```
[root@archiso /]# localectl set-locale LANG="en_US.UTF-8"
```

and we're done with this part.

Naming your machine

Now we can set the `hostname` and add `hosts` entries.

`hostname`

To change the `hostname`, simply edit `/etc/hostname`, enter the desired name, then save and quit.

/etc/hostname

```
DustArch
```

`hosts`

Now we need to specify some `hosts` entries by editing `/etc/hosts`

/etc/hosts

```
# Static table lookup for hostnames.
# See hosts(5) for details.

127.0.0.1   localhost           .
::1         localhost           .
127.0.1.1   DustArch.localhost  DustArch
```

```
[root@archiso /]# exit
root@archiso ~ # arch-chroot /mnt
```

User setup

Now you should probably change the default `root` password and create a new non-`root` user for yourself, as using your new system purely through the native `root` user is not recommended from a security standpoint.

Give `root` a password

To change the password for the current user (the `root` user) do

```
[root@DustArch /]# passwd
```

and choose a new password.

Create a personal user

We are going to make sure the `fish` shell is installed, create a new user, set the password for this user, make sure the `sudo` package is installed and allow the `wheel` group `sudo` access.

```
[root@DustArch /]# pacman -S fish
[root@DustArch /]# useradd -m -p "" -G
"adm,audio,floppy,kvm,log,lp,network,rfkill,scanner,storage,users,optical,power,wheel"
-s /usr/bin/fish dustvoice
[root@DustArch /]# passwd dustvoice
[root@DustArch /]# pacman -S sudo
```

We now have to allow the `wheel` group `sudo` access.

For that we edit `/etc/sudoers` and uncomment the `%wheel` line

/etc/sudoers

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

You could also add a new line below the `root` line

/etc/sudoers

```
root ALL=(ALL) ALL
```

with your new username

/etc/sudoers

```
dustvoice ALL=(ALL) ALL
```

to solely grant yourself `sudo` privileges.

Preparing to boot

Now onto installing the boot manager. We will use `grub` in this guide.

First make sure, all the required packages are installed

```
[root@DustArch /]# pacman -S grub dosfstools os-prober mtools
```

and if you want to use `UEFI`, also

```
[root@DustArch /]# pacman -S efibootmgr
```

BIOS

If you chose the `BIOS - MBR` variation, you'll have to **do nothing special**

If you chose the `BIOS - GPT` variation, you'll have to **have a `+1M` boot partition** created with the partition type set to `BIOS boot`.

In both cases you'll have to **run the following comman** now

```
[root@DustArch /]# grub-install --target=i386-pc /dev/sdb
```

It should obvious that you would need to replace `/dev/sdb` with the disk you actually want to use. Note however that you have to specify a **disk** and **not a partition**, so **no number**.

`UEFI`

If you chose the `UEFI - GPT` variation, you'll have to **have the `EFI System Partition` mounted** at `/boot` (where `/dev/sda2` is the partition holding said `EFI System Partition` in my particular setup)

Now **install `grub` to the `EFI System Partition`**

```
[root@DustArch /]# grub-install --target=x86_64-efi --efi-directory=/boot --bootloader
-id=grub --recheck
```

If you've planned on dual booting arch with Windows and therefore reused the `EFI System Partition` created by Windows, you might not be able to boot to grub just yet.

In this case, boot into Windows, open a `cmd` window as Administrator and type in

```
bcdedit /set {bootmgr} path \EFI\grub\grubx64.efi
```

To make sure that the path is correct, you can just

```
[root@DustArch /]# ls /boot/EFI/grub
```

to make sure, that the `grubx64.efi` file is really there.

`grub` config

In all cases, you now have to create the main configuration file.

But before we actually generate it, we'll make some changes to the default `grub` settings.

Adjust the timeout

First of all, I want my `grub` menu to wait indefinitely for my command to boot a OS.

/etc/default/grub

```
GRUB_TIMEOUT=-1
```

I decided on this, because I'm dual booting with Windows and after Windows updates itself, I don't want to accidentally boot into my Arch Linux, just because I wasn't quick enough to select it from the `grub` menu.

Of course you can set this parameter to whatever you want.

Another way of achieving what I described previously, would be to make `grub` remember the last selection. For that we would have to adjust the file accordingly

/etc/default/grub

```
GRUB_DEFAULT=saved
GRUB_SAVEDEFAULT="true"
```

Enable the recovery

After that I also want the recovery option showing up, which means that besides the standard and fallback images, also the recovery one would show up.

/etc/default/grub

```
GRUB_DISABLE_RECOVERY=false
```

NVIDIA fix

Now, as I'm using the binary nvidia driver for my graphics card, I also want to make sure, to revert `grub` back to text mode, after I selected a boot entry.

/etc/default/grub

```
GRUB_GFXPAYLOAD_LINUX=text
```

Add power options

I also want to add 2 new menu entries, to enable me to shut down the PC, or reboot it, right from the `grub` menu.

/etc/grub.d/40-custom

```
menuentry '=> Shutdown' {
    halt
}

menuentry '=> Reboot' {
    reboot
}
```

Installing `memtest`

As I want all possible options to possibly troubleshoot my PC, without booting into a live image, right there in my `grub` menu, I also want to have a memory tester there.

`BIOS`

For a `BIOS` setup, you'll need `memtest86+`

```
[root@DustArch /]# pacman -S memtest86+
```

`UEFI`

For a `UEFI` setup, you'll need `memtest86-efi`.

In order to install that `AUR` package, you'll need to switch to your normal user, because `makepkg` doesn't run as root.

```
[root@DustArch /]# pacman -S base-devel
[root@DustArch /]# sudo -iu dustvoice
[dustvoice@DustArch ~]$ git clone https://aur.archlinux.org/memtest86-efi
[dustvoice@DustArch ~]$ cd memtest86-efi
[dustvoice@DustArch ~/memtest86-efi]$ makepkg -si
[dustvoice@DustArch ~/memtest86-efi]$ cd ..
[dustvoice@DustArch ~]$ rm -rf memtest86-efi
[dustvoice@DustArch ~]$ exit
```

Now we still need to tell `memtest86-efi` how to install itself.

```
[root@DustArch /]# memtest86-efi -i
```

Now select option 3, to install it as a `grub2` menu item.

Generating the config

Now we can finally generate our `grub.cfg`

```
[root@DustArch /]# grub-mkconfig -o /boot/grub/grub.cfg
```

Now you’re good to boot into your new system

Inside the `DustArch`

Someone there?

First we have to check if the network interfaces are set up properly

```
dustvoice@DustArch ~> ip link
```

This outputs the interface status report.

To make sure that you really have a working *Internet* connection, issue

```
dustvoice@DustArch ~> ping archlinux.org
```

Everything should run smoothly if you have a wired connection. If there is still no connection try restarting the `NetworkManager.service` service

```
dustvoice@DustArch ~> sudo systemctl restart NetworkManager.service
```

and then try `ping` again.

If you're indeed trying to utilize a Wi-Fi connection, use `nmcli`, the `NetworkManager` command line tool, or `nmtui`, the `NetworkManager` terminal user interface, to connect to a Wi-Fi network.

I never got `nmtui` to behave like I wanted it to, in my particular case at least, which is the reason why I use `nmcli` or the GUI tools.

First make sure, the scanning of nearby Wi-Fi networks is enabled for your Wi-Fi device

```
dustvoice@DustArch ~> nmcli r
```

and if not, enable it

```
dustvoice@DustArch ~> nmcli r wifi on
```

Now make sure your Wi-Fi interface appears under

```
dustvoice@DustArch ~> nmcli d
```

Rescan for available networks

```
dustvoice@DustArch ~> nmcli d wifi rescan
```

and list all found networks

```
dustvoice@DustArch ~> nmcli d wifi list
```

After that connect to the network

```
dustvoice@DustArch ~> nmcli d wifi connect --ask
```

Now try `ping`ing again.

