

DustArch
*DustVoice's Arch Linux from
scratch*

David Holland

Version 4.3, 2019-12-28

Table Of Contents

Inside the archiso	1
Sync up pacman	1
Formatting the drive	1
Preparing the chroot environment	3
Entering the chroot	5
Installing additional packages	5
Master of time	6
Master of locales	7
Naming your machine	8
hostname	8
hosts	8
User setup	9
Give root a password	9
Create a personal user	9
Preparing to boot	11
BIOS	11
UEFI	12
grub config	13
Adjust the timeout	13
Enable the recovery	14
NVIDIA fix	15
Add power options	15
Installing memtest	15
Generating the config	17
Inside the DustArch	18

Someone there?	18
Update and upgrade	20
Enabling the multilib repository	20
Setting the correct shell	20
Version control	21
Security is important	21
Smartcard shenanigans	21
Additional required tools	22
Setting up a home environment	22
Use dotfiles for a base config	23
Set up gpg	23
Finalize the dotfiles	24
gpg-agent forwarding	25
JUCE and FRUT	27
Back to your roots	27
Password management	28
python	28
ruby & asciidoctor	29
Using JUCE	30
Additional development tools	30
Code formatting	31
Documentation	31
Build tools	31
fstab	31
Audio	33
alsa	33
pulseaudio	34

jack	35
Audio handling	35
Bluetooth	36
Graphical desktop environment	40
NVIDIA	40
Launching the graphical environment	41
The NVIDIA way	41
GUI Software	45
Desktop background	45
Compositing software	46
networkmanager applet	46
Keyboard	47
X clipboard	48
Taking screen shots	48
Image viewer	49
File manager	49
Android file transfer	50
Archive manager	53
Partition management	54
PDF viewer	54
Using the GUI	54
Using the framebuffer	55
Web browser	55
Entering the dark side	56
Office utilities	57
Printing	57
Process management	59
Communication	60

Email	60
Telegram	60
TeamSpeak 3.....	60
Discord	61
Video software	61
Viewing video.....	61
Creating video	61
Live stream a terminal session	63
Editing video.....	63
Utilizing video	64
Ardour	64
Virtualization	66
Gaming	67
Wacom	68
Upgrading the system	70
Fixing a faulty kernel upgrade	71
DustArch package list	73

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 **chrooted** 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 command** now

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




It should be 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 `nmcli` 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 **pinging** 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 ~> sudo 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 ~> sudo pacman -S make cmake clang  
jdk-openjdk python python-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 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 **password-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
$ sudo 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 asciidoctor-pdf asciidoctor-  
epub3 asciidoctor-latex 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

```
1 UUID=e26de048-6147-42e5-a34b-59f1a50621bb      /
  ext4          rw,relatime                        0 1
2
3 UUID="C8E3-A0FD"
  /boot         vfat                                defaults
  0 1
4
5 UUID="DC88-5A4E"
  /mnt/projects exfat                              rw,relatime
  0 0
6
7 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 sox 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-utils** 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 can 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 ~  
$ pulseaudio --kill  
[I] dustvoice@DustArch ~  
$ pulseaudio --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 `ttys`.

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

```
1 blacklist nvidia
2 blacklist nvidia-drm
3 blacklist nvidia-modeset
4 blacklist nvidia-uvm
5 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 gvfs
[I] dustvoice@DustArch ~
$ git clone https://aur.archlinux.org/gigolo.git
[I] dustvoice@DustArch ~
$ cd gigolo
[I] dustvoice@DustArch ~/gigolo
$ makepkg -si
[I] dustvoice@DustArch ~/gigolo
$ cd ..
[I] dustvoice@DustArch ~
$ rm -rf gigolo
```

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



You might have to check out how to import the **gpg** keys on the **AUR** page of **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 corresponding 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 ~  
$ 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

