

# DustArch

# DustVoice's Arch Linux from scratch

David Holland

Version 5.2, 2020-01-09

# **Table Of Contents**

1. Inside the archiso	
1.1. Sync up pacman	
1.2. Formatting the drive	4
1.3. Preparing the chroot environment	
2. Entering the chroot	6
2.1. Installing additional packages	
2.2. Master of time	
2.3. Master of locales	9
2.4. Naming your machine	10
2.4.1. hostname	
2.4.2. hosts	
2.5. User setup	
2.5.1. Give root a password	14
2.5.2. Create a personal user	15
2.6. Preparing to boot	16
2.6.1. BIOS	17
2.6.2. UEFI	
2.6.3. grub config.	19
2.6.3.1. Adjust the timeout	20
2.6.3.2. Enable the recovery	21
2.6.3.3. NVIDIA fix	
2.6.3.4. Add power options	23
2.6.3.5. Installing memtest	24
2.6.3.6. Enabling hibernation	25
2.6.3.7. Generating the grub config	25
2.7. Secure Boot.	26
3. Inside the DustArch	27
3.1. Someone there?	28
3.2. Update and upgrade	30
3.3. Enabling the multilib repository	
3.4. Setting the correct shell	32
3.5. Version control	
3.6. Security is important	34
3.6.1. Smartcard shenanigans	
3.7. Additional required tools	
3.8. Setting up a home environment	37
3.8.1. Use dotfiles for a base config	
3.8.2. Set up gpg	39

3.8.3. Finalize the dotfiles	0
3.8.4. gpg-agent forwarding4	1
3.8.5. JUCE and FRUT 4	2
3.8.6. Back to your roots 4	3
3.9. Password management 4	4
3.10. python	5
3.11. ruby & asciidoctor	6
3.12. Using JUCE	7
3.13. Additional development tools	8
3.13.1. Code formatting	9
3.13.2. Documentation	0
3.13.3. Build tools	1
3.14. fstab	2
3.15. Audio	3
3.15.1. alsa	4
3.15.2. pulseaudio	5
3.15.3. jack	6
3.15.4. Audio handling	7
3.16. Bluetooth	8
3.17. Graphical desktop environment	1
3.17.1. NVIDIA	2
3.17.2. Launching the graphical environment	3
3.17.2.1. The NVIDIA way	4
3.18. GUI Software	6
3.18.1. Session Lock	7
3.18.2. xfce-polkit	7
3.18.3. Desktop background 6	8
3.18.4. Compositing software 6	9
3.18.5. networkmanager applet	0
3.18.6. Keyboard	1
3.18.7. X clipboard	2
3.18.8. Taking screen shots	3
3.18.9. Image viewer	4
3.18.10. File manager	5
3.18.10.1. Android file transfer	6
3.18.11. Archive manager	8
3.18.12. Partition management	9
3.18.13. PDF viewer	0
3.18.13.1. Using the GUI	1
3.18.13.2. Using the framebuffer	
3.18.14. Web browser	3

	3.18.14.1. Entering the dark side	84
	3.18.15. Office utilities	85
	3.18.15.1. Printing	86
	3.18.16. Process management	87
	3.18.17. Communication	88
	3.18.17.1. Email	89
	3.18.17.2. Telegram	90
	3.18.17.3. TeamSpeak 3	91
	3.18.17.4. Discord	92
	3.18.18. Video software	93
	3.18.18.1. Viewing video	94
	3.18.18.2. Creating video	95
	3.18.18.3. Live stream a terminal session	96
	3.18.18.4. Editing video	97
	3.18.18.5. Utilizing video	98
	3.18.19. Ardour	99
	3.18.20. Virtualization1	00
	3.18.21. Gaming	01
	3.18.22. Wacom	02
	3.18.23. VNC & RDP	03
4. Up	ograding the system	04
4.	1. Fixing a faulty kernel upgrade	05

### 1. Inside the archiso

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

As Arch Linux is a rolling release Linux distribution, it is advised, to have a working internet connection, in order to get the latest package upgrades and to install additional software, as the archiso only has very few packages available from cache.

a

Furthermore, one should bear in mind that depending on the version, or rather modification date, the guide may already be outdated. If you encounter any problems along the way, you will either have to resolve the issue yourself, or utilize the great ArchWiki, or the Arch Linux forums.

