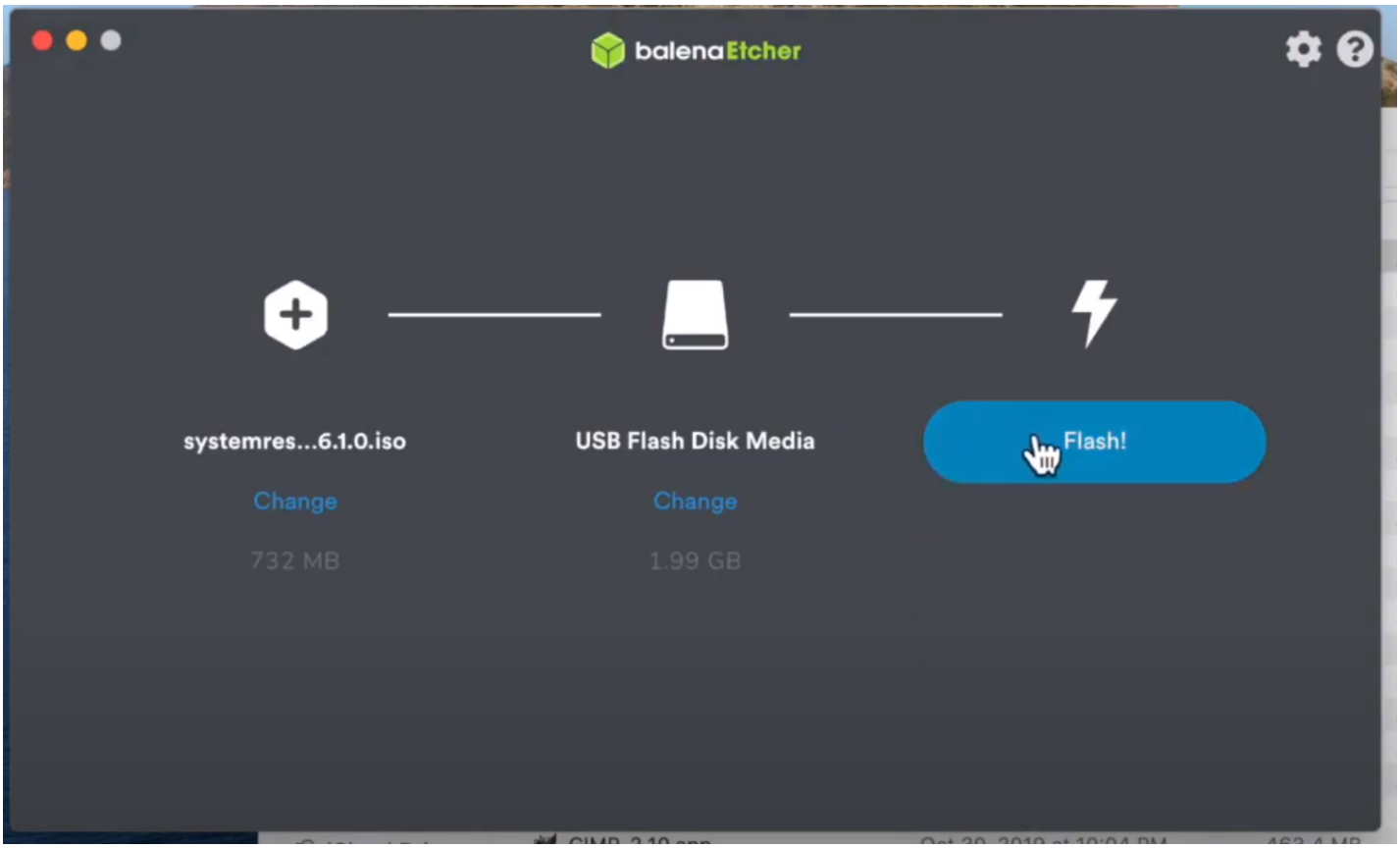
Prepare your live ISO USB

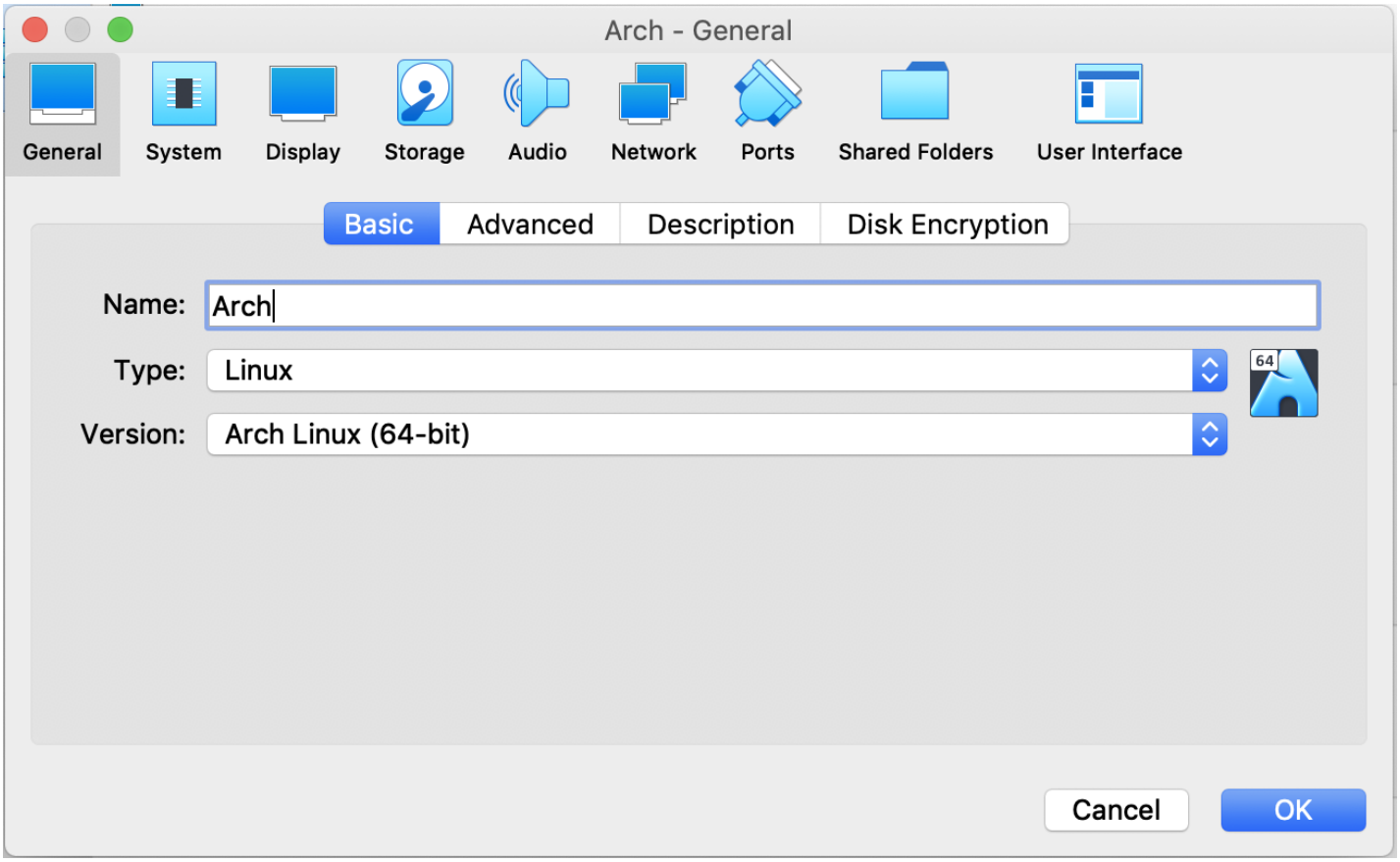
Go to here and pick your closest mirror to download the ISO, then burn into a USB. Plug your USB into computer and boot the live ISO, it will automatic login to install Arch, then follow the steps below to install your own `Arch Linux`.