```
1 a52dec
2 acl
3 adobe-source-code-pro-fonts
4 adwaita-icon-theme
5 alsa-lib
6 alsa-plugins
7 alsa-topology-conf
8 alsa-ucm-conf
9 alsa-utils
10 amd-ucode
11 android-tools
12 aom
13 apr
14 apr-util
15 arandr
16 arch-install-scripts
17 archlinux-keyring
18 ardour
19 argon2
20 aribb24
21 aspell
22 astyle
23 at-spi2-atk
24 at-spi2-core
25 atk
26 atkmm
27 attica
```

```
28 attr
29 aubio
30 audit
31 autoconf
32 autoconf-archive
33 automake
34 avahi
35 b43-fwcutter
36 base
37 bash
38 bbswitch
39 bc
40 bind-tools
41 binutils
42 bison
43 blueman
44 bluez
45 bluez-libs
46 bluez-utils
47 boost-libs
48 brave-bin
49 breeze-grub
50 broadcom-wl
51 brother-mfc-j497dw
52 brotli
53 browserpass
54 btrfs-progs
55 bubblewrap
56 bzip2
57 bzip
58 c-ares
59 ca-certificates
60 ca-certificates-mozilla
61 ca-certificates-utils
```

62 cabextract
63 cadence
64 cairo
65 cairomm
66 cantarell-fonts
67 caps2esc
68 ccid
69 cdparanoia
70 celt
71 celt0.5.1
72 clang
73 clonezilla
74 clucene
75 cmake
76 colord
77 compiler-rt
78 confuse
79 coreutils
80 cpio
81 cracklib
82 crda
83 cryptsetup
84 cups
85 cups-filters
86 cups-pdf
87 curl
88 darkhttpd
89 dav1d
90 davinci-resolve
91 db
92 dbus
93 dbus-glib
94 dconf
95 ddrescue

96 desktop-file-utils
97 device-mapper
98 devtools
99 dhclient
100 dhcpcd
101 dialog
102 diffutils
103 ding-lib
104 discord
105 djvulibre
106 dmraid
107 dnsmasq
108 dnssec-anchors
109 docbook-xml
110 docbook-xsl
111 dosfstools
112 double-conversion
113 doxygen
114 drbl
115 e2fsprogs
116 ecryptfs-utils
117 efibootmgr
118 efityls
119 efivar
120 egl-wayland
121 eglxexternalplatform
122 elinks
123 enchant
124 ethtool
125 evince
126 exfat-utils
127 exiv2
128 exo
129 expat

```
130 f2fs-tools
131 faad2
132 fakeroot
133 fbida
134 ffmpeg
135 fftw
136 file
137 filesystem
138 findutils
139 fish
140 flac
141 flex
142 fontconfig
143 freeglut
144 freetype2
145 fribidi
146 fsarchiver
147 fuse-common
148 fuse2
149 fuse3
150 gawk
151 gc
152 gcc
153 gcc-libs
154 gconf
155 gcr
156 gd
157 gdbm
158 gdk-pixbuf2
159 geoip
160 geoip-database
161 gettext
162 ghostscript
163 gliblib
```

164 giflib
165 gigolo
166 git
167 gksu
168 glew
169 glib-networking
170 glib2
171 glib2-docs
172 glibc
173 glibmm
174 glu
175 gmp
176 gnome-common
177 gnome-desktop
178 gnu-free-fonts
179 gnu-netcat
180 gnupg
181 gnutls
182 gobject-introspection-runtime
183 gparted
184 gperf
185 gpgme
186 gpm
187 gptfdisk
188 graphene
189 graphite
190 graphviz
191 grep
192 grml-zsh-config
193 groff
194 grub
195 grub-theme-vimix
196 gsettings-desktop-schemas
197 gsfonts

198 gsl
199 gsm
200 gspell
201 gssproxy
202 gst-plugins-base
203 gst-plugins-base-libs
204 gstreamer
205 gtk-doc
206 gtk-update-icon-cache
207 gtk2
208 gtk3
209 gtkdialog
210 gtkmm
211 gtkmm3
212 gts
213 guile
214 gvfs
215 gvfs-mtp
216 gzip
217 harfbuzz
218 harfbuzz-icu
219 haveged
220 hdparm
221 hicolor-icon-theme
222 hspell
223 htop
224 hunspell
225 hwids
226 hyphen
227 i3-gaps
228 i3blocks
229 i3lock
230 i3status
231 iana-etc