### 1.1. Sync up pacman

First of all we need to sync up pacman's package repository, in order to be able to install packages

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

After doing that, we can now install any software from the official repositories by issuing

```
root@archiso ~ # pacman -S <package_name>
```

where you would replace <package\_name> with the actual package name.

If you don't know the exact package name, or if you just want to search for a keyword, for example xfce to list all packages having to do something with xfce, use

```
root@archiso ~ # pacman -Ss <keyword>
```

If you want to remove an installed package, just use

```
root@archiso ~ # pacman -Rsu <package_name>
```



If you have to force remove, which you should use **with extreme caution**, you can use

```
root@archiso ~ # pacman -Rdd <package_name>
```

If you want to install a package from the AUR, I would proceed as follows

1. cd into a dedicated directory

```
dustvoice@archiso ~ $ mkdir AUR
dustvoice@archiso ~ $ cd AUR
```

2. Clone the package with git

```
dustvoice@archiso ~/AUR $ git clone https://aur.archlinux.org/pacman-git.git
```

3. Switch to the package directory

```
dustvoice@archiso ~/AUR $ cd pacman-git
```

4. Execute makepkg

```
dustvoice@archiso ~/AUR/pacman-git $ makepkg -si
```

5. Delete all files created by makepkg, in order to easily see, if a package needs an update by using git fetch --all and git status

```
dustvoice@archiso ~/AUR/pacman-git $ git reset HEAD --hard dustvoice@archiso ~/AUR/pacman-git $ git clean -fdx
```



You might have to resolve any AUR dependencies, which can't be resolved with pacman.



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

### 1.2. 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.



With bigger RAM sizes available today, this isn't necessary anymore. To be exact, every distribution has different recommendations for swap sizes.

Also swap size heavily depends on whether you want to be able to hibernate, etc.

You should make the swap size at least your RAM size and for RAM sizes over 4GB and the wish to hibernate, at least one and a half your RAM size.



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

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
```

## 1.3. 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 an fstab using genfstab

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

and you're ready to enter the chroot environment.

# 2. Entering the chroot



As we want to set up our new system, we need to have access to the different partitions, the internet, etc. which we wouldn't get by solely using chroot.

That's why we are using arch-chroot, provided by the arch-install-scripts package already shipped with the archiso. This script takes care of all that stuff, so we can set up our system properly.

root@archiso ~ # arch-chroot /mnt

Et Voila! You successfully chrooted inside your new system and you'll be greeted by a bash prompt.

### 2.1. Installing additional packages

First off you'll probably need a text editor.

There are many command line text editors available, like nano, vi, vim, emacs, etc.

I'll be using neovim, though it shouldn't matter what editor you choose.

```
[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 the need of 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
```

### 2.2. 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. In my case, my timezone resides under /usr/share/zoneinfo/Europe/Berlin.

To achieve the desired result, I want to symlink this to /etc/localtime and set the hardware clock.

```
[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

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

and check that everything is alright

```
[root@archiso /]# timedatectl status
```

### 2.3. Master of locales

Now you have to generate your locale information.

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



I recommend to always uncomment en\_US.UTF-8 UTF8, 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"
```

After that we're done with this part.

## 2.4. Naming your machine

2.4. Naming your machine
Now we can set the hostname and add hosts entries.
Apart from being mentioned in your command prompt, the hostname also serves the purpose of identifying, or naming your machine. This enables you to see your PC in your router, etc.

## **2.4.1.** hostname

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

### **2.4.2.** hosts

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

#### /etc/hosts

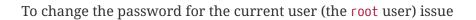
```
# Static table lookup for hostnames.
# See hosts(5) for details.