After that, reboot into your live ISO USB to continue the installation.

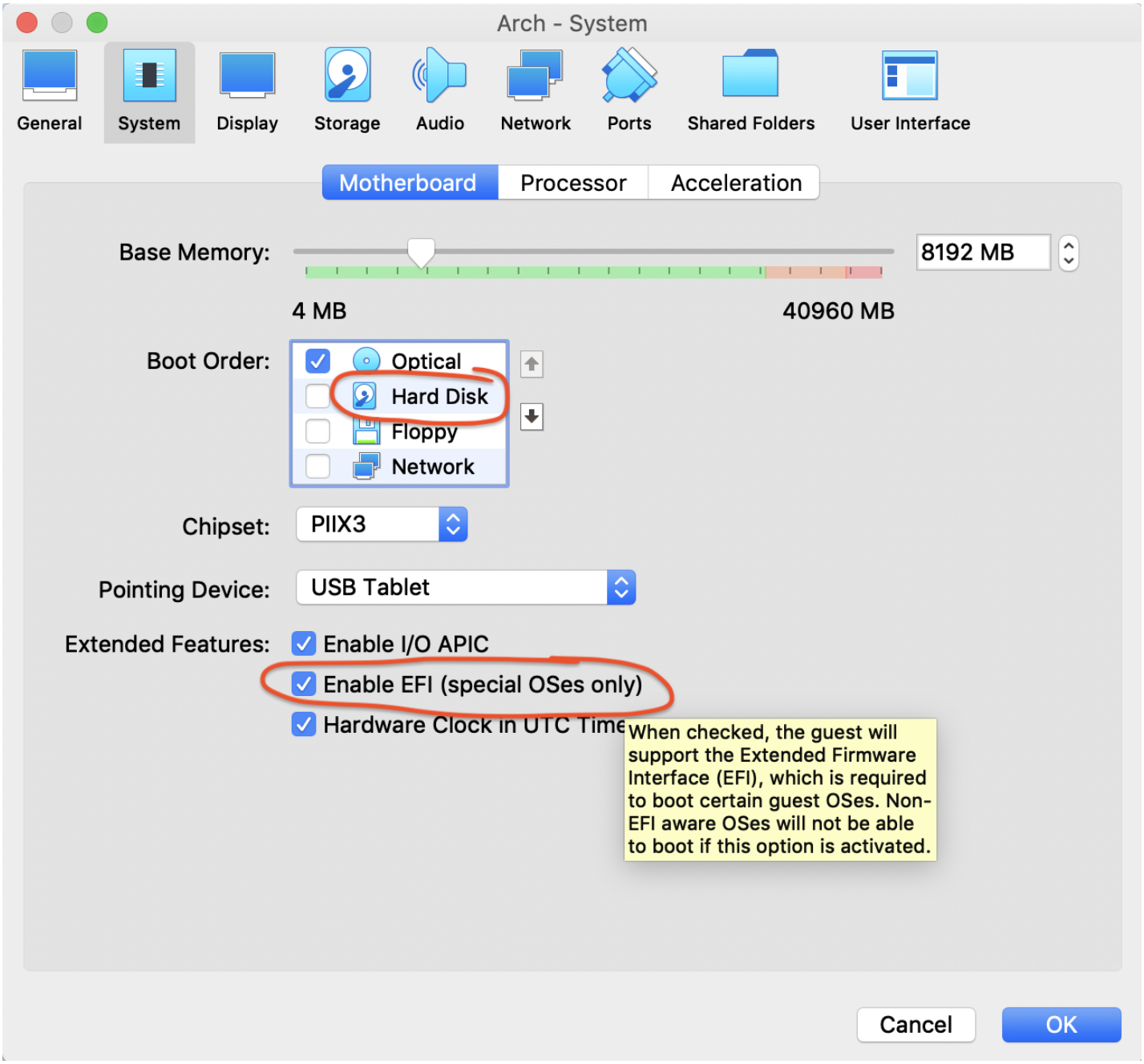
Use `VirtualBox` to boot live ISO

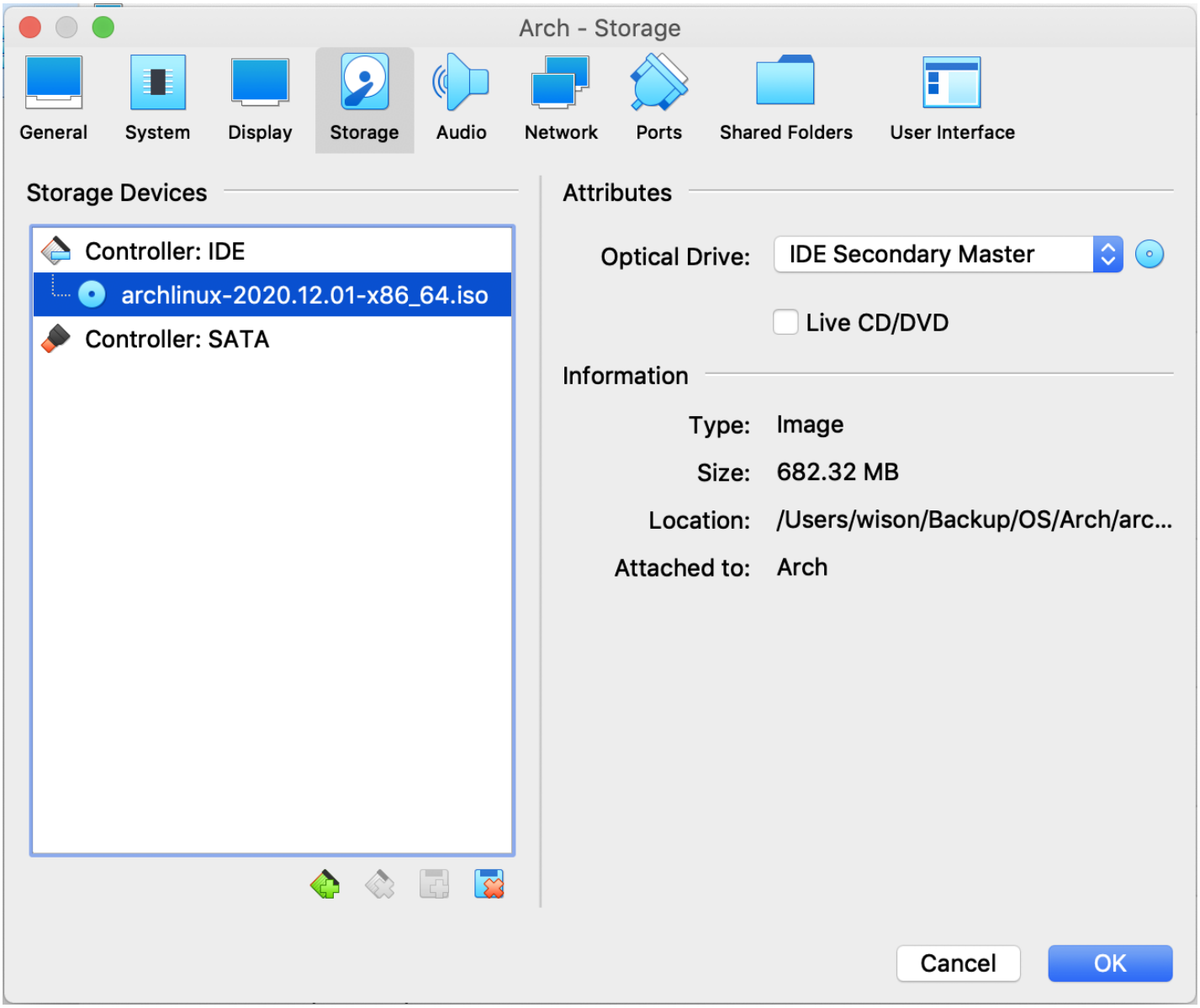
Actually, you can run the installation process inside `VirutalBox`. Here is the step-by-step walk through:

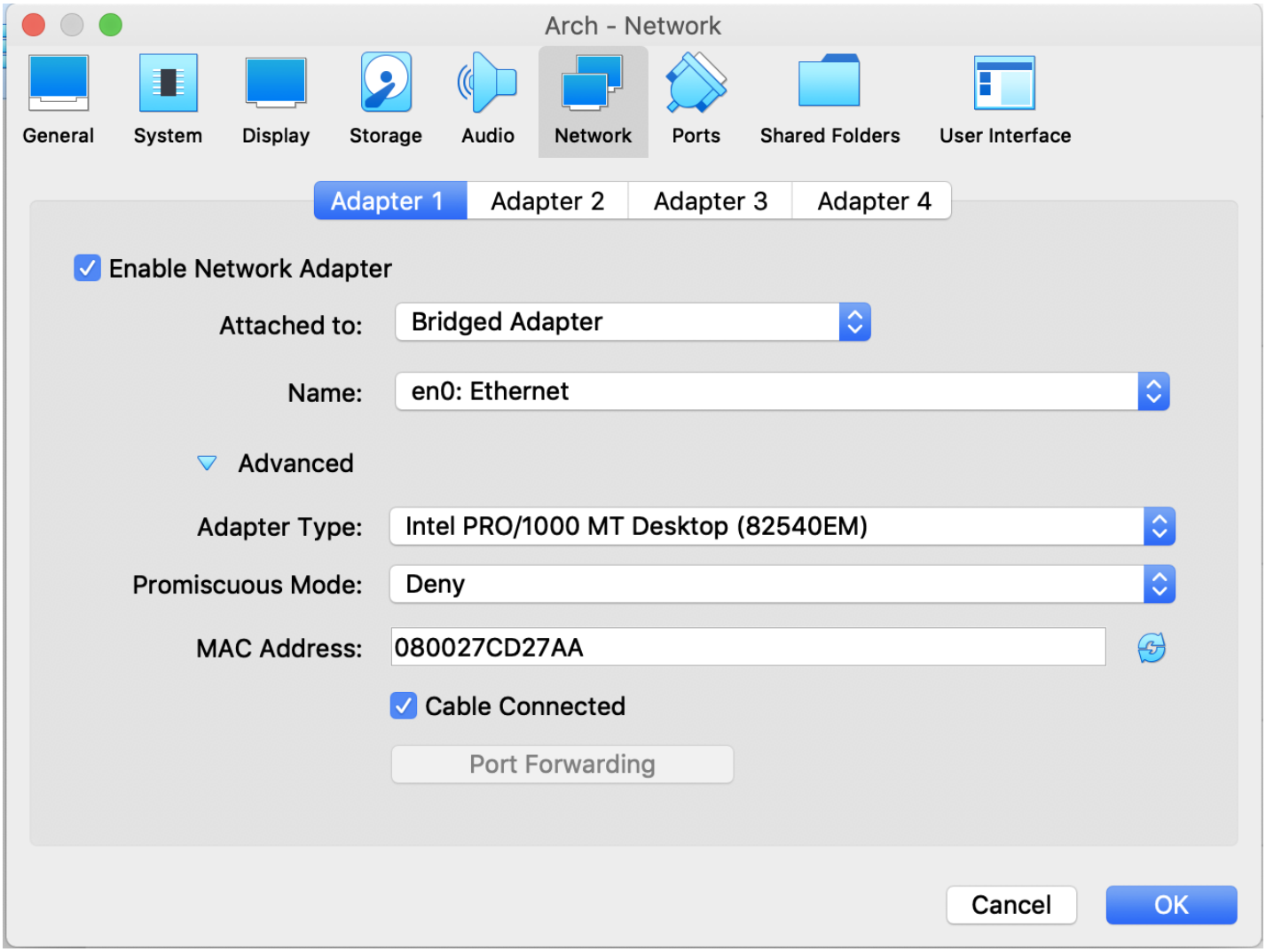


As you boot from the live ISO, so you don't any harddrive at all.

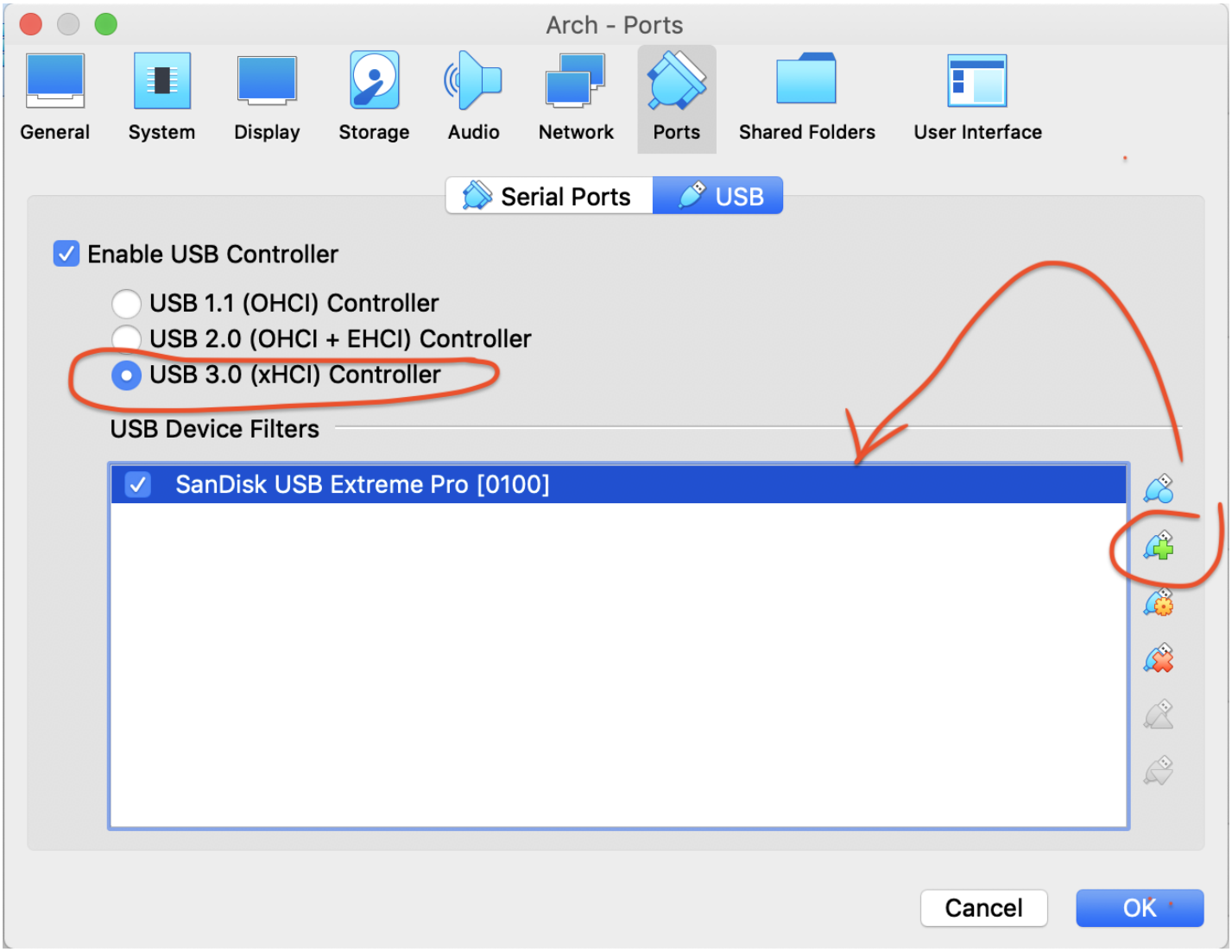
And make sure check the `Enable EFI` !!!