Update and upgrade

After making sure that you have established an Internet connection, you can then proceed to update and upgrade all installed packages by issuing

```
dustvoice@DustArch ~> sudo pacman -Syu
```

Enabling the `multilib` repository

In order to make 32-bit packages available to `pacman`, we'll need to enable the `multilib` entry in `/etc/pacman.conf` first. Simply uncomment

/etc/pacman.conf

```
[multilib]
Include = /etc/pacman.d/mirrorlist
```

Setting the correct shell

I'll be using the `fish` shell.

We already set the correct shell for the `dustvoice` user in the Create a personal user step, but I want to use `fish` for the `root` user too, so I'll have to change `root`'s default shell to it.

```
dustvoice@DustArch ~> chsh -s /usr/bin/fish root
```

Don't worry about the looks by the way, we're gonna change all that in just a second.

Version control

Next you'll probably want to install `git`. Just do

```
dustvoice@DustArch ~> sudo pacman -S git
```

and you're good to go. We'll care about the `.gitconfig` in just a second.

Security is important

If you've followed the tutorial using a recent version of Arch Linux, you'll probably already have the most recent version of `gnupg` installed by default. Just to make sure, issue

```
dustvoice@DustArch ~> sudo pacman -S gnupg
```

Smartcard shenanigans

After that you'll still have to setup `gnupg` correctly. In my case I have my private keys stored on a smartcard. To use it, I'll have to install some packages first

```
dustvoice@DustArch ~> sudo pacman -S pcsclite libusb-compat ccid opensc
```

and then enable and start the `pcscd` service

```
dustvoice@DustArch ~> sudo systemctl enable pcscd.service
dustvoice@DustArch ~> sudo systemctl start pcscd.service
```

Additional required tools

To minimize the effort required in the following steps, we'll install most of the required tools now

```
dustvoice@DustArch ~> pacman -S make cmake clang jdk-openjdk pyhton pyhton-pip pass
openssh
```

Setting up a `home` environment

In this step we're going to setup a home environment for both the `root` and my personal `dustvoice` user.

In my case these 2 home environments are mostly equivalent, which is why I'll execute the following commands as the `dustvoice` user first and then switch to the `root` user and repeat the same commands.

In my case, I want to access all my `git` repositories with my `gpg` key on my smartcard. For that I have to configure the `gpg-agent` though. So I will have to reside to first use the `https` url and later change the url in the corresponding `.git/config` file.

Use `dotfiles` for a base config

```
dustvoice@DustArch ~> git init
dustvoice@DustArch ~> git remote add origin https://github.com/DustVoice/dotfiles.git
dustvoice@DustArch ~> git fetch
dustvoice@DustArch ~> git reset origin/master --hard
dustvoice@DustArch ~> git branch --set-upstream-to=origin/master master
```

Set up `gpg`

Before we'll be able to update the `submodules` (`nvim` config files and `pass`word-store), we will have to setup our `gpg` key as a `ssh` key

```
[I] dustvoice@DustArch ~>
$ chmod 700 .gnupg
[I] dustvoice@DustArch ~>
$ gpg --card-status
[I] dustvoice@DustArch ~>
$ gpg --card-edit
(insert) gpg/card> fetch
(insert) gpg/card> q
[I] dustvoice@DustArch ~>
$ gpg-connect-agent updatestartuptty /bye
[I] dustvoice@DustArch ~>
$ git remote set-url origin git@github.com:DustVoice/dotfiles.git
[I] dustvoice@DustArch ~>
$ exit
```

You would have to adapt the `keygrip` present in the `~/.gnupg/sshcontrol` file to your specific `keygrip`, retrieved with `gpg -K --with-keygrip`.

Finalize the `dotfiles`

Now log back in and continue

```
[I] dustvoice@DustArch ~
$ git submodule update --init --recursive
[I] dustvoice@DustArch ~
$ cd .config/nvim
[I] dustvoice@DustArch ~/.config/nvim
$ echo 'let g:platform = "linux"' >> platform.vim
[I] dustvoice@DustArch ~/.config/nvim
$ echo 'let g:use_autocomplete = 3' >> custom.vim
[I] dustvoice@DustArch ~/.config/nvim
$ echo 'let g:use_clang_format = 1' >> custom.vim
[I] dustvoice@DustArch ~/.config/nvim
$ echo 'let g:use_font = 0' >> custom.vim
[I] dustvoice@DustArch ~/.config/nvim
$ pip3 install neovim
[I] dustvoice@DustArch ~/.config/nvim
$ nvim --headless +PlugInstall +qa
[I] dustvoice@DustArch ~/.config/nvim
$ cd plugged/YouCompleteMe
[I] dustvoice@DustArch ~/.config/nvim/plugged/YouCompleteMe
$ python3 install.py --clang-completer --java-completer
[I] dustvoice@DustArch ~/.config/nvim/plugged/YouCompleteMe
$ cd ~
```

`gpg-agent` forwarding

Now there is only one thing left to do, in order to make the `gpg` setup complete: `gpg-agent` forwarding over ssh. This is very important for me, as I want to use my smartcard on my development server too, which requires me, to forward/tunnel my `gpg-agent` to my remote machine.

First of all, I want to setup a config file for `ssh`, as I don’t want to pass all parameters manually to ssh every time.

~/.ssh/config

```
Host <connection name>
    HostName <remote address>
    ForwardAgent yes
    ForwardX11 yes
    RemoteForward <remote agent-socket> <local agent-extra-socket>
    RemoteForward <remote agent-ssh-socket> <local agent-ssh-socket>
```

You would of course, need to adapt the content in between the < and > brackets.

To get the paths needed as parameters for RemoteForward, issue

```
[I] dustvoice@DustArch ~
$ !gpgconf --list-dirs
```

Now you'll still need to enable some settings on the remote machine.

/etc/ssh/sshd_config

```
StreamLocalBindUnlink yes
AllowAgentForwarding yes
X11Forwarding yes
```

Now just restart your remote machine and you're ready to go.

JUCE and FRUT

Your personal environment will be complete, after getting JUCE and FRUT

```
[I] dustvoice@DustArch ~
$ git clone https://github.com/WeAreROLI/JUCE.git
[I] dustvoice@DustArch ~
$ cd JUCE
[I] dustvoice@DustArch ~/JUCE
$ git checkout develop
[I] dustvoice@DustArch ~/JUCE
$ cd ..
[I] dustvoice@DustArch ~
$ git clone https://github.com/McMartin/FRUT.git
```

Back to your roots

As mentioned before, you would now switch to the root user, either by logging in as root, or by using

```
[I] dustvoice@DustArch ~
$ sudo -iu root
```

Now go back to Setting up a home environment to repeat all commands for the root user.

A native login would be better compared to sudo -iu root, as there could be some complications, like already running gpg-agent instances, etc., which you would need to manually resolve, when using sudo -iu root.

Password management

I'm using `pass` as my password manager. As we already installed it in the Additional required tools step and updated the `submodule` that holds our `.password-store`, there is nothing left to do in this step

`python`

Python has become really important for a magnitude of use cases. We need `python3` in particular as well as `pip` for it.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S python python-pip
```

For `asciidoctor`, which will be installed in just a second, we also need to install the `pygments` module

```
[I] dustvoice@DustArch ~
$ sudo pip3 install pygments
```

`ruby` & `asciidoctor`

In order to use `asciidoctor`, we have to install `ruby` and `rubygems`. After that we can install `asciidoctor` and all its required gems.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ruby rubygems
[I] dustvoice@DustArch ~
$ gem install asciidoctor --pre
[I] dustvoice@DustArch ~
$ gem install asciidoctor-pdf --pre
[I] dustvoice@DustArch ~
$ gem install asciidoctor-epub3 --pre
[I] dustvoice@DustArch ~
$ gem install pygments.rb --pre
```

Now the only thing left, in my case at least, is adding `~/.gem/ruby/2.6.0/bin` to your path.