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

## 2.5. User setup

1
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.

### 2.5.1. Give root a password





and choose a new password.

#### 2.5.2. 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@archiso /]# pacman -S fish
[root@archiso /]# useradd -m -p "" -G
"adm,audio,disk,floppy,kvm,log,lp,network,rfkill,scanner,storage,users,optical,power,w
heel" -s /usr/bin/fish dustvoice
[root@archiso /]# passwd dustvoice
[root@archiso /]# 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.

# 2.6. 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@archiso /]# pacman -S grub dosfstools os-prober mtools

and if you want to use **UEFI**, also

[root@archiso /]# pacman -S efibootmgr

#### **2.6.1.** 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@archiso /]# 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.

#### 2.6.2. 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@archiso /]# 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

A

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

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

[root@archiso /]# ls /boot/EFI/grub

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

## 262 acub confi

	2.6.3. grub config							
	In all cases, you now have to create the main grub.cfg configuration file.							
But before we actually generate it, we'll make some changes to the default <code>grub</code> settings, wh <code>grub.cfg</code> will be generated from.								

#### 2.6.3.1. Adjust the timeout

First of all, I want my grub menu to wait indefinitely for my command to boot an 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 the Windows Boot Loader from the grub menu.

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

a

Another way of achieving what I described, would be to make grub remember the last selection.

/etc/default/grub

GRUB\_TIMEOUT=5
GRUB\_DEFAULT=saved
GRUB\_SAVEDEFAULT="true"

### 2.6.3.2. Enable the recovery

GRUB\_DISABLE\_RECOVERY=false

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

/etc/default/grub		

#### 2.6.3.3. NVIDIA fix

Now, as I'm using the binary NVIDIA driver for my graphics card, I also want to make sure, to revert <code>grub</code> back to text mode, after I select a boot entry, in order for the NVIDIA driver to work properly. You might not need this

/etc/de	fault/grub
/ CLC/ CLC	Junity Si up

GRUB\_GFXPAYLOAD\_LINUX=text

### 2.6.3.4. Add power options

I also want to add two 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
}
```

#### 2.6.3.5. Installing memtest

As I want all possible options to possibly troubleshoot my PC right there in my grub menu, without the need to boot into a live OS, I also want to have a memory tester there.

#### BIOS

For a BIOS setup, you'll need memtest86+

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

#### UEFI

For a UEFI setup, you'll need memtest86-efi from the AUR.

```
[root@archiso /]# pacman -S base-devel
[root@archiso /]# sudo -iu dustvoice
[dustvoice@archiso ~]$ cd AUR
[dustvoice@archiso ~/AUR]$ git clone https://aur.archlinux.org/memtest86-efi
[dustvoice@archiso ~/AUR]$ cd memtest86-efi
[dustvoice@archiso ~/AUR/memtest86-efi]$ makepkg -si
[dustvoice@archiso ~/AUR/memtest86-efi]$ git reset HEAD --hard
[dustvoice@archiso ~/AUR/memtest86-efi]$ git clean -fdx
[dustvoice@archiso ~/AUR/memtest86-efi]$ exit
```

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

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

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

#### 2.6.3.6. Enabling hibernation

In order to use the hibernation feature, you'll have to make sure that your swap partition/file is at least the size of your RAM.

After that we need to perform two tasks

1. Add the resume hook to /etc/mkinitcpio.conf, before fsck and definetely after block

/etc/mkinitcpio.conf

HOOKS=(base udev autodetect modconf block filesystems keyboard resume fsck)

2. Add the resume kernel parameter to /etc/default/grub, containing my swap partition UUID, in my case

/etc/default/grub

GRUB\_CMDLINE\_LINUX\_DEFAULT="loglevel=3 quiet resume=UUID=097c6f11-f246-40eb-a702-ba83c92654f2"

After that we have to run

[root@archiso /]# mkinitcpio -p linux



If you have to change anything, like the swap partition UUID, inside the grub configuration files, you'll always have to rerun grub-mkconfig as explained in Generating the grub config.

#### 2.6.3.7. Generating the grub config

Now we can finally generate our grub.cfg

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

Now you're good to boot into your new system.

### 2.7. Secure Boot

I know I told you that you're now good to boot into your new system. That is only correct, if you're **not** using Secure Boot.

You can either proceed by disabling Secure Boot in your firmware settings, or by using shim as kind of a pre-bootloader, as well as signing your bootloader (grub) and your kernel.

If you decided on using Secure Boot, you will first have to install shim-signed from AUR.