```
232 ibus
233 icu
234 ijs
235 imagemagick
236 imlib2
237 inetutils
238 intel-ucode
239 interception-tools
240 intltool
241 iproute2
242 iptables
243 iputils
244 ipw2100-fw
245 ipw2200-fw
246 irssi
247 iso-codes
248 itstool
249 iw
250 iwd
251 jack2
252 jansson
253 jasper
254 java-environment-common
255 java-runtime-common
256 jbig2dec
257 jdk-openjdk
258 jfsutils
259 jq
260 jre-openjdk
261 jre-openjdk-headless
262 js60
263 json-c
264 json-glib
265 jsoncpp
```

266 kactivities
267 karchive
268 kauth
269 kbd
270 kbookmarks
271 kcmutils
272 kcodecs
273 kcompletion
274 kconfig
275 kconfigwidgets
276 kcoreaddons
277 kcrash
278 kdbusaddons
279 kdeclarative
280 keyutils
281 kglobalaccel
282 kguiaddons
283 ki18n
284 kiconthemes
285 kio
286 kirigami2
287 kitemmodels
288 kitemviews
289 kjobwidgets
290 kmod
291 knotifications
292 kpackage
293 krb5
294 krita
295 kservice
296 ktextwidgets
297 kwallet
298 kwayland
299 kwidgetsaddons