Please note that if you run a ruby version different from `2.6.0`, you have to use the `bin` path for that version.

For `fish` you'll want to run the following command

```
[I] dustvoice@DustArch ~
$ set -U fish_user_paths $fish_user_paths ~/.gem/ruby/2.6.0/bin
```

If you use another shell than `fish`, you might have to do something different to add a directory to your `PATH`.

Using `JUCE`

In order to use `JUCE`, you'll need to have some dependency packages installed

```
[I] dustvoice@DustArch ~
$ sudo pacman -S clang gcc freeglut alsa-lib gnutls libcurl-gnutls freetype2 jack2
libx11 libxcomposite libxinerama libxrandr mesa webkit2gtk
```

If you want to use every feature of `JUCE` you'll need to install 2 more packages

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ladspa lib32-freeglut
```

Additional development tools

Here are just some examples of development tools one could install in addition to what we already have.

Code formatting

We already have `clang-format` as a code formatter, but this only works for `C`-family languages. For `java` stuff, we can use `astyle`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S astyle
```

Documentation

To generate a documentation from source code, I mostly use `doxygen`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S doxygen
```

Build tools

In addition to `make`, I'll often times use `ninja` for my builds

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ninja
```

`fstab`

In my case, I'm sharing an `exFat` partition between my `DustArch` and my Windows. This was a result of some major inconvenience because of some weird `NTFS` permission stuff, which apparently Windows didn't like. Since I've avoided directly writing to Windows partitions since then, I'll quickly demonstrate what `fstab` entries I have and why

/etc/fstab

```
UUID=e26de048-6147-42e5-a34b-59f1a50621bb       /               ext4
rw,relatime             0 1

UUID="C8E3-A0FD"                                /boot           vfat
defaults                0 1

UUID="DC88-5A4E"                                /mnt/projects   exfat
rw,relatime             0 0

UUID=7A16569B51903310                           /mnt/data       ntfs
ro,nosuid,nodev,noauto  0 0
```

The

1. entry should be pretty straight forward. It's my root partition of my `DustArch` install.
2. entry is quite important too. It's my `EFI System Partition`, which gets mounted at boot time, in order to prevent kernel orphaning, which means, that the kernel version installed on the system doesn't match the one on the `boot` partition.
3. entry is my shared `exFat` partition, which we are allowed to write to.
4. entry is important, because of the options. These options prevent me from modifying files on that `NTFS` partition.

Audio

Well, why wouldn't you want audio…

`alsa`

You're probably better off using `pulseaudio` and/or `jack`.

To quickly setup audio this way, install `alsa` and `alsa-utils`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S alsa alsa-utils
```

Now choose the sound card you want to use

```
[I] dustvoice@DustArch ~
$ cat /proc/asound/cards
```

and then create `/etc/asound.conf`

/etc/asound.conf

```
defaults.pcm.card 2
defaults.ctl.card 2
```

It should be apparent, that you would have to switch out `2` with the number corresponding to the sound card you want to use.

`pulseaudio`

Some applications require `pulseaudio`, or work better with it, for example `discord`, so it might make sense to use `pulseaudio`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S pulseaudio pulsemixer pavucontrol
```

For enabling real-time priority for `pulseaudio` on Arch Linux, please make sure your user is part of the `audio` group and edit the file `/etc/pulse/daemon.conf`, so that you uncomment the lines

/etc/pulse/daemon.conf

```
high-priority = yes
nice-level = -11

realtime-scheduling = yes
realtime-priority = 5
```

If your system can handle the load, you can also increase the remixing quality, by changing the `resample-method`

/etc/pulse/daemon.conf

```
resample-method = speex-float-10
```

Of course a restart of the `pulseaudio` daemon is necessary to reflect the changes you just made

```
[I] dustvoice@DustArch ~
$ pulseaudio --kill
[I] dustvoice@DustArch ~
$ pulseaudio --start
```

`jack`

If you either want to manually control audio routing, or if you use some kind of audio application like `ardour`, you'll probably want to use `jack`.

To install `jack` and a GUI to configure it, just do

```
[I] dustvoice@DustArch ~
$ sudo pacman -S jack2 cadence
```

If you also want to use `pulseaudio` applications, that don't have native support for `jack`, you'll need to install `pulseaudio-jack`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S pulseaudio-jack
```

Audio handling

To also play audio, we need to install some other packages too

```
[I] dustvoice@DustArch ~
$ sudo pacman -S soc libao libmad libid3tag wavpack libpulse opus file twolame
```

Now you can simply do

```
[I] dustvoice@DustArch ~
$ play audio.wav
[I] dustvoice@DustArch ~
$ play audio.mp3
```

etc. to play audio.

Bluetooth

To set up Bluetooth, we need to install the `bluez` and `bluez-util` packages in order to have at least a command line utility `bluetoothctl` to configure connections

```
[I] dustvoice@DustArch ~
$ sudo pacman -S bluez bluez-utils
```

Now we need to check if the `btusb` kernel module was already loaded

```
[I] dustvoice@DustArch ~
$ sudo lsmod | grep btusb
```

After that we'll enable and start the `bluetooth.service` service

```
[I] dustvoice@DustArch ~
$ sudo systemctl enable bluetooth.service
[I] dustvoice@DustArch ~
$ sudo systemctl start bluetooth.service
```

To use `bluetoothctl` and get access to the Bluetooth device of your PC, your user needs to be a member of the `lp` group.

Now simply enter `bluetoothctl`

```
[I] dustvoice@DustArch ~
$ bluetoothctl
```

In most cases your Bluetooth interface will be preselected and defaulted, but in some cases, you might need to first select the Bluetooth controller

```
(insert) [DustVoice]# list
(insert) [DustVoice]# select <MAC_address>
```

After that, power on the controller

```
(insert) [DustVoice]# power on
```

Now enter device discovery mode

```
(insert) [DustVoice]# scan on
```

and list found devices

```
(insert) [DustVoice]# devices
```

You can turn device discovery mode off again, after your desired device has been found

```
(insert) [DustVoice]# scan off
```

Now turn on the agent

```
(insert) [DustVoice]# agent on
```

and pair with your device

```
(insert) [DustVoice]# pair <MAC_address>
```

If your device doesn’t support PIN verification you might need to manually trust the device

```
(insert) [DustVoice]# trust <MAC_address>
```

Finally connect to your device

```
(insert) [DustVoice]# connect <MAC_address>
```

If your device is an audio device, of some kind you might have to install `pulseaudio-bluetooth` and append 2 lines to `/etc/pulse/system.pa` as well.

So first install `pulseaudio-bluetooth`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S pulseaudio-bluetooth
```

append the following 2 lines

/etc/pulse/system.pa

```
load-module module-bluetooth-policy
load-module module-bluetooth-discover
```

and restart `pulseaudio`

```
[I] dustvoice@DustArch ~
$ pulseaudo --kill
[I] dustvoice@DustArch ~
$ pulseaudo --start
```

If you want a GUI to do all of this, just install `blueman` and launch `blueman-manager`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S blueman
```

Graphical desktop environment

If you decide, that you want to use a graphical desktop environment, you have to install additional packages in order for that to work.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S xorg xorg-xinit xorg-drivers i3 i3status rofi ttf-hack xfce4-terminal
alsa alsa-utils arandr
```

NVIDIA

If you also want to use NVIDIA functionality, for example for `davinci-resolve`, you'll most likely need to install their proprietary driver

```
[I] dustvoice@DustArch ~
$ sudo pacman -S nvidia nvidia-utils nvidia-settings opencl-nvidia
```

You would have to reboot sooner or later after installing the NVIDIA drivers.

Also to get the best performance, at least for something like screen capturing in `obs`, go to **X Server Display Configuration** inside `nvidia-settings`, switch to **Advanced** and enable **Force Composition Pipeline**, as well as **Force Full Composition Pipeline**.

Launching the graphical environment

After that you can now do `startx` in order to launch the graphical environment.