```
[root@archiso /]# sudo -iu dustvoice
[dustvoice@archiso ~]$ cd AUR
[dustvoice@archiso ~/AUR]$ git clone https://aur.archlinux.org/shim-signed
[dustvoice@archiso ~/AUR]$ cd shim-signed
[dustvoice@archiso ~/AUR/shim-signed]$ makepkg -si
[dustvoice@archiso ~/AUR/shim-signed]$ git reset HEAD --hard
[dustvoice@archiso ~/AUR/shim-signed]$ git clean -fdx
[dustvoice@archiso ~/AUR/shim-signed]$ exit
```

Now we just need to copy shimx64.efi, as well as mmx64.efi to our EFI System Partition

```
[root@archiso /]# cp /usr/share/shim-signed/shimx64.efi /boot/EFI/grub/
[root@archiso /]# cp /usr/share/shim-signed/mmx64.efi /boot/EFI/grub/
```



If you have to use bcdedit from within Windows, as explained previously, you need to adapt the command accordingly

```
bcdedit /set {bootmgr} path \EFI\grub\shimx64.efi
```

Now you will be greeted by MokManager everytime you update your bootloader or kernel.

Just choose Enroll hash from disk and enroll your bootloader (grubx64.efi) and kernel (vmlinuz-linux).

Reboot and your system should fire up just fine.

## 3. Inside the DustArch

This section helps at setting up the customized system from within an installed system.

This section mainly provides aid with the basic set up tasks, like networking, dotfiles, etc.

Not everything in this section is mandatory.

a

This section is rather a guideline, because it is easy to forget some steps needed, for example jack for audio production, that only become apparent, when they're needed.

It is furthermore the responsibility of the reader to decide which steps to skip and which need further research. As I mentioned, this is only a guide and not the answer to everything.

### 3.1. Someone there?

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

To view the network interfaces with all their properties, we can issue

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

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

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

Everything should run smoothly if you have a wired connection.

If there is no connection and you're indeed using a wired connection, try restarting the NetworkManager service

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

and then try pinging again.

If you're trying to utilize a Wi-Fi connection, use nmcli, the NetworkManager's 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 radio
```

and if not, enable it

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

Now make sure your Wi-Fi interface appears under

```
dustvoice@DustArch ~> nmcli device
```

Rescan for available networks

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

#### and list all found networks

dustvoice@DustArch ~> nmcli device wifi list

#### After that connect to the network

dustvoice@DustArch ~> nmcli device wifi connect --ask

Now try pinging again.

# 3.2. Update and upgrade



ustvoice@DustArch ~> sudo pac	man -Syu		

## 3.3. Enabling the multilib repository

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

/etc/pacman.conf

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

and update pacman's package repositories afterwards

dustvoice@DustArch ~> sudo pacman -Sy

### 3.4. Setting the correct shell

Of course you can use any shell you want. In my case I'll be using the fish shell.



I am using fish because of its superb auto completion functionality, though that might not be what you're looking for.

Also, be aware that fish might not have the best documentation and is also not sh compliant.

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.