```
300 kwindowssystem
301 kxmlgui
302 l-smash
303 lame
304 lbzip2
305 lcms2
306 ldns
307 less
308 lftp
309 lib32-acl
310 lib32-alsa-lib
311 lib32-alsa-plugins
312 lib32-atk
313 lib32-attr
314 lib32-bzip2
315 lib32-cairo
316 lib32-dbus
317 lib32-e2fsprogs
318 lib32-expat
319 lib32-flac
320 lib32-fontconfig
321 lib32-freetype2
322 lib32-fribidi
323 lib32-gcc-libs
324 lib32-gdk-pixbuf2
325 lib32-gettext
326 lib32-glib2
327 lib32-glibc
328 lib32-glu
329 lib32-gmp
330 lib32-gnutls
331 lib32-gtk2
332 lib32-harfbuzz
333 lib32-icu
```

334 lib32-keyutils
335 lib32-krb5
336 lib32-lcms2
337 lib32-libappindicator-gtk2
338 lib32-libasyncns
339 lib32-libcap
340 lib32-libcups
341 lib32-libdatrie
342 lib32-libdbusmenu-glib
343 lib32-libdbusmenu-gtk2
344 lib32-libdrm
345 lib32-libelf
346 lib32-libffi
347 lib32-libgcrypt
348 lib32-libglvnd
349 lib32-libgpg-error
350 lib32-libice
351 lib32-libidn
352 lib32-libindicator-gtk2
353 lib32-libjpeg-turbo
354 lib32-libldap
355 lib32-libnl
356 lib32-libogg
357 lib32-libpcap
358 lib32-libpciaccess
359 lib32-libpng
360 lib32-libpng12
361 lib32-libpulse
362 lib32-libsm
363 lib32-libsndfile
364 lib32-libtasn1
365 lib32-libthai
366 lib32-libtiff
367 lib32-libusb

368 lib32-libvorbis
369 lib32-libx11
370 lib32-libxau
371 lib32-libxcb
372 lib32-libxcomposite
373 lib32-libxcursor
374 lib32-libxdamage
375 lib32-libxdmcp
376 lib32-libxext
377 lib32-libxfixes
378 lib32-libxft
379 lib32-libxi
380 lib32-libxinerama
381 lib32-libxml2
382 lib32-libxrandr
383 lib32-libxrender
384 lib32-libxshmfence
385 lib32-libxss
386 lib32-libxtst
387 lib32-libxxf86vm
388 lib32-llvm-libs
389 lib32-lm_sensors
390 lib32-lz4
391 lib32-mesa
392 lib32-ncurses
393 lib32-nettle
394 lib32-nspr
395 lib32-nss
396 lib32-nvidia-utils
397 lib32-openssl
398 lib32-p11-kit
399 lib32-pango
400 lib32-pcre
401 lib32-pixman

402 lib32-readline
403 lib32-sqlite
404 lib32-systemd
405 lib32-util-linux
406 lib32-wayland
407 lib32-xz
408 lib32-zlib
409 lib32-zstd
410 libabw
411 libaio
412 libao
413 libarchive
414 libass
415 libassuan
416 libasyncns
417 libatasmart
418 libatomic_ops
419 libavc1394
420 libblockdev
421 libbluray
422 libbytesize
423 libcanberra
424 libcanberra-pulse
425 libcap
426 libcap-ng
427 libcddb
428 libcdio
429 libcdio-paranoia
430 libcdr
431 libcmis
432 libconfig
433 libcroco
434 libcups
435 libcurl-gnutls

436 libdaemon
437 libdatrie
438 libdbusmenu-glib
439 libdbusmenu-gtk2
440 libdbusmenu-qt5
441 libdca
442 libdrm
443 libdvbpsi
444 libe-book
445 libebml
446 libedit
447 libelf
448 libepoxy
449 libepubgen
450 libetonyek
451 libev
452 libevdev
453 libevent
454 libexif
455 libexttextcat
456 libfdk-aac
457 libffi
458 libfontenc
459 libfreehand
460 libgcrypt
461 libgksu
462 libglade
463 libglvnd
464 libgnome-keyring
465 libgpg-error
466 libgssglue
467 libgtop
468 libgudev
469 libgusb

470 libgxps
471 libibus
472 libical
473 libice
474 libid3tag
475 libidn
476 libidn2
477 libiec61883
478 libimagequant
479 libimobiledevice
480 libindicator-gtk2
481 libinput
482 libixion
483 libjpeg-turbo
484 libksba
485 liblangtag
486 libldap
487 liblo
488 liblouis
489 liblqr
490 liblrdf
491 libluv
492 libmad
493 libmatroska
494 libmaxminddb
495 libmm-glib
496 libmng
497 libmnl
498 libmodplug
499 libmpc
500 libmpcdec
501 libmpeg2
502 libmtp
503 libmtp

504 libmwaw
505 libndp
506 libnet
507 libnetctlgui
508 libnetfilter_conntrack
509 libnewt
510 libnfnetlink
511 libnftnl
512 libnghttp2
513 libnl
514 libnm
515 libnma
516 libnotify
517 libnsl
518 libnumbertext
519 libodfgen
520 libogg
521 libomxil-bellagio
522 liborcus
523 libotr
524 libpagemaker
525 libpaper
526 libpcap
527 libpciaccess
528 libpgm
529 libpipeline
530 libplist
531 libpng
532 libpng12
533 libproxy
534 libpsl
535 libpulse
536 libqxp
537 libraqm

538 libraw
539 libraw1394
540 libreoffice-fresh
541 librevenge
542 librsvg
543 libsamplerate
544 libsasldb
545 libseccomp
546 libsecret
547 libsigc++
548 libsm
549 libsndfile
550 libsodium
551 libsoup
552 libsoxr
553 libspectre
554 libssh
555 libssh2
556 libstaroffice
557 libsyntax
558 libtar
559 libtasn1
560 libteam
561 libtermkey
562 libthai
563 libtheora
564 libtiff
565 libtirpc
566 libtommath
567 libtool
568 libuihook
569 libunistring
570 libunwind
571 libupnp

572 libusb
573 libusb-compat
574 libusbmuxd
575 libutempter
576 libutf8proc
577 libutil-linux
578 libuv
579 libva
580 libvdpau
581 libvisio
582 libvisual
583 libvoikko
584 libvorbis
585 libvpx
586 libvterm
587 libwacom
588 libwebp
589 libwnck3
590 libwpd
591 libwpe
592 libwpg
593 libwps
594 libx11
595 libxau
596 libxaw
597 libxcb
598 libxcomposite
599 libxcursor
600 libxdamage
601 libxdg-basedir
602 libxdmcp
603 libxext
604 libxfce4ui
605 libxfce4util