If anything goes wrong in the process, remember that you can press **Ctrl**+**Alt**+<**Number**> to switch `tty`s.

The NVIDIA way

If you're using an NVIDIA graphics card, you might want to use `nvidia-xrun` instead of `startx`. This has the advantage, of the `nvidia` kernel modules, as well as the `nouveau` ones not loaded at boot time, thus saving power. `nvidia-xrun` will then load the correct kernel modules and run the `.nvidia-xinitrc` script in your home directory (for more file locations look into the documentation for `nvidia-xrun`).

At the time of writing, `nvidia-xrun` needs `sudo` permissions before executing its task.

Simply install `nvidia-xrun`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S nvidia bbswitch
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/nvidia-xrun.git
[I] dustvoice@DustArch ~
$ cd nvidia-xrun
[I] dustvoice@DustArch ~/nvidia-xrun
$ makepkg -si
[I] dustvoice@DustArch ~/nvidia-xrun
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf nvidia-xrun
```

If your hardware doesn't support `bbswitch`, you would need to run

```
[I] dustvoice@DustArch ~
$ sudo pacman -S nvidia
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/nvidia-xrun-pm.git
[I] dustvoice@DustArch ~
$ cd nvidia-xrun-pm
[I] dustvoice@DustArch ~/nvidia-xrun-pm
$ makepkg -si
[I] dustvoice@DustArch ~/nvidia-xrun-pm
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf nvidia-xrun-pm
```

instead.

Now we need to blacklist **both** `nouveau` **and** `nvidia` kernel modules.

To do that, we first have to find out, where our active `modprobe.d` directory is located. There are 2 possible locations, generally speaking: `/etc/modprobe.d` and `/usr/lib/modprobe.d`. In my case it was the latter, which I could tell, because this directory already had files in it.

Now I'll create a new file named `nvidia-xrun.conf` and write the following into it

/usr/lib/modprobe.d/nvidia-xrun.conf

```
blacklist nvidia
blacklist nvidia-drm
blacklist nvidia-modeset
blacklist nvidia-uvm
blacklist nouveau
```

With this config in place,

```
[I] dustvoice@DustArch ~
$ lsmod | grep nvidia
```

and

```
[I] dustvoice@DustArch ~
$ lsmod | grep nouveau
```

should return no output. Else you might have to place some additional entries into the file.

Of course, you'll need to reboot, after blacklisting the modules and before issuing the 2 commands mentioned.

If you installed `nvidia-xrun-pm` instead of `nvidia-xrun` and `bbswitch`, you might want to also enable the `nvidia-xrun-pm` service

```
[I] dustvoice@dustArch ~
$ sudo systemctl enable nvidia-xrun-pm.service
```

The required `.nvidia-xinitrc` file, mentioned previously, should already be provided in the `dotfiles` repository.

Now instead of `startx`, just run `nvidia-xrun`, enter your `sudo` password and you're good to go.

GUI Software

As you now have a working graphical desktop environment, you might want to install some software to utilize your newly gained power.

Desktop background

You might want to consider installing `nitrogen`, in order to be able to set a background image

```
[I] dustvoice@DustArch ~
$ sudo pacman -S nitrogen
```

Compositing software

To get buttery smooth animation as well as e.g. smooth video playback in `brave` without screen tearing, you might want to consider using a compositor, in my case one named `picom`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S picom
```

Now edit the file `~/.config/i3/config` and uncomment the `picom` line in order to start `picom` with `i3`.

In order for obs' screen capture to work correctly, you need to kill picom completely before using obs.

```
[I] dustvoice@DustArch ~
$ pkill picom
```

or

```
[I] dustvoice@DustArch ~
$ ps aux | grep picom
[I] dustvoice@DustArch ~
$ kill -9 <pid>
```

networkmanager applet

To install the NetworkManager applet, which lives in your tray and provides you with a quick method to connect to different networks, you have to install the network-manager-applet package

```
[I] dustvoice@DustArch ~
$ sudo pacman -S network-manager-applet
```

Now you can start the applet with

```
[I] dustvoice@DustArch ~
$ nm-applet &
```

If you want to edit the network connections with a more full screen approach, you can also launch nm-connection-editor.

The nm-connection-editor doesn't search for available Wi-Fis. You would have to set up a Wi-Fi connection completely by hand, which could be desirable depending on how difficult to set up your Wi-Fi is.

Keyboard

To show, which keyboard layout and variant is currently in use, you can use xkblayout-state, which you can acquire from the AUR

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/xkblayout-state.git
[I] dustvoice@DustArch ~
$ cd xkblayout-state
[I] dustvoice@DustArch ~/xkblayout-state
$ makepkg -si
[I] dustvoice@DustArch ~/xkblayout-state
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf xkblayout-state
```

Now simply issue the `layout` alias, provided by our custom `fish` configuration.

X clipboard

To copy something from the terminal to the `xorg` clipboard, use `xclip`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S xclip
[I] dustvoice@DustArch ~
$ xclip some_random_text
```

Taking screen shots

For this functionality, especially in combination with `rofi`, use `scrot`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S scrot
```

`scrot ~/Pictures/filename.png` then saves the screen shot under `~/Pictures/filename.png`.

Image viewer

Now that we can create screen shots, we might also want to view those

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ristretto
[I] dustvoice@DustArch ~
$ ristretto filename.png
```

File manager

You probably also want to use a file manager. In my case, `thunar`, the `xfce` file manager, worked best.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S thunar
```

To also be able to mount removable drives, without being root or using sudo, and in order to have a GUI for mounting stuff, you would need to install gigolo and gvfs

```
[I] dustvoice@DustArch ~
$ sudo pacman -S gigolo gvfs
```

Android file transfer

To furthermore enable the transfer of files between your PC and your android phone, you'll have to install mtp and gvfs-mtp

```
[I] dustvoice@DustArch ~
$ sudo pacman -S libmtp gvfs-mtp
```

Now you should be able to see your phone inside either thunar, or gigolo.

If you want to access the android's file system from the command line, you will need to either install and use simple-mtpfs, or adb

simple-mtpfs

Install simple-mtpfs

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/simple-mtpfs.git
[I] dustvoice@DustArch ~
$ cd simple-mtpfs
[I] dustvoice@DustArch ~/simple-mtpfs
$ makepkg -si
[I] dustvoice@DustArch ~/simple-mtpfs
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf simple-mtpfs
```

edit /etc/fuse.conf to uncomment

/etc/fuse.conf

```
user_allow_other
```

and mount the android device

```
[I] dustvoice@DustArch ~
$ simple-mtpfs -l
[I] dustvoice@DustArch ~
$ mkdir ~/mnt
[I] dustvoice@DustArch ~
$ simple-mtpfs --device <number> ~/mnt -allow_other
```

and respectively unmount it

```
[I] dustvoice@DustArch ~
$ fusermount -u mnt
[I] dustvoice@DustArch ~
$ rmdir mnt
```

`adb`

Install `adb`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S adb
```

kill the `adb` server, if it is running

```
[I] dustvoice@DustArch ~
$ adb kill-server
```

If the server is currently not running, `adb` will output an error with a `Connection refused` message.

Now connect your phone, unlock it and start the `adb` server

```
[I] dustvoice@DustArch ~
$ adb start-server
```

If the PC is unknown to the android device, it will display a confirmation dialog. Accept it and ensure that the device was recognized

```
[I] dustvoice@DustArch ~
$ adb devices
```

Now you can `push`/`pull` files.