Arch - System
General
System
Display
Storage
Audio
Network
Ports
Shared Folders
User Interface
Motherboard
Processor
Acceleration
Base Memory:
8192 MB
4 MB
40960 MB
Boot Order:
Optical
Hard Disk
Floppy
Network
Chipset:
PIIX3
Pointing Device:
USB Tablet
Extended Features:
Enable I/O APIC
Enable EFI (special OSes only)
Hardware Clock in UTC Time
When checked, the guest will support the Extended Firmware Interface (EFI), which is required to boot certain guest OSes. Non-EFI aware OSes will not be able to boot if this option is activated.
Cancel
OK

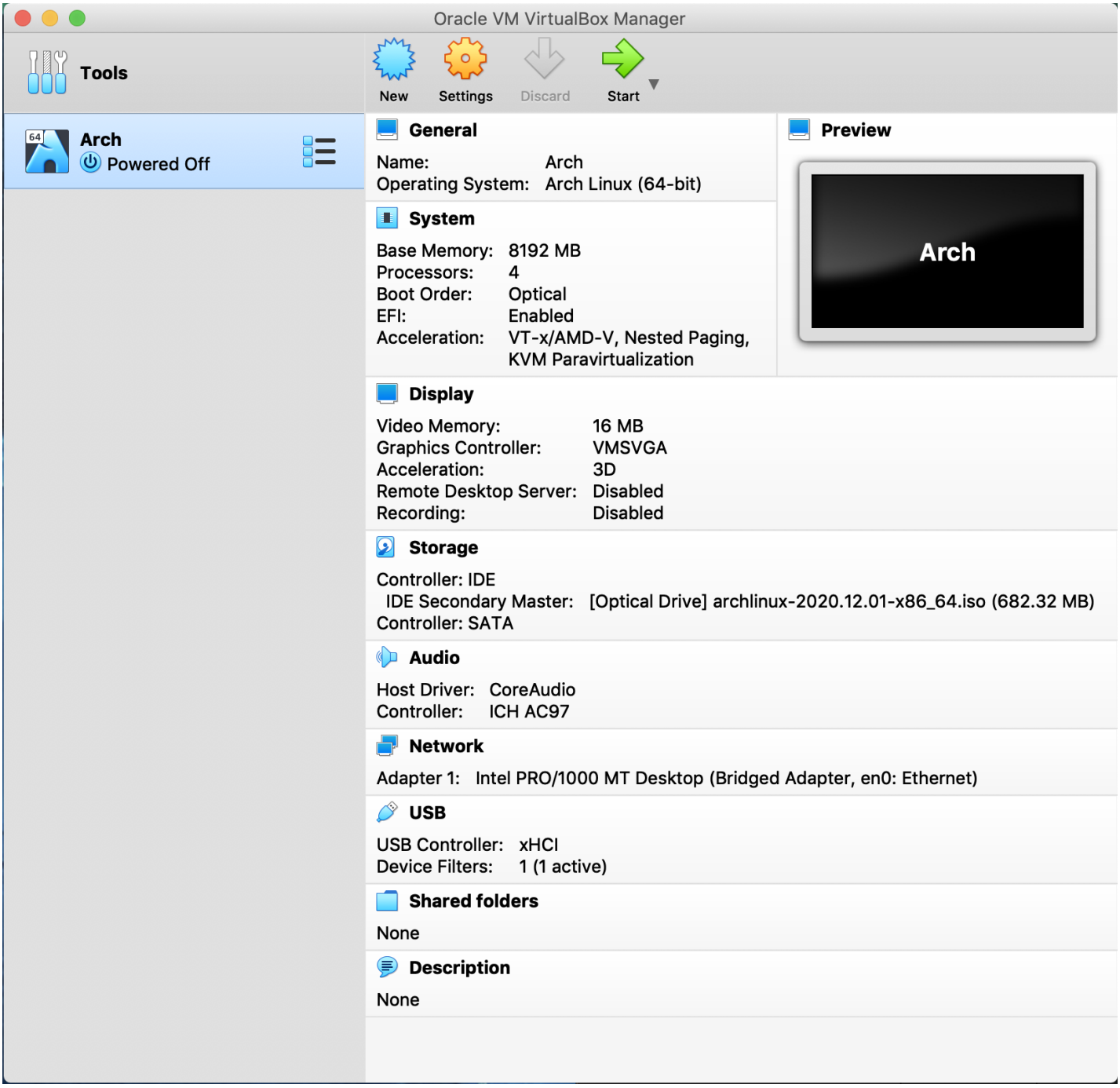

Arch - Storage
General
System
Display
Storage
Audio
Network
Ports
Shared Folders
User Interface
Storage Devices
Controller: IDE
archlinux-2020.12.01-x86_64.iso
Controller: SATA
Attributes
Optical Drive:
IDE Secondary Master
Live CD/DVD
Information
Type: Image
Size: 682.32 MB
Location: /Users/wison/Backup/OS/Arch/arc...
Attached to: Arch
Cancel
OK



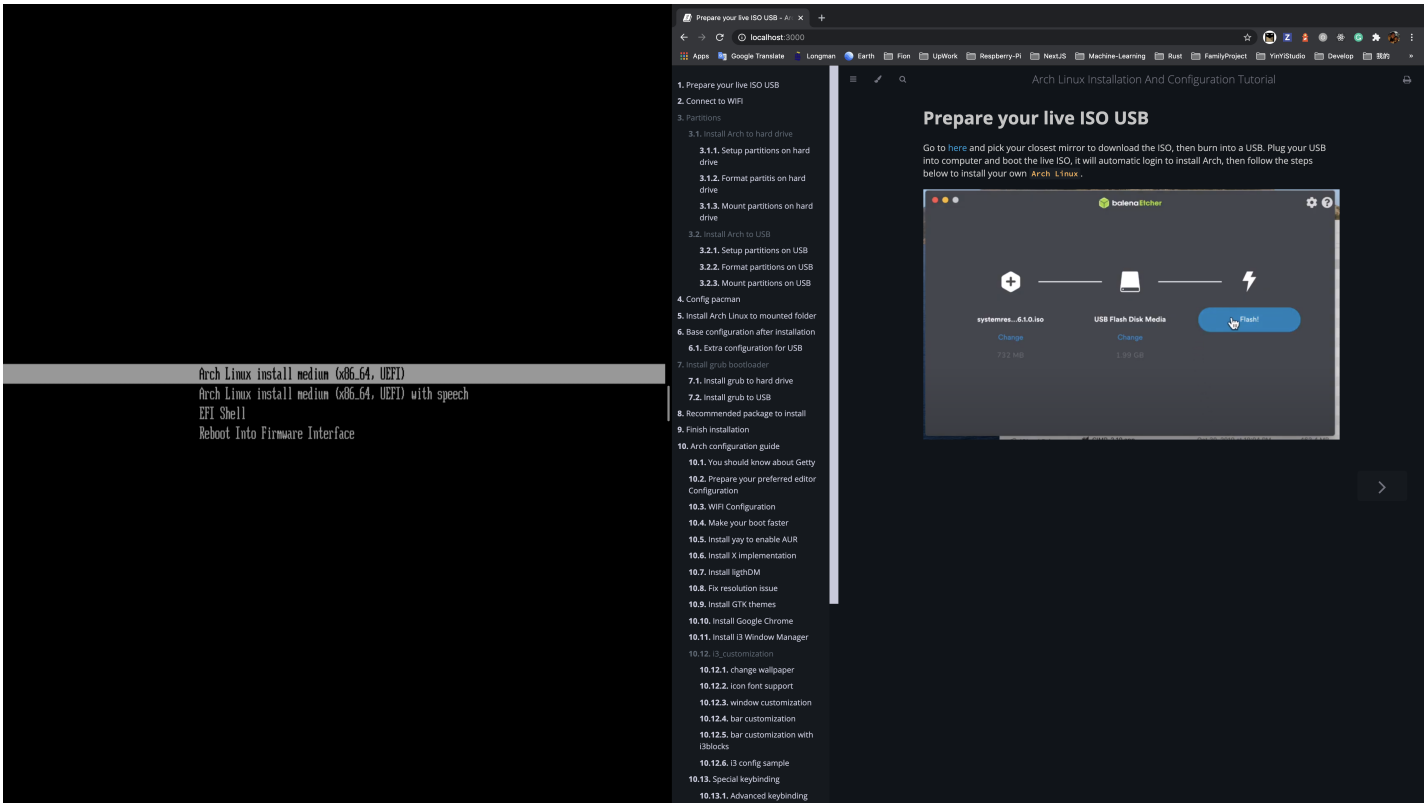
If you want to install to USB, add your USB here:

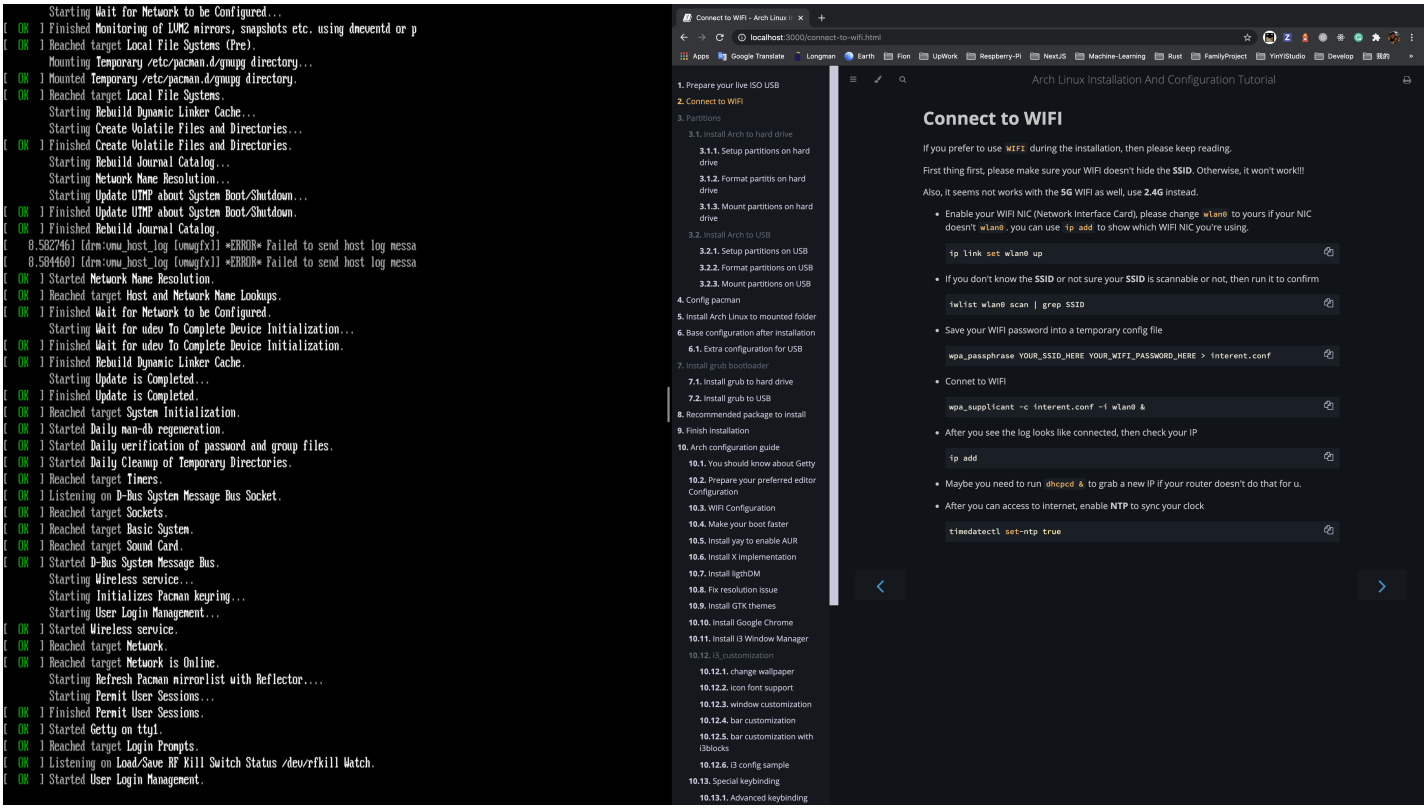


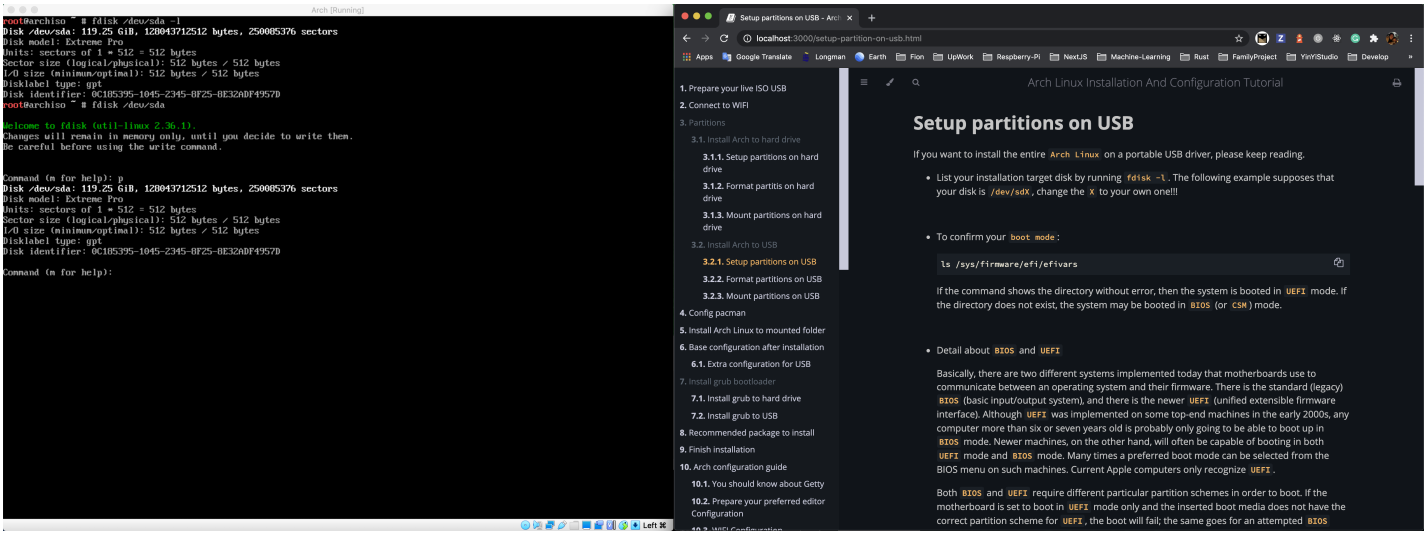
Now, you can run the virtual machine:



Of course, you can open this book side-by-side which is more easy to walk through step-by-step