# 3.5. 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  $\mbox{.}\mbox{\tt gitconfig}$  in just a second.

# 3.6. Security is important

If you've followed the tutorial using a recent version of the archiso, 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						

# 3.6.1. 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
```

After that, you should be able to see your smartcard being detected

```
dustvoice@DustArch ~> gpg --card-status
```



If your smartcard still isn't detected, try logging off completely or even restarting, as that sometimes is the solution to the problem.

# 3.7. Additional required tools

To minimize the effort required by the following steps, we'll install most of the required packages beforehand

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

This will ensure, we proceed through the following section without the need for interruption, because a package needs to be installed, so the following content can be condensed to the relevant informations.

# 3.8. 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.

I decided on this, as I want to edit files with elevated permissions and still have the same editor style and functions/plugins.

a

Note that this comes with some drawbacks. For example, if I change a configuration for my dustvoice user, I would have to regularly update it for the root user too. This bears the problem, that I have to register my smartcard for the root user. This in turn is problematic, cause the gpg-agent used for ssh authentication, doesn't behave well when used within a su or sudo -i session. So in order to update root's config files I would either need to symlink everything, which I won't do, or I'll need to login as the root user now and then, to update everything.



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 with some configuration files that reside in a git repository. This means I will have to reside to using the https URL of the repository first and later changing the URL either in the corresponding .git/config file, or by issuing the appropriate command.

## 3.8.1. Use dotfiles for a base config

To provide myself with a base configuration, which I can then extend, I have created a dotfiles repository, which contains all kinds of configurations.

The special thing about this dotfiles repository is that it is my home folder. By using a curated .gitignore file, I'm able to only include the configuration files I want to keep between installs into the repository and ignore everything else.

To achieve this very specific setup, I have to turn my home directory into said dotfiles repository first

```
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
```

Now I can issue any git command in my ~ directory, because it now is a git repository.

### 3.8.2. Set up gpg

As I wanted to keep my dotfiles repository as modular as possible, I utilize git's submodule feature. Furthermore I want to use my nvim repository, which contains all my configurations and plugins for neovim, on Windows, but without all the Linux specific configuration files. I am also using the Pass repository on my Android phone and Windows PC, where I only need this repository without the other Linux configuration files.

Before we'll be able to update the submodules (nvim config files and password-store) though, we will have to setup our gpg key as an ssh key, as I use it to authenticate

```
[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
```



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

Now, as mentioned before, I'll switch to using ssh for authentication, rather than https

```
[I] dustvoice@DustArch ~>
$ git remote set-url origin git@github.com:DustVoice/dotfiles.git
```

As the best method to both make fish recognize all the configuration changes, as well as the gpgagent behave properly, is to re-login, we'll do just that

```
[I] dustvoice@DustArch ~>
$ exit
```

It is very important to note, that I mean a real re-login.



That means that if you've used ssh to log into your machine, it probably won't be sufficient to login into a new ssh session. You'll probably need to restart the machine completely.

### 3.8.3. 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 ~
```

## 3.8.4. gpg-agent forwarding

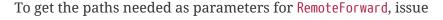
Now there is only one thing left to do, in order to make the gpg setup complete: gpg-agent forwarding over ssh. This is very important for me, as I want to use my smartcard on my development server too, which requires me, to forward/tunnel my gpg-agent to my remote machine.

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.





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

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

/etc/ssh/sshd\_config

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

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

## 3.8.5. 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
```

# 3.8.6. 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.

# 3.9. Password management

9	
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	

# **3.10.** 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-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
```