```
[I] dustvoice@DustArch ~
$ adb pull /storage/emulated/0/DCIM/Camera/IMG.jpg .
[I] dustvoice@DustArch ~
$ adb push IMG.jpg /storage/emulated/0/DCIM/Camera/IMG2.jpg
[I] dustvoice@DustArch ~
$ adb kill-server
```

Of course you would need to have the *developer options* unlocked, as well as the *USB debugging* option enabled within them, for `adb` to even work.

Archive manager

As we now have a file manager, it might be annoying, to open up a terminal every time you simply want to extract an archive of some sort. That's why we'll install `xarchiver`.

In order for `xarchiver` to work at its full potential, we're first gonna install some additional archive types

```
[I] dustvoice@DustArch ~
$ sudo pacman -S p7zip zip unrar cpio
```

Now we can proceed to install `xarchiver`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S xarchiver
```

Partition management

You may also choose to use a graphical partitioning software instead of `fdisk` or `cfdisk`. For that you can install `gparted`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S gparted
```

PDF viewer

As we've installed `asciidoctor-pdf` previously, you might be wondering how you are supposed to open the generated PDFs. There are two ways.

Using the GUI

Installing `mupdf` is as simple as issuing

```
[I] dustvoice@DustArch ~
$ sudo pacman -S mupdf
```

If you want to have changes made to the PDF reflected immediately in the viewer, you would need `evince` instead

```
[I] dustvoice@DustArch ~
$ sudo pacman -S evince
```

Using the framebuffer

If you want to not always use the graphical desktop with `mupdf`, you might be interested in the `fbgs` software.

This software renders a PDF document using the native framebuffer. To install it simply do

```
[I] dustvoice@DustArch ~
$ pacman -S fbida ghostscript
```

and to view this PDF document (`Documentation.pdf`) for example, you would run

```
[I] dustvoice@DustArch ~
$ fbgs Documentation.pdf
```

You can view all the controls by pressing `h`.

Web browser

As you're already using a GUI, you also might be interested in a web browser. In my case, I'll install `brave` from the `AUR`, as well as `browserpass` from the official repositories, in order to use my passwords in `brave`.

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/brave-bin.git
[I] dustvoice@DustArch ~/brave-bin
$ makepkg -si
[I] dustvoice@DustArch ~/brave-bin
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf brave-bin
[I] dustvoice@DustArch ~
$ sudo pacman -S browserpass
```

Now we still have to setup `browserpass`

```
[I] dustvoice@DustArch ~
$ cd /usr/local/lib/browserpass
[I] dustvoice@DustArch /usr/local/lib/browserpass
$ make hosts-brave-user
[I] dustvoice@DustArch /usr/local/lib/browserpass
$ make policies-brave-user
[I] dustvoice@DustArch /usr/local/lib/browserpass
$ cd ~
```

Now the only thing left is, to fire up `brave` and install the `browserpass` extension from the chrome store.

Entering the dark side

You might want to be completely anonymous whilst browsing the web at some point. Although this shouldn't be your only precaution, using `tor-browser` would be the first thing to do

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/tor-browser.git
[I] dustvoice@DustArch ~
$ cd tor-browser
[I] dustvoice@DustArch ~/tor-browser
$ makepkg -si
[I] dustvoice@DustArch ~/tor-browser
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf tor-browser
```

Office utilities

For now we'll install `libreoffice-fresh`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S libreoffice-fresh
```

Printing

In order for printing to work with my printer, I had to install `avahi`, `cups`, `cups-pdf`, `nss-mdns` and the correspoding driver for my printer. In order to be able to print from the `gtk` print dialog, we'll also need to install `system-config-printer` and `print-manager`.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S avahi
[I] dustvoice@DustArch ~
$ sudo pacman -S cups cups-pdf nss-mdns
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/brother-mfc-j497dw.git
[I] dustvoice@DustArch ~
$ cd brother-mfc-j497dw
[I] dustvoice@DustArch ~/brother-mfc-j497dw
$ makepkg -si
[I] dustvoice@DustArch ~/brother-mfc-j497dw
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf brother-mfc-j497dw
[I] dustvoice@DustArch ~
$ sudo systemctl enable avahi-daemon.service
[I] dustvoice@DustArch ~
$ sudo systemctl start avahi-daemon.service
```

Now you have to edit `/etc/nsswitch.conf`

so this line

/etc/nsswitch.conf

```
hosts: files mymachines myhostname resolve [!UNAVAIL=return] dns
```

becomes this line

/etc/nsswitch.conf

```
hosts: files mymachines myhostname mdns4_minimal [NOTFOUND=return] resolve
[!UNAVAIL=return] dns
```

Now continue with this

```
[I] dustvoice@DustArch ~
$ avahi-browse --all --ignore-local --resolve --terminate
[I] dustvoice@DustArch ~
$ sudo systemctl enable org.cups.cupsd.service
[I] dustvoice@DustArch ~
$ sudo systemctl start org.cups.cupsd.service
[I] dustvoice@DustArch ~
$ sudo pacman -S system-config-printer print-manager
```

Just open up `system-config-printer` now and configure your printer.

To test if everything is working, you could open up `brave`, then go to **Print** and then try printing.

Process management

The native tool is `top`.

The next evolutionary step would be `htop`, which is an improved version of `top` (like `vi` and `vim` for example)

```
[I] dustvoice@DustArch ~
$ sudo pacman -S htop
```

If you prefer a GUI for that kind of task, install `xfce4-taskmanager`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S xfce4-taskmanager
```

Communication

Life is all about communicating. Here are some pieces of software to do exactly that.

Email

There is nothing better than some classical email.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S thunderbird
```

Telegram

You want to have your `telegram` messages on your desktop PC?

```
[I] dustvoice@DustArch ~
$ sudo pacman -S telegram-desktop
```

TeamSpeak 3

Wanna chat with your gaming friends and they have a `teamspeak3` server? Go for it

```
[I] dustvoice@DustArch ~
$ sudo pacman -S teamspeak3
```

Discord

You’d rather use `discord`? No problem

```
[I] dustvoice@DustArch ~
$ sudo pacman -S discord
```

Video software

Just some additional software related to videos.

Viewing video

You might consider using `vlc`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S vlc
```

Creating video

`obs` should be the right choice

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/obs-studio-git
[I] dustvoice@DustArch ~
$ cd obs-studio-git
[I] dustvoice@DustArch ~/obs-studio-git
$ makepkg -si
[I] dustvoice@DustArch ~/obs-studio-git
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf obs-studio-git
```

Showing keystrokes

In order to show the viewers what keystrokes you're pressing, you can use something like `screenkey`

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/screenkey.git
[I] dustvoice@DustArch ~
$ cd screenkey
[I] dustvoice@DustArch ~/screenkey
$ makepkg -si
[I] dustvoice@DustArch ~/screenkey
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf screenkey
[I] dustvoice@DustArch ~
$ screenkey
```

For ideal use with `obs`, my `dotfiles` repository already provides you with the `screenkey-obs` script for you to run with `fish`.

Live stream a terminal session

For this task, you'll need a program called `tmate`. Just install

```
[I] dustvoice@DustArch ~
$ sudo pacman -S tmate
```

and run it

```
[I] dustvoice@DustArch ~
$ tmate
```

Editing video

In my case, I'm using `davinci-resolve`.