Arch Linux install medium (x86_64, UEFI)
Arch Linux install medium (x86_64, UEFI) with speech
EFI Shell
Reboot Into Firmware Interface
Prepare your live ISO USB - Ar
localhost:3000
Arch Linux Installation And Configuration Tutorial
Prepare your live ISO USB
Go to here and pick your closest mirror to download the ISO, then burn into a USB. Plug your USB into computer and boot the live ISO, it will automatic login to install Arch, then follow the steps below to install your own Arch Linux.
balenaEtcher
systemres...6.1.0.iso
USB Flash Disk Media
Flash!
Change
Change
732 MB
1.99 GB
1. Prepare your live ISO USB
2. Connect to WiFi
3. Partitions
3.1. Install Arch to hard drive
3.1.1. Setup partitions on hard drive
3.1.2. Format partitis on hard drive
3.1.3. Mount partitions on hard drive
3.2. Install Arch to USB
3.2.1. Setup partitions on USB
3.2.2. Format partitions on USB
3.2.3. Mount partitions on USB
4. Config pacman
5. Install Arch Linux to mounted folder
6. Base configuration after installation
6.1. Extra configuration for USB
7. Install grub bootloader
7.1. Install grub to hard drive
7.2. Install grub to USB
8. Recommended package to install
9. Finish installation
10. Arch configuration guide
10.1. You should know about Getty
10.2. Prepare your preferred editor Configuration
10.3. WIFI Configuration
10.4. Make your boot faster
10.5. Install yay to enable AUR
10.6. Install X implementation
10.7. Install ligthDM
10.8. Fix resolution issue
10.9. Install GTK themes
10.10. Install Google Chrome
10.11. Install i3 Window Manager
10.12. i3_customization
10.12.1. change wallpaper
10.12.2. icon font support
10.12.3. window customization
10.12.4. bar customization
10.12.5. bar customization with i3blocks
10.12.6. i3 config sample
10.13. Special keybinding
10.13.1. Advanced keybinding


Starting Wait for Network to be Configured...
[OK] Finished Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or p
[OK] Reached target Local File Systems (Pre).
Mounting Temporary /etc/pacman.d/gnupg directory...
[OK] Mounted Temporary /etc/pacman.d/gnupg directory.
[OK] Reached target Local File Systems.
Starting Rebuild Dynamic Linker Cache...
Starting Create Volatile Files and Directories...
[OK] Finished Create Volatile Files and Directories.
Starting Rebuild Journal Catalog...
Starting Network Name Resolution...
Starting Update UTMP about System Boot/Shutdown...
[OK] Finished Update UTMP about System Boot/Shutdown.
[OK] Finished Rebuild Journal Catalog.
[8.582746] [drm:vmw_host_log [vmwgfx]] *ERROR* Failed to send host log messa
[8.584460] [drm:vmw_host_log [vmwgfx]] *ERROR* Failed to send host log messa
[OK] Started Network Name Resolution.
[OK] Reached target Host and Network Name Lookups.
[OK] Finished Wait for Network to be Configured.
Starting Wait for udev To Complete Device Initialization...
[OK] Finished Wait for udev To Complete Device Initialization.
[OK] Finished Rebuild Dynamic Linker Cache.
Starting Update is Completed...
[OK] Finished Update is Completed.
[OK] Reached target System Initialization.
[OK] Started Daily man-db regeneration.
[OK] Started Daily verification of password and group files.
[OK] Started Daily Cleanup of Temporary Directories.
[OK] Reached target Timers.
[OK] Listening on D-Bus System Message Bus Socket.
[OK] Reached target Sockets.
[OK] Reached target Basic System.
[OK] Reached target Sound Card.
[OK] Started D-Bus System Message Bus.
Starting Wireless service...
Starting Initializes Pacman keyring...
Starting User Login Management...
[OK] Started Wireless service.
[OK] Reached target Network.
[OK] Reached target Network is Online.
Starting Refresh Pacman mirrorlist with Reflector....
Starting Permit User Sessions...
[OK] Finished Permit User Sessions.
[OK] Started Getty on tty1.
[OK] Reached target Login Prompts.
[OK] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.
[OK] Started User Login Management.
Connect to WiFi - Arch Linux
localhost:3000/connect-to-wifi.html
Arch Linux Installation And Configuration Tutorial
Connect to WIFI
If you prefer to use WIFI during the installation, then please keep reading.
First thing first, please make sure your WIFI doesn't hide the SSID. Otherwise, it won't work!!!
Also, it seems not works with the 5G WIFI as well, use 2.4G instead.
Enable your WIFI NIC (Network Interface Card), please change wlan0 to yours if your NIC doesn't wlan0. you can use ip add to show which WIFI NIC you're using.
ip link set wlan0 up
If you don't know the SSID or not sure your SSID is scannable or not, then run it to confirm
iwlist wlan0 scan | grep SSID
Save your WIFI password into a temporary config file
wpa_passphrase YOUR_SSID_HERE YOUR_WIFI_PASSWORD_HERE > interent.conf
Connet to WIFI
wpa_supplicant -c interent.conf -i wlan0 &
After you see the log looks like connected, then check your IP
ip add
Maybe you need to run dhcpcd & to grab a new IP if your router doesn't do that for u.
After you can access to internet, enable NTP to sync your clock
timedatectl set-ntp true
1. Prepare your live ISO USB
2. Connect to WiFi
3. Partitions
3.1. Install Arch to hard drive
3.1.1. Setup partitions on hard drive
3.1.2. Format partitis on hard drive
3.1.3. Mount partitions on hard drive
3.2. Install Arch to USB
3.2.1. Setup partitions on USB
3.2.2. Format partitions on USB
3.2.3. Mount partitions on USB
4. Config pacman
5. Install Arch Linux to mounted folder
6. Base configuration after installation
6.1. Extra configuration for USB
7. Install grub bootloader
7.1. Install grub to hard drive
7.2. Install grub to USB
8. Recommended package to install
9. Finish installation
10. Arch configuration guide
10.1. You should know about Getty
10.2. Prepare your preferred editor Configuration
10.3. WIFI Configuration
10.4. Make your boot faster
10.5. Install yay to enable AUR
10.6. Install X implementation
10.7. Install ligthDM
10.8. Fix resolution issue
10.9. Install GTK themes
10.10. Install Google Chrome
10.11. Install i3 Window Manager
10.12. i3_customization
10.12.1. change wallpaper
10.12.2. icon font support
10.12.3. window customization
10.12.4. bar customization
10.12.5. bar customization with i3blocks
10.12.6. i3 config sample
10.13. Special keybinding
10.13.1. Advanced keybinding



Connect to WIFI

If you prefer to use `WIFI` during the installation, then please keep reading.

First thing first, please make sure your WIFI doesn't hide the **SSID**. Otherwise, it won't work!!!

Also, the steps below seems not works with the **5G** WIFI if you use encrypted password, use **2.4G** instead.

For the `5G` network and hidden SSID, it works after you install `netctl`, plz read the details in `WIFI Configuration` chapter.

- Enable your WIFI NIC (Network Interface Card), please change `wlan0` to yours if your NIC doesn't `wlan0`. you can use `ip add` to show which WIFI NIC you're using.

```
ip link set wlan0 up
```

- If you don't know the **SSID** or not sure your **SSID** is scannable or not, then run it to confirm

```
iwlist wlan0 scan | grep SSID
```

- Save your WIFI password into a temporary config file

```
wpa_passphrase YOUR_SSID_HERE YOUR_WIFI_PASSWORD_HERE > interent.conf
```

- Connet to WIFI

```
wpa_supplicant -c interent.conf -i wlan0 &
```

- After you see the log looks like connected, then check your IP

```
ip add
```

- Maybe you need to run `dhcpcd &` to grab a new IP if your router doesn't do that for u.
- After you can access to internet, enable **NTP** to sync your clock

```
timedatectl set-ntp true
```

Setup partitions on hard drive

If you want to install the entire `Arch Linux` on any computer hard drive, please keep reading.

- List your installation target disk by running `fdisk -l`. The following example supposes that your disk is `/dev/sdX`, change the `X` to your own one!!!

- To confirm your `boot mode`:

```
ls /sys/firmware/efi/efivars
```

If the command shows the directory without error, then the system is booted in `UEFI` mode. If the directory does not exist, the system may be booted in `BIOS`

(or `CSM`) mode.

- Detail about `BIOS` and `UEFI`

  Basically, there are two different systems implemented today that motherboards use to communicate between an operating system and their firmware. There is the standard (legacy) `BIOS` (basic input/output system), and there is the newer `UEFI` (unified extensible firmware interface). Although `UEFI` was implemented on some top-end machines in the early 2000s, any computer more than six or seven years old is probably only going to be able to boot up in `BIOS` mode. Newer machines, on the other hand, will often be capable of booting in both `UEFI` mode and `BIOS` mode. Many times a preferred boot mode can be selected from the BIOS menu on such machines. Current Apple computers only recognize `UEFI`.