# 3.11. 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 2.7.0, or if you upgrade your ruby version, 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_path $fish_user_path ~/.gem/ruby/2.7.0/bin
```



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

# 3.12. 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

# 3.13. Additional development tools

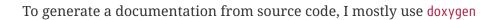
Here are just some exa have.	mples of developmen	t tools one could in	nstall in addition to	what we already

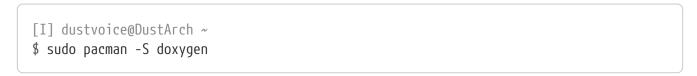
# 3.13.1. Code formatting

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

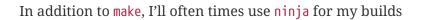
[I] dustvoice@DustArch ~
\$ sudo pacman -S astyle

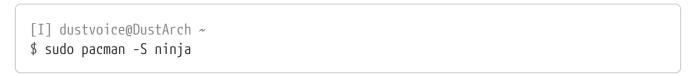
# 3.13.2. Documentation





# 3.13.3. Build tools





## **3.14.** 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
                           0 1
 defaults
4
5 UUID="DC88-5A4E"
                                                     /mnt/projects
                                                                      exfat
  rw, relatime
                           0 0
6
7 UUID=7A16569B51903310
                                                     /mnt/data
                                                                      ntfs
 ro, nosuid, nodev, noauto 00
```

### 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.

Woll why wouldn't way want and			
Well, why wouldn't you want audio	•		

### 3.15.1. alsa



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

To quickly setup audio this way, install also and also-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.ctl.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.

## 3.15.2. 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
```

# **3.15.3.** 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
```

# 3.15.4. 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.

## 3.16. 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 ~
$ 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
```

# 3.17. 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 arandr

### 3.17.1. 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.

3.17.2. 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 <b>Ctrl+Alt+<number></number></b> to switch ttys.

### **3.17.2.1. 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 ~

$ cd AUR
[I] dustvoice@DustArch ~/AUR

$ git clone https://aur.archlinux.org/nvidia-xrun.git
[I] dustvoice@DustArch ~/AUR

$ cd nvidia-xrun
[I] dustvoice@DustArch ~/AUR/nvidia-xrun

$ makepkg -si
[I] dustvoice@DustArch ~/AUR/nvidia-xrun

$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/nvidia-xrun

$ git clean -fdx
```

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

```
[I] dustvoice@DustArch ~
$ sudo pacman -S nvidia
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/nvidia-xrun-pm.git
[I] dustvoice@DustArch ~/AUR
$ cd nvidia-xrun-pm
[I] dustvoice@DustArch ~/AUR/nvidia-xrun-pm
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/nvidia-xrun-pm
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/nvidia-xrun-pm
$ git clean -fdx
```

instead.

Now we need to blacklist **both** nouveau and nyidia 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.

f Of the

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

0

```
[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.

# 3.18. GUI Software

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

### 3.18.1. Session Lock

Probably the first thing you'll want to set up is a session locker, which locks your X-session after resuming from sleep, hibernation, etc. It then requires you to input your password again, so no unauthorized user can access you machine.

I'll use xss-lock to hook into the necessary systemd events and i3lock as my locker.

For that I have to install both

```
[I] dustvoice@DustArch ~
$ sudo pacman -S xss-lock i3lock
```

And we're done actually, as I have placed the required command to start xss-lock with the right parameters inside my i3 configuration file.

If you use something other than i3, you need to make sure this command gets executed upon start of the X-session

```
xss-lock -- i3lock -n -e -c 33333
```

### 3.18.2. xfce-polkit

In order for GUI applications to acquire sudo permissions, we need to install a PolicyKit authentication agent.

We could use <code>gnome-polkit</code> for that purpose, which resides inside the official repositories, but I decided on using <code>xfce-polkit</code> from the AUR.

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/xfce-polkit.git
[I] dustvoice@DustArch ~/AUR
$ cd xfce-polkit
[I] dustvoice@DustArch ~/AUR/xfce-polkit
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/xfce-polkit
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/xfce-polkit
$ git clean -fdx
```

Now you just need to startup xfce-polkit before trying to execute something like gparted and you'll be prompted for your password.

As I already launch it as a part of my i3 configuration, I won't have to worry about that.