```
606 libxfixes
607 libxfont2
608 libxft
609 libxi
610 libxinerama
611 libxkbcommon
612 libxkbcommon-x11
613 libxkbfile
614 libxml2
615 libxmu
616 libxnvctrl
617 libxp
618 libxpm
619 libxrandr
620 libxrender
621 libxres
622 libxshmfence
623 libxslt
624 libxss
625 libxt
626 libxtst
627 libxv
628 libxvmc
629 libxxf86vm
630 libyaml
631 libzmf
632 licenses
633 lilv
634 linux
635 linux-api-headers
636 linux-atm
637 linux-firmware
638 linux-rt-docs
639 linux-rt-headers
```



```
640 llvm-libs
641 lm_sensors
642 lmdb
643 lpsolve
644 lrzip
645 lsb-release
646 lsof
647 lsscsi
648 lua
649 lua51
650 luajit
651 lutris
652 lvm2
653 lz4
654 lzo
655 lzop
656 m4
657 mailcap
658 make
659 mallard-ducktype
660 man-db
661 man-pages
662 mc
663 md4c
664 mdadm
665 media-player-info
666 memtest86+
667 memtest86-efi
668 mercurial
669 mesa
670 mesa-demos
671 minecraft-launcher
672 minizip
673 mkinitcpio
```

674 mkinitcpio-busybox
675 mkinitcpio-nfs-utils
676 mobile-broadband-provider-info
677 mozilla-common
678 mpfr
679 msgpack-c
680 mtdev
681 mtools
682 mupdf
683 nano
684 nbd
685 ncurses
686 ndctl
687 ndisc6
688 neon
689 neovim
690 net-tools
691 netctl
692 netctl-tray
693 netctlgui-helper
694 netpbm
695 nettle
696 network-manager-applet
697 networkmanager
698 networkmanager-openconnect
699 networkmanager-openvpn
700 nfs-utils
701 nfsidmap
702 nilfs-utils
703 ninja
704 nitrogen
705 nm-connection-editor
706 nmap
707 node-gyp

708 nodejs
709 npm
710 npth
711 nspr
712 nss
713 nss-mdns
714 ntfs-3g
715 ntp
716 nvidia
717 nvidia-settings
718 nvidia-utils
719 nvidia-xrun
720 obs-input-overlay-bin
721 obs-linuxbrowser-bin
722 obs-studio-git
723 ocl-icd
724 oniguruma
725 openal
726 openc1-nvidia
727 openconnect
728 opencore-amr
729 openexr
730 openjpeg2
731 openmotif
732 openresolv
733 opencs
734 openssh
735 openssl
736 openssl-1.0
737 openvpn
738 opera-ffmpeg-codecs
739 opus
740 opusfile
741 orc

742 os-prober
743 p11-kit
744 p7zip
745 pacman
746 pacman-mirrorlist
747 pam
748 pambase
749 pango
750 pangomm
751 partclone
752 parted
753 partimage
754 pass
755 patch
756 pavucontrol
757 pbzip2
758 pciutils
759 pcre
760 pcre2
761 pcsclite
762 pepper-flash
763 perl
764 perl-data-dump
765 perl-encode-locale
766 perl-error
767 perl-file-listing
768 perl-html-parser
769 perl-html-tagset
770 perl-http-cookies
771 perl-http-daemon
772 perl-http-date
773 perl-http-message
774 perl-http-negotiate
775 perl-io-html

776 perl-io-socket-ssl
777 perl-json
778 perl-libwww
779 perl-lwp-mediatypes
780 perl-lwp-protocol-https
781 perl-mailtools
782 perl-net-http
783 perl-net-ssleay
784 perl-timedate
785 perl-try-tiny
786 perl-uri
787 perl-www-robotrules
788 perl-xml-parser
789 picom
790 pigz
791 pinentry
792 pixman
793 pixz
794 pkcs11-helper
795 pkgconf
796 plasma-framework
797 plasma5-applet-netctl-gui
798 polkit
799 polkit-qt5
800 poppler
801 poppler-glib
802 popt
803 postgresql
804 postgresql-libs
805 ppp
806 pptpclient
807 print-manager
808 procs-ng
809 progsreiserfs