  Both `BIOS` and `UEFI` require different particular partition schemes in order to boot. If the motherboard is set to boot in `UEFI` mode only and the inserted boot media does not have the correct partition scheme for `UEFI`, the boot will fail; the same goes for an attempted `BIOS` boot. This is one place I believe some of the USB installation guides out there fail: they often only describe how to create a USB boot device that uses only one mode. It is possible, however, to setup a USB drive that will have a partition scheme allowing it to boot in both modes and still use the same persistent installation of Linux. This guide will setup such a scheme in the partitioning and formatting sections below.

  On most newer machines, you will be presented with the option to boot in either `BIOS` or `UEFI` mode from your bootable USB. This means the machine recognizes the `UEFI` boot media, but it does not always mean the machine is actually set to boot in `UEFI` mode, and selecting the `UEFI` boot option may fail. Selecting a mode that the motherboard is not set to boot in will not damage anything or touch any of the other drives on the machine, the boot will simply fail and one can reboot in the other mode. The USB stick created with this guide has been able to boot on every (about a dozen) desktop and laptop, new and old, `BIOS` and `UEFI`, machine that I have tried it on.

- Follow the steps below to setup your Arch Linux partition:
  - Run `fdisk /dev/sdX`
  - Delete all existing partitions

```
# First, make sure you use `d` to delete all existing
partitions!!!
# First, make sure you use `d` to delete all existing
partitions!!!
# First, make sure you use `d` to delete all existing
partitions!!!
`d` // Untill all partitions are be deleted!
```

  - Create new partition table

```
# As I confirmed my boot mode is **`UEFI`**, then I should use
`GPT` partition.
# Press `g` to create a new empty `GPT` partition table, it will
remove all your exists partitions.
# After pressing `g`, you should be able to see something like
below:
#
# "Created a new GPT dislabel (GUID: xxxxxxxxx)."
`g`
```

  - Create `512MB` ESP (EFI System Partition) which will be mounted to `/boot` and hold the bootloader

```
`n`
`1` or just `Enter` to use partition no `1`
`Enter` to use the default start sector
`+512M` to set this partition size to `512MB`
# If success, you should see `Created a new partition 1 of type
`Linux filesystem` and of size 512 MiB.
```

  - Create SWAP partition:

```
`n`
`3`, set this partition no to `3` rather than `2`, then when you
print the partition table, it shows as the last partition which
is more easy to read.
`Enter` to use the default start sector
`+8G` to use 8GB as SWAP, usually, should set to equal to your
RAM amout or 1.5 time of  your RAM amount.
# If success, you should see `Created a new partition 3 of type
`Linux filesystem` and of size 8 MiB.
```

- Finally, create the Linux root partition which will be mount to `/root`

```
`n`
`Enter` to use partition no `2`
`Enter` to use the default start sector
`Enter` to use all the left spaces
# If success, you should see `Created a new partition 2 of type
`Linux filesystem` and of size XXXX MiB.
```

- Then you can print the partition table by pressing `p` to have a look if you want
- The last step is press `w` to write all changes into the disk!!!

```
w
```

Format partitions on hard drive

```
# The boot partition has to be `FAT` format
mkfs.fat -F32 /dev/sdX1

# The root partition, we use `EXT4` format
mkfs.ext4 /dev/sdX2

# Swap partition
mkswap /dev/sdX3

# Enable SWAP partition
swapon /dev/sdX3
```

Mount partitions on hard drive

Before installing `Arch Liunx` to the hard drive, you need to mount the `Arch Linux root partition` and `Arch Linux boot partition`, then you can use `pacmanc` to install the newer Linux filesystem to the mounted folder.

You can run `fdisk -l /dev/sdX` to print the partition table again if you forgot.

```
mount /dev/sdX2 /mnt
mkdir /mnt/boot
mount /dev/sdX1 /mnt/boot

# You can `ls` to confirm
ls -lht /mnt/boot
ls -lht /mnt

# You can run `df -Th` to confirm both partitions are using the correct
format before you do the real installation.
df -Th | grep \/mnt
/dev/sdX2   ext4    /mnt
/dev/sdX1   vfat    /mnt/boot
```

Setup partitions on USB

If you want to install the entire `Arch Linux` on a portable USB driver, please keep reading.

- List your installation target disk by running `fdisk -l`. The following example supposes that your disk is `/dev/sdX`, change the `X` to your own one!!!

- To confirm your `boot mode`:

  ```
  ls /sys/firmware/efi/efivars
  ```

  If the command shows the directory without error, then the system is booted in **`UEFI`** mode. If the directory does not exist, the system may be booted in `BIOS` (or `CSM`) mode.

- Detail about **`BIOS`** and **`UEFI`**

  Basically, there are two different systems implemented today that motherboards use to communicate between an operating system and their firmware. There is the standard (legacy) **`BIOS`** (basic input/output system), and there is the newer **`UEFI`** (unified extensible firmware interface). Although **`UEFI`** was implemented on some top-end machines in the early 2000s, any computer more than six or seven years old is probably only going to be able to boot up in **`BIOS`** mode. Newer machines, on the other hand, will often be capable of booting in both **`UEFI`** mode and **`BIOS`** mode. Many times a preferred boot mode can be selected from the BIOS menu on such machines. Current Apple computers only recognize **`UEFI`**.

  Both **`BIOS`** and **`UEFI`** require different particular partition schemes in order to boot. If the motherboard is set to boot in **`UEFI`** mode only and the inserted boot media does not have the correct partition scheme for **`UEFI`**, the boot will fail; the same goes for an attempted **`BIOS`** boot. This is one place I believe some of the USB installation guides out there fail: they often only describe how to create a USB boot device that uses only one mode. It is possible, however, to setup a USB drive that will have a partition scheme allowing it to boot in both modes and still use the same persistent installation of Linux. This guide will setup such a scheme in the partitioning and formatting sections below.

  On most newer machines, you will be presented with the option to boot in either **`BIOS`** or **`UEFI`** mode from your bootable USB. This means the machine recognizes the **`UEFI`** boot media, but it does not always mean the machine is

actually set to boot in **`UEFI`** mode, and selecting the **`UEFI`** boot option may fail. Selecting a mode that the motherboard is not set to boot in will not damage anything or touch any of the other drives on the machine, the boot will simply fail and one can reboot in the other mode. The USB stick created with this guide has been able to boot on every (about a dozen) desktop and laptop, new and old, **`BIOS`** and **`UEFI`**, machine that I have tried it on.

- Follow the steps below to setup your Arch Linux partition:

  We will setup a USB which compitlabe with both **`BIOS`** and **`UEFI`**!!!