# 3.18.3. Desktop background



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

## 3.18.4. 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>
```

## 3.18.5. 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.

# **3.18.6. 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 ~

$ cd AUR
[I] dustvoice@DustArch ~/AUR

$ git clone https://aur.archlinux.org/xkblayout-state.git
[I] dustvoice@DustArch ~/AUR

$ cd xkblayout-state
[I] dustvoice@DustArch ~/AUR/xkblayout-state

$ makepkg -si
[I] dustvoice@DustArch ~/AUR/xkblayout-state

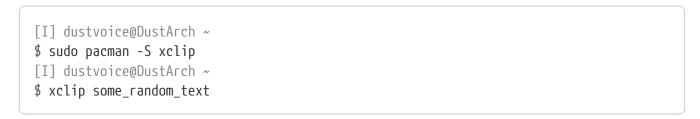
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/xkblayout-state

$ git clean -fdx
```

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

# 3.18.7. X clipboard

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



# 3.18.8. Taking screen shots

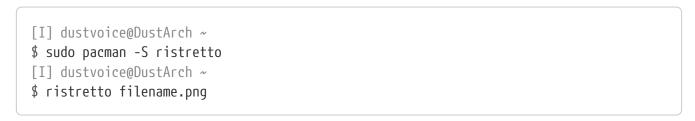
For this functionality, especially in combination with rofi, use scrot

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

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

# 3.18.9. Image viewer

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



## 3.18.10. 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 ~

$ cd AUR
[I] dustvoice@DustArch ~/AUR

$ git clone https://aur.archlinux.org/gigolo.git
[I] dustvoice@DustArch ~/AUR

$ cd gigolo
[I] dustvoice@DustArch ~/AUR/gigolo

$ makepkg -si
[I] dustvoice@DustArch ~/AUR/gigolo

$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/gigolo

$ git clean -fdx
```

#### 3.18.10.1. Android file transfer

To furthermore enable the transfer of files between your PC and your android phone, you'll have to install <a href="mailto:mtp">mtp</a> and <a href="mailto:gvfs-mtp">gvfs-mtp</a>

```
[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 ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/simple-mtpfs.git
[I] dustvoice@DustArch ~/AUR
$ cd simple-mtpfs
[I] dustvoice@DustArch ~/AUR/simple-mtpfs
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/simple-mtpfs
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/simple-mtpfs
$ git clean -fdx
```

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.

# 3.18.11. 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
```

# 3.18.12. 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

### 3.18.13. 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.

## 3.18.13.1. 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
```

### 3.18.13.2. Using the framebuffer

If you want to not always use the graphical desktop with <code>mupdf</code>, you might be interested in the <code>fbgs</code> 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.

#### 3.18.14. 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 ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/brave-bin.git
[I] dustvoice@DustArch ~/AUR
$ cd brave-bin
[I] dustvoice@DustArch ~/AUR/brave-bin
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/brave-bin
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/brave-bin
$ git clean -fdx
[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.

### 3.18.14.1. 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 ~

$ cd AUR
[I] dustvoice@DustArch ~/AUR

$ git clone https://aur.archlinux.org/tor-browser.git
[I] dustvoice@DustArch ~/AUR

$ cd tor-browser
[I] dustvoice@DustArch ~/AUR/tor-browser

$ makepkg -si
[I] dustvoice@DustArch ~/AUR/tor-browser

$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/tor-browser

$ git clean -fdx
```



You might have to check out how to import the gpg keys on the AUR page of torbrowser.

# 3.18.15. Office utilities



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

#### 3.18.15.1. 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 ~
$ 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.

## 3.18.16. Process management

The native tool is top.

The next evolutionary step would be <a href="https://h

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

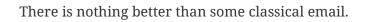
If you prefer a GUI for that kind of task, install xfce4-taskmanager

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

### 3.18.17. Communication

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

### 3.18.17.1. Email



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

## **3.18.17.2. Telegram**



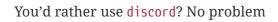
[I] dustvoice@DustArch ~
\$ sudo pacman -S telegram-desktop

## 3.18.17.3. TeamSpeak 3





### 3.18.17.4. Discord



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

## /idoo

Just some additional software related to videos.							
jast some additional somware i	cluted to videos.						