```
810 protobuf
811 psmisc
812 pulseaudio
813 pulseaudio-alsa
814 pulseaudio-bluetooth
815 pulseaudio-jack
816 pulsemixer
817 pygobject-devel
818 pygtk
819 python
820 python-anytree
821 python-appdirs
822 python-babel
823 python-beaker
824 python-cachecontrol
825 python-cairo
826 python-chardet
827 python-colorama
828 python-dbus
829 python-dbus-common
830 python-distlib
831 python-distro
832 python-distutils-extra
833 python-docutils
834 python-evdev
835 python-gobject
836 python-html5lib
837 python-idna
838 python-imagesize
839 python-importlib-metadata
840 python-jinja
841 python-lockfile
842 python-lxml
843 python-mako
```

844 python-markdown
845 python-markupsafe
846 python-more-itertools
847 python-msgpack
848 python-ordered-set
849 python-packaging
850 python-pep517
851 python-pillow
852 python-pip
853 python-ply
854 python-progress
855 python-pycups
856 python-pycurl
857 python-pygments
858 python-pyparsing
859 python-pyqt5
860 python-pytoml
861 python-pytz
862 python-requests
863 python-retrying
864 python-setuptools
865 python-six
866 python-snowballstemmer
867 python-sphinx-alabaster-theme
868 python-sphinxcontrib-applehelp
869 python-sphinxcontrib-devhelp
870 python-sphinxcontrib-htmlhelp
871 python-sphinxcontrib-jsmath
872 python-sphinxcontrib-qthelp
873 python-sphinxcontrib-serializinghtml
874 python-urllib3
875 python-webencodings
876 python-yaml
877 python-zipp

```
878 python2
879 python2-appdirs
880 python2-cairo
881 python2-dbus
882 python2-gobject
883 python2-gobject2
884 python2-ordered-set
885 python2-packaging
886 python2-pyparsing
887 python2-pytz
888 python2-setuptools
889 python2-six
890 qpdf
891 qt5-base
892 qt5-declarative
893 qt5-graphicaleffects
894 qt5-imageformats
895 qt5-location
896 qt5-multimedia
897 qt5-quickcontrols
898 qt5-quickcontrols2
899 qt5-sensors
900 qt5-speech
901 qt5-svg
902 qt5-tools
903 qt5-webchannel
904 qt5-webengine
905 qt5-webkit
906 qt5-websockets
907 qt5-x11extras
908 quazip
909 raptor
910 rasqal
911 rdesktop
```


912 re2
913 readline
914 reaper-bin
915 redland
916 refind-efi
917 reiserfsprogs
918 rest
919 rhash
920 ristretto
921 rofi
922 rp-pppoe
923 rpcbind
924 rsync
925 rtkit
926 rubberband
927 ruby
928 rubygems
929 run-parts
930 sbc
931 screpy
932 screen
933 screenkey
934 scrot
935 sdl
936 sdl2
937 sdparm
938 sed
939 semver
940 serd
941 serf
942 sg3_utils
943 shadow
944 shared-mime-info
945 simple-mtpfs

946 slang
947 slop
948 smartmontools
949 snappy
950 solid
951 sonnet
952 sord
953 sound-theme-freedesktop
954 sox
955 speex
956 speexdsp
957 spice
958 sqlite
959 sratom
960 sshfs
961 startup-notification
962 steam
963 stoken
964 subversion
965 sudo
966 suil
967 swig
968 sysfsutils
969 syslinux
970 system-config-printer
971 systemd
972 systemd-libs
973 systemd-sysvcompat
974 t1lib
975 taglib
976 tar
977 tcl
978 tcpdump
979 tdb