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/davinci-resolve.git
[I] dustvoice@DustArch ~
$ cd davinci-resolve
[I] dustvoice@DustArch ~/davinci-resolve
$ makepkg -si
[I] dustvoice@DustArch ~/davinci-resolve
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf davinci-resolve
```

Utilizing video

Wanna remote control your own or another PC? `teamviewer` might just be the right choice for you

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/teamviewer.git
[I] dustvoice@DustArch ~
$ cd teamviewer
[I] dustvoice@DustArch ~/teamviewer
$ makepkg -si
[I] dustvoice@DustArch ~/teamviewer
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf teamviewer
```

Ardour

To e.g. edit and produce audio, I would recommend `ardour`, because it's easy to use, stable and cross platform.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ardour
```

You might have to edit `/etc/security/limits.conf`, to increase the allowed locked memory amount.

In my case I have 32GB of RAM and I want the `audio` group to be allocate most of the RAM, which is why I added the following line to the file

/etc/security/limits.conf

```
@audio - memlock 29360128
```

Ardour won't natively save in the `mp3` format, due to licensing stuff. In order to create `mp3` files, for sharing with other devices, because they have problems with `wav` files, for example, you can just use `ffmpeg`.

First make sure it's installed

```
[I] dustvoice@DustArch ~
$ sudo pacman -S ffmpeg
```

and after that we're going to convert `in.wav` to `out.mp3`

```
[I] dustvoice@DustArch ~
$ ffmpeg -i in.wav -acodec mp3 out.mp3
```

Virtualization

You might need to run another OS, for example Mac OS, from within Linux, e.g. for development/testing purposes. For that you can use `virtualbox`

```
[I] dustvoice@DustArch ~
$ sudo pacman -S virtualbox virtualbox-host-modules-arch
```

Now when you want to use `virtualbox` just load the kernel module

```
[I] dustvoice@DustArch ~
$ sudo modprobe vboxdrv
```

and add the user which is supposed to run `virtualbox` to the `vboxusers` group

```
[I] dustvoice@DustArch ~
$ sudo usermod -a G vboxusers $USER
```

and if you want to use `rawdisk` functionality, also to the `disk` group

```
[I] dustvoice@DustArch ~
$ sudo usermod -a G disk $USER
```

Now just re-login and you're good to go.

Gaming

The first option for native/emulated gaming on Linux is obviously `steam`.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S steam lib32-nvidia-utils pulseaudio pulseaudio-alsa lib32-libpulse
```

The second option would be `lutris`, a program, that configures a wine instance correctly, etc.

```
[I] dustvoice@DustArch ~
$ sudo pacman -S lutris
```

Wacom

In order to use a Wacom graphics tablet, you'll have to install some packages

```
[I] dustvoice@DustArch ~
$ sudo pacman -S libwacom xf86-input-wacom
```

You could now configure your tablet using the `xsetwacom` command. But on the other hand there is also `wacom-utility`, a GUI software for all of that, so you could try if that works first.

```
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/wacom-utility.git
[I] dustvoice@DustArch ~
$ cd wacom-utility
[I] dustvoice@DustArch ~/wacom-utility
$ git clone https://aur.archlinux.org/gksu.git
[I] dustvoice@DustArch ~/wacom-utility
$ cd gksu
[I] dustvoice@DustArch ~/wacom-utility/gksu
$ git clone https://aur.archlinux.org/libgks.git
[I] dustvoice@DustArch ~/wacom-utility/gksu
$ cd libgks
[I] dustvoice@DustArch ~/wacom-utility/gksu/libgks
$ makepkg -si
[I] dustvoice@DustArch ~/wacom-utility/gksu/libgks
$ cd ..
[I] dustvoice@DustArch ~/wacom-utility/gksu
$ makepkg -si
[I] dustvoice@DustArch ~/wacom-utility/gksu
$ cd ..
[I] dustvoice@DustArch ~/wacom-utility
$ makepkg -si
[I] dustvoice@DustArch ~/wacom-utility
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf wacom-utility
```

Upgrading the system

You're probably wondering why this gets a dedicated section.

You'll probably think that it would be just a matter of issuing

```
[I] dustvoice@DustArch ~
$ sudo pacman -Syu
```

That's both true and false.

You have to make sure, **that your boot partition is mounted at `/boot`** in order for everything to upgrade correctly. That's because the moment you upgrade the `linux` package without having the correct partition mounted at `/boot`, your system won't boot. You also might have to do `grub-mkconfig -o /boot/grub/grub.cfg` after you install a different kernel image.

If your system **indeed doesn't boot** and **boots to a recovery console**, then double check that the issue really is the not perfectly executed kernel update by issuing

```
[I] root@DustArch ~
$ uname -a
```

and

```
[I] root@DustArch ~
$ pacman -Q linux
```

The version of these two packages should be exactly the same!

If it isn't there is an easy fix for it.

Fixing a faulty kernel upgrade

First off we need to restore the old `linux` package.

For that note the version number of

```
[I] root@DustArch ~
$ uname -a
```

Now we'll make sure first that nothing is mounted at `/boot`, because the process will likely create some unwanted files. The process will also create a new `/boot` folder, which we're going to delete afterwards.

```
[I] root@DustArch ~
$ umount /boot
```

Now `cd` into `pacman`'s package cache

```
[I] root@DustArch ~
$ cd /var/cache/pacman/pkg
```

There should be a file located named something like `linux-<version>.pkg.tar.xz`, where `<version>` would be somewhat equivalent to the previously noted version number

Now downgrade the `linux` package

```
[I] root@DustArch ~
$ pacman -U linux-<version>.pkg.tar.xz
```

After that remove the possibly created `/boot` directory

```
[I] root@DustArch ~
$ rm -rf /boot
[I] root@DustArch ~
$ mkdir /boot
```

Now reboot and `mount` the `boot` partition, in my case an `EFI System Partition`.

Now simply rerun

```
[I] dustvoice@DustArch ~
$ sudo pacman -Syu
```

and you should be fine now.

> Consider setting up a `fstab` entry for the `boot` partition, in order to avoid such dilemma in the future.
>
> See `fstab` for more.

DustArch package list

A complete list of all the packages present on my full fledged system at the time of writing