# 3.18.18.1. Viewing video



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

#### 3.18.18.2. Creating video

obs should be the right choice

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/obs-studio-git
[I] dustvoice@DustArch ~/AUR
$ cd obs-studio-git
[I] dustvoice@DustArch ~/AUR/obs-studio-git
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/obs-studio-git
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/obs-studio-git
$ git clean -fdx
```

#### **Showing keystrokes**

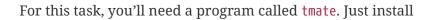
In order to show the viewers what keystrokes you're pressing, you can use something like screenkey

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/screenkey.git
[I] dustvoice@DustArch ~/AUR
$ cd screenkey
[I] dustvoice@DustArch ~/AUR/screenkey
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/screenkey
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/screenkey
$ git clean -fdx
[I] dustvoice@DustArch ~/Sur/screenkey
$ git clean -fdx
[I] dustvoice@DustArch ~/Sur/screenkey
```



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

### 3.18.18.3. Live stream a terminal session





### and run it



### **3.18.18.4. Editing video**

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

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/davinci-resolve.git
[I] dustvoice@DustArch ~/AUR
$ cd davinci-resolve
[I] dustvoice@DustArch ~/AUR/davinci-resolve
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/davinci-resolve
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/davinci-resolve
$ git clean -fdx
```

### 3.18.18.5. Utilizing video

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

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/teamviewer.git
[I] dustvoice@DustArch ~/AUR
$ cd teamviewer
[I] dustvoice@DustArch ~/AUR/teamviewer
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/teamviewer
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/teamviewer
$ git clean -fdx
```

### 3.18.19. 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
```

### 3.18.20. 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.

# 3.18.21. 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
```

#### 3.18.22. 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 ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/wacom-utility.git
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/gksu.git
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/libgks.git
[I] dustvoice@DustArch ~/AUR
$ cd libgks
[I] dustvoice@DustArch ~/AUR/libgks
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/libgks
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/libgks
$ git clean -fdx
[I] dustvoice@DustArch ~/AUR/libgks
$ cd ..
[I] dustvoice@DustArch ~/AUR
$ cd qksu
[I] dustvoice@DustArch ~/AUR/gksu
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/gksu
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/gksu
$ git clean -fdx
[I] dustvoice@DustArch ~/AUR/gksu
$ cd ..
[I] dustvoice@DustArch ~/AUR
$ cd wacom-utility
[I] dustvoice@DustArch ~/AUR/wacom-utility
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/wacom-utility
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/wacom-utility
$ git clean -fdx
```

#### 3.18.23. VNC & RDP

In order to connect to a machine over VNC or to connect to a machine using the Remote Desktop Protocol, for example to connect to a Windows machine, I'll need to install freerdp from the AUR, as well as libvncserver, for RDP and VNC functionality respectively, as well as remmina, to have a GUI client for those two protocols.

```
[I] dustvoice@DustArch ~
$ cd AUR
[I] dustvoice@DustArch ~/AUR
$ git clone https://aur.archlinux.org/freerdp.git
[I] dustvoice@DustArch ~/AUR
$ cd freerdp
[I] dustvoice@DustArch ~/AUR/freerdp
$ makepkg -si
[I] dustvoice@DustArch ~/AUR/freerdp
$ git reset HEAD --hard
[I] dustvoice@DustArch ~/AUR/freerdp
$ git clean -fdx
[I] dustvoice@DustArch ~/AUR/freerdp
$ cd ~
[I] dustvoice@DustArch ~
$ sudo pacman -S libvncserver remmina
```

Now you can set up all your connections inside remnina.

# 4. 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.

# 4.1. 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 an fstab entry for the boot partition, in order to avoid such dilemma in the future.

See fstab for more.