```
980 teamspeak3
981 teamviewer
982 telegram-desktop
983 testdisk
984 texinfo
985 thin-provisioning-tools
986 thunar
987 thunderbird
988 tidy
989 tmate
990 tre
991 tree
992 tslib
993 ttf-hack
994 tumblr
995 twolame
996 tzdata
997 udisks2
998 unibilium
999 unrar
1000 unzip
1001 upower
1002 usb_modeswitch
1003 usbmuxd
1004 usbutils
1005 util-linux
1006 v4l-utils
1007 vamp-plugin-sdk
1008 vi
1009 vid.stab
1010 vim
1011 vim-runtime
1012 virtualbox
1013 virtualbox-host-modules-arch
```

1014 vlc
1015 volume_key
1016 vpncl
1017 vte-common
1018 vte-legacy
1019 vte3
1020 vulkan-icd-loader
1021 wacom-utility
1022 wavpack
1023 wayland
1024 wayland-protocols
1025 webkit2gtk
1026 webrtc-audio-processing
1027 wget
1028 which
1029 wine-staging
1030 wireless-regdb
1031 wireless_tools
1032 woff2
1033 wpa_supplicant
1034 wpebackend-fdo
1035 wvdial
1036 wvstreams
1037 x264
1038 x265
1039 xarchiver
1040 xcb-proto
1041 xcb-util
1042 xcb-util-cursor
1043 xcb-util-image
1044 xcb-util-keysyms
1045 xcb-util-renderutil
1046 xcb-util-wm
1047 xcb-util-xrm

1048 xclip
1049 xdg-dbus-proxy
1050 xdg-utils
1051 xf86-input-evdev
1052 xf86-input-keyboard
1053 xf86-input-libinput
1054 xf86-input-mouse
1055 xf86-input-synaptics
1056 xf86-input-vmmouse
1057 xf86-input-void
1058 xf86-input-wacom
1059 xf86-video-amdgpu
1060 xf86-video-ati
1061 xf86-video-dummy
1062 xf86-video-fbdev
1063 xf86-video-intel
1064 xf86-video-nouveau
1065 xf86-video-openchrome
1066 xf86-video-qxl
1067 xf86-video-vesa
1068 xf86-video-vmware
1069 xf86-video-voodoo
1070 xfce4-taskmanager
1071 xfce4-terminal
1072 xfconf
1073 xfsprogs
1074 xkblayout-state
1075 xkeyboard-config
1076 x12tpd
1077 xmlsec
1078 xorg-bdftopcf
1079 xorg-docs
1080 xorg-font-util
1081 xorg-font-utils

1082 xorg-fonts-100dpi
1083 xorg-fonts-75dpi
1084 xorg-fonts-alias
1085 xorg-fonts-encodings
1086 xorg-iceauth
1087 xorg-luit
1088 xorg-mkfontscale
1089 xorg-server
1090 xorg-server-common
1091 xorg-server-devel
1092 xorg-server-xdmx
1093 xorg-server-xephyr
1094 xorg-server-xnest
1095 xorg-server-xvfb
1096 xorg-server-xwayland
1097 xorg-sessreg
1098 xorg-setxkbmap
1099 xorg-smproxy
1100 xorg-util-macros
1101 xorg-x11perf
1102 xorg-xauth
1103 xorg-xbacklight
1104 xorg-xcmsdb
1105 xorg-xcursorgen
1106 xorg-xdpyinfo
1107 xorg-xdriinfo
1108 xorg-xev
1109 xorg-xgamma
1110 xorg-xhost
1111 xorg-xinit
1112 xorg-xinput
1113 xorg-xkbcomp
1114 xorg-xkbevd
1115 xorg-xkbutils

1116 xorg-xkill
1117 xorg-xlsatoms
1118 xorg-xlsclients
1119 xorg-xmodmap
1120 xorg-xpr
1121 xorg-xprop
1122 xorg-xrandr
1123 xorg-xrdb
1124 xorg-xrefresh
1125 xorg-xset
1126 xorg-xsetroot
1127 xorg-xvinfo
1128 xorg-xwd
1129 xorg-xwininfo
1130 xorg-xwud
1131 xorgproto
1132 xplc
1133 xvidcore
1134 xz
1135 yajl
1136 yaml-cpp
1137 yelp-tools
1138 yelp-xsl
1139 youtube-viewer
1140 zenity
1141 zeromq
1142 zip
1143 zlib
1144 zsh
1145 zstd