.packages-x86_64

```
a52dec
acl
adobe-source-code-pro-fonts
adwaita-icon-theme
alsa-lib
alsa-plugins
alsa-topology-conf
alsa-ucm-conf
alsa-utils
amd-ucode
android-tools
aom
apr
apr-util
arandr
arch-install-scripts
archlinux-keyring
ardour
argon2
aribb24
aspell
astyle
at-spi2-atk
at-spi2-core
atk
atkmm
attica
attr
aubio
audit
autoconf
autoconf-archive
automake
avahi
b43-fwcutter
base
bash
bbswitch
bc
bind-tools
binutils
bison
blueman
bluez
```

```
bluez-libs
bluez-utils
boost-libs
brave-bin
breeze-grub
broadcom-wl
brother-mfc-j497dw
brotli
browserpass
btrfs-progs
bubblewrap
bzip2
bzr
c-ares
ca-certificates
ca-certificates-mozilla
ca-certificates-utils
cabextract
cadence
cairo
cairomm
cantarell-fonts
caps2esc
ccid
cdparanoia
celt
celt0.5.1
clang
clonezilla
clucene
cmake
colord
compiler-rt
confuse
coreutils
cpio
cracklib
crda
cryptsetup
cups
cups-filters
cups-pdf
curl
darkhttpd
dav1d
davinci-resolve
db
dbus
dbus-glib
dconf
ddrescue
```

```
desktop-file-utils
device-mapper
devtools
dhclient
dhcpcd
dialog
diffutils
ding-libs
discord
djvulibre
dmraid
dnsmasq
dnssec-anchors
docbook-xml
docbook-xsl
dosfstools
double-conversion
doxygen
drbl
e2fsprogs
ecryptfs-utils
efibootmgr
efitools
efivar
egl-wayland
eglexternalplatform
elinks
enchant
ethtool
evince
exfat-utils
exiv2
exo
expat
f2fs-tools
faad2
fakeroot
fbida
ffmpeg
fftw
file
filesystem
findutils
fish
flac
flex
fontconfig
freeglut
freetype2
fribidi
fsarchiver
```

```
fuse-common
fuse2
fuse3
gawk
gc
gcc
gcc-libs
gconf
gcr
gd
gdbm
gdk-pixbuf2
geoip
geoip-database
gettext
ghostscript
giblib
giflib
gigolo
git
gksu
glew
glib-networking
glib2
glib2-docs
glibc
glibmm
glu
gmp
gnome-common
gnome-desktop
gnu-free-fonts
gnu-netcat
gnupg
gnutls
gobject-introspection-runtime
gparted
gperf
gpgme
gpm
gptfdisk
graphene
graphite
graphviz
grep
grml-zsh-config
groff
grub
grub-theme-vimix
gsettings-desktop-schemas
gsfonts
```

```
gsl
gsm
gspell
gssproxy
gst-plugins-base
gst-plugins-base-libs
gstreamer
gtk-doc
gtk-update-icon-cache
gtk2
gtk3
gtkdialog
gtkmm
gtkmm3
gts
guile
gvfs
gvfs-mtp
gzip
harfbuzz
harfbuzz-icu
haveged
hdparm
hicolor-icon-theme
hspell
htop
hunspell
hwids
hyphen
i3-gaps
i3blocks
i3lock
i3status
iana-etc
ibus
icu
ijs
imagemagick
imlib2
inetutils
intel-ucode
interception-tools
intltool
iproute2
iptables
iputils
ipw2100-fw
ipw2200-fw
irssi
iso-codes
itstool
```

```
iw
iwd
jack2
jansson
jasper
java-environment-common
java-runtime-common
jbig2dec
jdk-openjdk
jfsutils
jq
jre-openjdk
jre-openjdk-headless
js60
json-c
json-glib
jsoncpp
kactivities
karchive
kauth
kbd
kbookmarks
kcmutils
kcodecs
kcompletion
kconfig
kconfigwidgets
kcoreaddons
kcrash
kdbusaddons
kdeclarative
keyutils
kglobalaccel
kguiaddons
ki18n
kiconthemes
kio
kirigami2
kitemmodels
kitemviews
kjobwidgets
kmod
knotifications
kpackage
krb5
krita
kservice
ktextwidgets
kwallet
kwayland
kwidgetsaddons
```

```
kwindowsystem
kxmlgui
l-smash
lame
lbzip2
lcms2
ldns
less
lftp
lib32-acl
lib32-alsa-lib
lib32-alsa-plugins
lib32-atk
lib32-attr
lib32-bzip2
lib32-cairo
lib32-dbus
lib32-e2fsprogs
lib32-expat
lib32-flac
lib32-fontconfig
lib32-freetype2
lib32-fribidi
lib32-gcc-libs
lib32-gdk-pixbuf2
lib32-gettext
lib32-glib2
lib32-glibc
lib32-glu
lib32-gmp
lib32-gnutls
lib32-gtk2
lib32-harfbuzz
lib32-icu
lib32-keyutils
lib32-krb5
lib32-lcms2
lib32-libappindicator-gtk2
lib32-libasyncns
lib32-libcap
lib32-libcups
lib32-libdatrie
lib32-libdbusmenu-glib
lib32-libdbusmenu-gtk2
lib32-libdrm
lib32-libelf
lib32-libffi
lib32-libgcrypt
lib32-libglvnd
lib32-libgpg-error
lib32-libice
```

```
lib32-libidn
lib32-libindicator-gtk2
lib32-libjpeg-turbo
lib32-libldap
lib32-libnl
lib32-libogg
lib32-libpcap
lib32-libpciaccess
lib32-libpng
lib32-libpng12
lib32-libpulse
lib32-libsm
lib32-libsndfile
lib32-libtasn1
lib32-libthai
lib32-libtiff
lib32-libusb
lib32-libvorbis
lib32-libx11
lib32-libxau
lib32-libxcb
lib32-libxcomposite
lib32-libxcursor
lib32-libxdamage
lib32-libxdmcp
lib32-libxext
lib32-libxfixes
lib32-libxft
lib32-libxi
lib32-libxinerama
lib32-libxml2
lib32-libxrandr
lib32-libxrender
lib32-libxshmfence
lib32-libxss
lib32-libxtst
lib32-libxxf86vm
lib32-llvm-libs
lib32-lm_sensors
lib32-lz4
lib32-mesa
lib32-ncurses
lib32-nettle
lib32-nspr
lib32-nss
lib32-nvidia-utils
lib32-openssl
lib32-p11-kit
lib32-pango
lib32-pcre
lib32-pixman
```

```
lib32-readline
lib32-sqlite
lib32-systemd
lib32-util-linux
lib32-wayland
lib32-xz
lib32-zlib
lib32-zstd
libabw
libaio
libao
libarchive
libass
libassuan
libasyncns
libatasmart
libatomic_ops
libavc1394
libblockdev
libbluray
libbytesize
libcanberra
libcanberra-pulse
libcap
libcap-ng
libcddb
libcdio
libcdio-paranoia
libcdr
libcmis
libconfig
libcroco
libcups
libcurl-gnutls
libdaemon
libdatrie
libdbusmenu-glib
libdbusmenu-gtk2
libdbusmenu-qt5
libdca
libdrm
libdvbpsi
libe-book
libebml
libedit
libelf
libepoxy
libepubgen
libetonyek
libev
libevdev
```

```
libevent
libexif
libexttextcat
libfdk-aac
libffi
libfontenc
libfreehand
libgcrypt
libgksu
libglade
libglvnd
libgnome-keyring
libgpg-error
libgssglue
libgtop
libgudev
libgusb
libgxps
libibus
libical
libice
libid3tag
libidn
libidn2
libiec61883
libimagequant
libimobiledevice
libindicator-gtk2
libinput
libixion
libjpeg-turbo
libksba
liblangtag
libldap
liblo
liblouis
liblqr
liblrdf
libluv
libmad
libmatroska
libmaxminddb
libmm-glib
libmng
libmnl
libmodplug
libmpc
libmpcdec
libmpeg2
libmspub
libmtp
```

```
libmwaw
libndp
libnet
libnetctlgui
libnetfilter_conntrack
libnewt
libnfnetlink
libnftnl
libnghttp2
libnl
libnm
libnma
libnotify
libnsl
libnumbertext
libodfgen
libogg
libomxil-bellagio
liborcus
libotr
libpagemaker
libpaper
libpcap
libpciaccess
libpgm
libpipeline
libplist
libpng
libpng12
libproxy
libpsl
libpulse
libqxp
libraqm
libraw
libraw1394
libreoffice-fresh
librevenge
librsvg
libsamplerate
libsasl
libseccomp
libsecret
libsigc++
libsm
libsndfile
libsodium
libsoup
libsoxr
libspectre
libssh
```

```
libssh2
libstaroffice
libsynctex
libtar
libtasn1
libteam
libtermkey
libthai
libtheora
libtiff
libtirpc
libtommath
libtool
libuiohook
libunistring
libunwind
libupnp
libusb
libusb-compat
libusbmuxd
libutempter
libutf8proc
libutil-linux
libuv
libva
libvdpau
libvisio
libvisual
libvoikko
libvorbis
libvpx
libvterm
libwacom
libwebp
libwnck3
libwpd
libwpe
libwpg
libwps
libx11
libxau
libxaw
libxcb
libxcomposite
libxcursor
libxdamage
libxdg-basedir
libxdmcp
libxext
libxfce4ui
libxfce4util
```

```
libxfixes
libxfont2
libxft
libxi
libxinerama
libxkbcommon
libxkbcommon-x11
libxkbfile
libxml2
libxmu
libxnvctrl
libxp
libxpm
libxrandr
libxrender
libxres
libxshmfence
libxslt
libxss
libxt
libxtst
libxv
libxvmc
libxxf86vm
libyaml
libzmf
licenses
lilv
linux
linux-api-headers
linux-atm
linux-firmware
linux-rt-docs
linux-rt-headers
llvm-libs
lm_sensors
lmdb
lpsolve
lrzip
lsb-release
lsof
lsscsi
lua
lua51
luajit
lutris
lvm2
lz4
lzo
lzop
m4
```

```
mailcap
make
mallard-ducktype
man-db
man-pages
mc
md4c
mdadm
media-player-info
memtest86+
memtest86-efi
mercurial
mesa
mesa-demos
minecraft-launcher
minizip
mkinitcpio
mkinitcpio-busybox
mkinitcpio-nfs-utils
mobile-broadband-provider-info
mozilla-common
mpfr
msgpack-c
mtdev
mtools
mupdf
nano
nbd
ncurses
ndctl
ndisc6
neon
neovim
net-tools
netctl
netctl-tray
netctlgui-helper
netpbm
nettle
network-manager-applet
networkmanager
networkmanager-openconnect
networkmanager-openvpn
nfs-utils
nfsidmap
nilfs-utils
ninja
nitrogen
nm-connection-editor
nmap
node-gyp
```

```
nodejs
npm
npth
nspr
nss
nss-mdns
ntfs-3g
ntp
nvidia
nvidia-settings
nvidia-utils
nvidia-xrun
obs-input-overlay-bin
obs-linuxbrowser-bin
obs-studio-git
ocl-icd
oniguruma
openal
opencl-nvidia
openconnect
opencore-amr
openexr
openjpeg2
openmotif
openresolv
opensc
openssh
openssl
openssl-1.0
openvpn
opera-ffmpeg-codecs
opus
opusfile
orc
os-prober
p11-kit
p7zip
pacman
pacman-mirrorlist
pam
pambase
pango
pangomm
partclone
parted
partimage
pass
patch
pavucontrol
pbzip2
pciutils
```

```
pcre
pcre2
pcsclite
pepper-flash
perl
perl-data-dump
perl-encode-locale
perl-error
perl-file-listing
perl-html-parser
perl-html-tagset
perl-http-cookies
perl-http-daemon
perl-http-date
perl-http-message
perl-http-negotiate
perl-io-html
perl-io-socket-ssl
perl-json
perl-libwww
perl-lwp-mediatypes
perl-lwp-protocol-https
perl-mailtools
perl-net-http
perl-net-ssleay
perl-timedate
perl-try-tiny
perl-uri
perl-www-robotrules
perl-xml-parser
picom
pigz
pinentry
pixman
pixz
pkcs11-helper
pkgconf
plasma-framework
plasma5-applet-netctl-gui
polkit
polkit-qt5
poppler
poppler-glib
popt
postgresql
postgresql-libs
ppp
pptpclient
print-manager
procps-ng
progsreiserfs
```

```
protobuf
psmisc
pulseaudio
pulseaudio-alsa
pulseaudio-bluetooth
pulseaudio-jack
pulsemixer
pygobject-devel
pygtk
python
python-anytree
python-appdirs
python-babel
python-beaker
python-cachecontrol
python-cairo
python-chardet
python-colorama
python-dbus
python-dbus-common
python-distlib
python-distro
python-distutils-extra
python-docutils
python-evdev
python-gobject
python-html5lib
python-idna
python-imagesize
python-importlib-metadata
python-jinja
python-lockfile
python-lxml
python-mako
python-markdown
python-markupsafe
python-more-itertools
python-msgpack
python-ordered-set
python-packaging
python-pep517
python-pillow
python-pip
python-ply
python-progress
python-pycups
python-pycurl
python-pygments
python-pyparsing
python-pyqt5
python-pytoml
```

```
python-pytz
python-requests
python-retrying
python-setuptools
python-six
python-snowballstemmer
python-sphinx-alabaster-theme
python-sphinxcontrib-applehelp
python-sphinxcontrib-devhelp
python-sphinxcontrib-htmlhelp
python-sphinxcontrib-jsmath
python-sphinxcontrib-qthelp
python-sphinxcontrib-serializinghtml
python-urllib3
python-webencodings
python-yaml
python-zipp
python2
python2-appdirs
python2-cairo
python2-dbus
python2-gobject
python2-gobject2
python2-ordered-set
python2-packaging
python2-pyparsing
python2-pytz
python2-setuptools
python2-six
qpdf
qt5-base
qt5-declarative
qt5-graphicaleffects
qt5-imageformats
qt5-location
qt5-multimedia
qt5-quickcontrols
qt5-quickcontrols2
qt5-sensors
qt5-speech
qt5-svg
qt5-tools
qt5-webchannel
qt5-webengine
qt5-webkit
qt5-websockets
qt5-x11extras
quazip
raptor
rasqal
rdesktop
```

```
re2
readline
reaper-bin
redland
refind-efi
reiserfsprogs
rest
rhash
ristretto
rofi
rp-pppoe
rpcbind
rsync
rtkit
rubberband
ruby
rubygems
run-parts
sbc
scrcpy
screen
screenkey
scrot
sdl
sdl2
sdparm
sed
semver
serd
serf
sg3_utils
shadow
shared-mime-info
simple-mtpfs
slang
slop
smartmontools
snappy
solid
sonnet
sord
sound-theme-freedesktop
sox
speex
speexdsp
spice
sqlite
sratom
sshfs
startup-notification
steam
```

```
stoken
subversion
sudo
suil
swig
sysfsutils
syslinux
system-config-printer
systemd
systemd-libs
systemd-sysvcompat
t1lib
taglib
tar
tcl
tcpdump
tdb
teamspeak3
teamviewer
telegram-desktop
testdisk
texinfo
thin-provisioning-tools
thunar
thunderbird
tidy
tmate
tre
tree
tslib
ttf-hack
tumbler
twolame
tzdata
udisks2
unibilium
unrar
unzip
upower
usb_modeswitch
usbmuxd
usbutils
util-linux
v4l-utils
vamp-plugin-sdk
vi
vid.stab
vim
vim-runtime
virtualbox
virtualbox-host-modules-arch
```

```
vlc
volume_key
vpnc
vte-common
vte-legacy
vte3
vulkan-icd-loader
wacom-utility
wavpack
wayland
wayland-protocols
webkit2gtk
webrtc-audio-processing
wget
which
wine-staging
wireless-regdb
wireless_tools
woff2
wpa_supplicant
wpebackend-fdo
wvdial
wvstreams
x264
x265
xarchiver
xcb-proto
xcb-util
xcb-util-cursor
xcb-util-image
xcb-util-keysyms
xcb-util-renderutil
xcb-util-wm
xcb-util-xrm
xclip
xdg-dbus-proxy
xdg-utils
xf86-input-evdev
xf86-input-keyboard
xf86-input-libinput
xf86-input-mouse
xf86-input-synaptics
xf86-input-vmmouse
xf86-input-void
xf86-input-wacom
xf86-video-amdgpu
xf86-video-ati
xf86-video-dummy
xf86-video-fbdev
xf86-video-intel
xf86-video-nouveau
```

```
xf86-video-openchrome
xf86-video-qxl
xf86-video-vesa
xf86-video-vmware
xf86-video-voodoo
xfce4-taskmanager
xfce4-terminal
xfconf
xfsprogs
xkblayout-state
xkeyboard-config
xl2tpd
xmlsec
xorg-bdftopcf
xorg-docs
xorg-font-util
xorg-font-utils
xorg-fonts-100dpi
xorg-fonts-75dpi
xorg-fonts-alias
xorg-fonts-encodings
xorg-iceauth
xorg-luit
xorg-mkfontscale
xorg-server
xorg-server-common
xorg-server-devel
xorg-server-xdmx
xorg-server-xephyr
xorg-server-xnest
xorg-server-xvfb
xorg-server-xwayland
xorg-sessreg
xorg-setxkbmap
xorg-smproxy
xorg-util-macros
xorg-x11perf
xorg-xauth
xorg-xbacklight
xorg-xcmsdb
xorg-xcursorgen
xorg-xdpyinfo
xorg-xdriinfo
xorg-xev
xorg-xgamma
xorg-xhost
xorg-xinit
xorg-xinput
xorg-xkbcomp
xorg-xkbevd
xorg-xkbutils
```

```
xorg-xkill
xorg-xlsatoms
xorg-xlsclients
xorg-xmodmap
xorg-xpr
xorg-xprop
xorg-xrandr
xorg-xrdb
xorg-xrefresh
xorg-xset
xorg-xsetroot
xorg-xvinfo
xorg-xwd
xorg-xwininfo
xorg-xwud
xorgproto
xplc
xvidcore
xz
yajl
yaml-cpp
yelp-tools
yelp-xsl
youtube-viewer
zenity
zeromq
zip
zlib
zsh
zstd
```