  - Run `fdisk /dev/sdX`
  - Delete all existing partitions

    ```
    # First, make sure you use `d` to delete all existing
    partitions!!!
    # First, make sure you use `d` to delete all existing
    partitions!!!
    # First, make sure you use `d` to delete all existing
    partitions!!!
    `d` // Untill all partitions are be deleted!
    ```

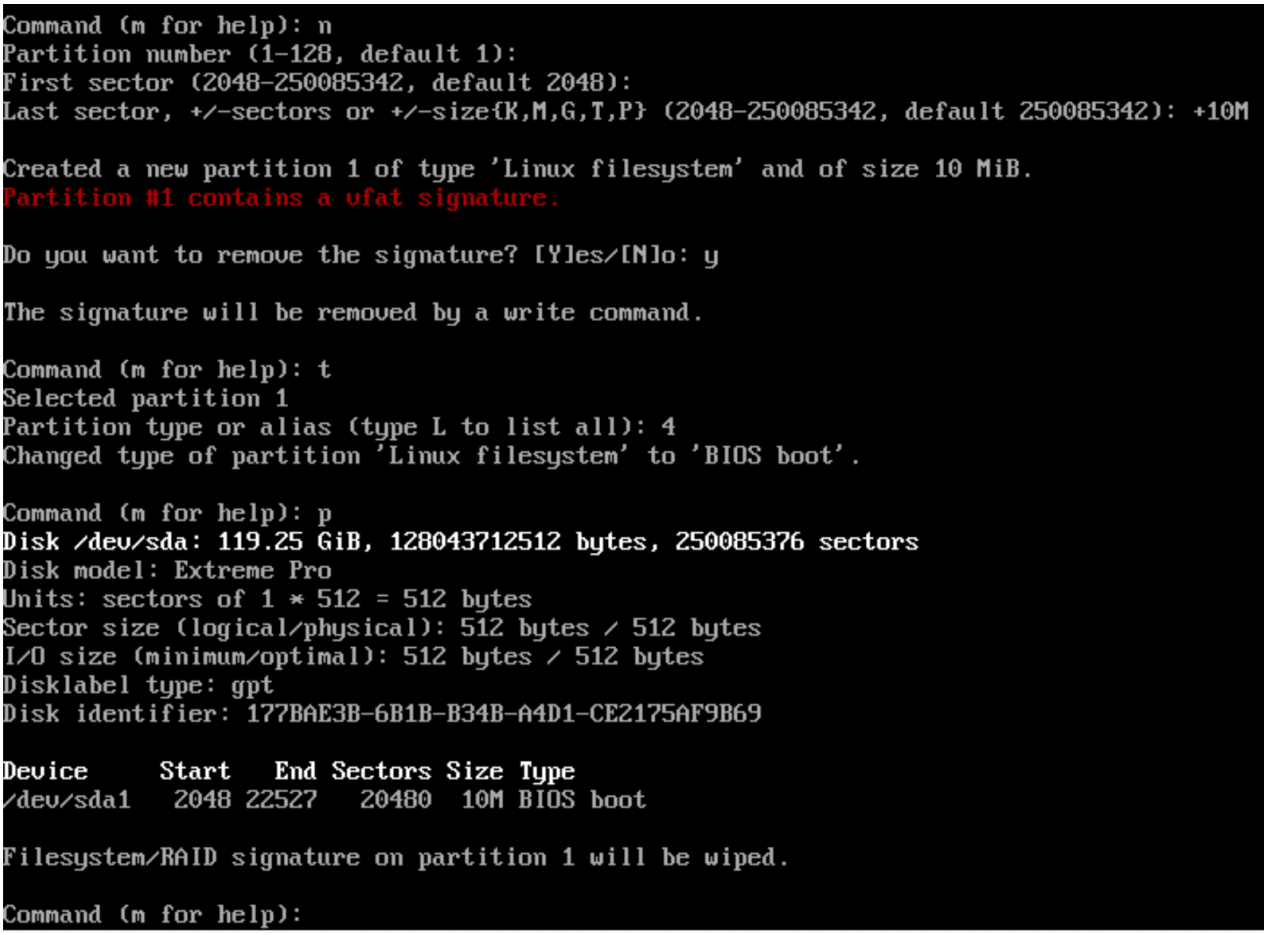
  - Create new partition table

    ```
    # As I confirmed my boot mode is **`UEFI`**, then I should use
    `GPT` partition.
    # Press `g` to create a new empty `GPT` partition table, it will
    remove all
    # your exists partitions.
    # After pressing `g`, you should be able to see something like
    below:
    #
    # "Created a new GPT dislabel (GUID: xxxxxxxxx)."
    `g`
    ```

```
root@archiso ~ # fdisk /dev/sda

Welcome to fdisk (util-linux 2.36.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d
Partition number (1-3, default 3):

Partition 3 has been deleted.

Command (m for help): d
Partition number (1,2, default 2):

Partition 2 has been deleted.

Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): p
Disk /dev/sda: 119.25 GiB, 128043712512 bytes, 250085376 sectors
Disk model: Extreme Pro
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 0C185395-1045-2345-8F25-8E32ADF4957D

Command (m for help): g
Created a new GPT disklabel (GUID: 177BAE3B-6B1B-B34B-A4D1-CE2175AF9B69).

Command (m for help): p
Disk /dev/sda: 119.25 GiB, 128043712512 bytes, 250085376 sectors
Disk model: Extreme Pro
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 177BAE3B-6B1B-B34B-A4D1-CE2175AF9B69

Command (m for help):
```

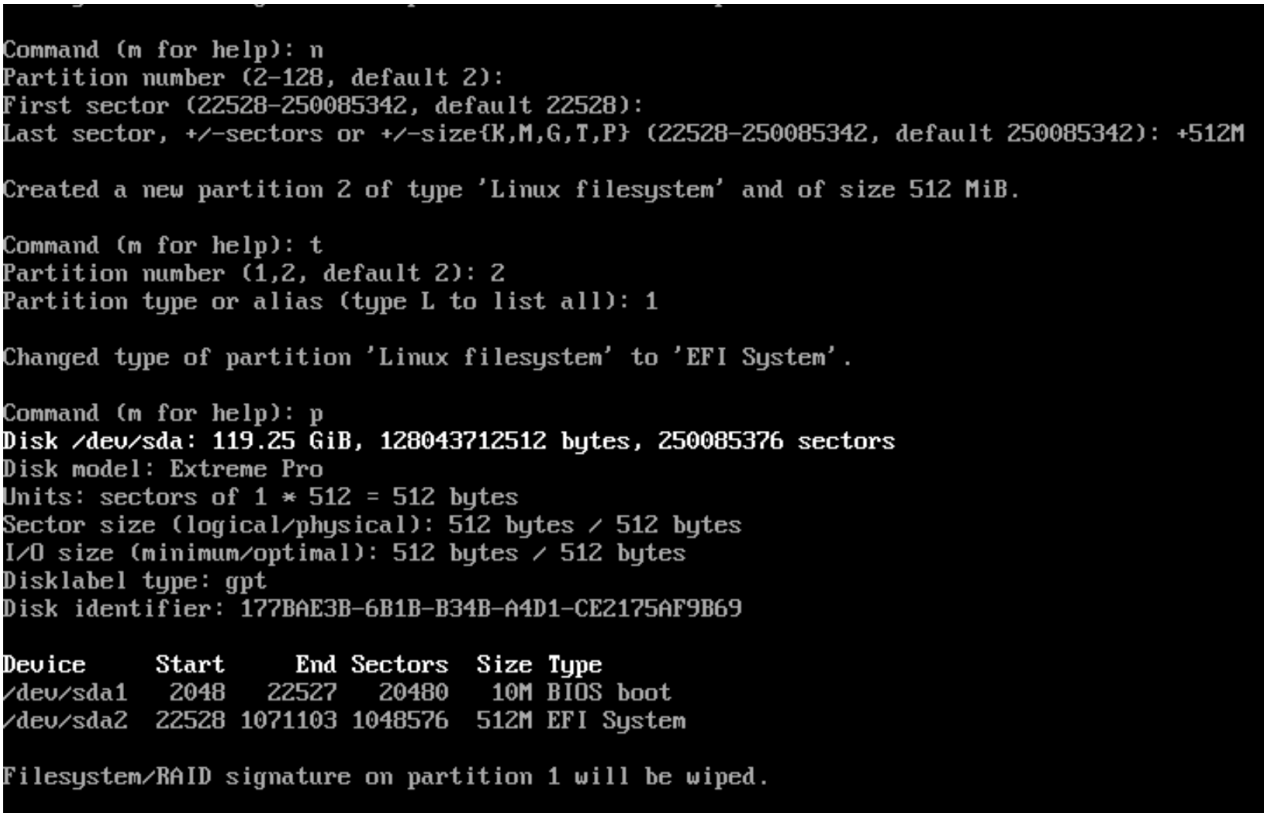
- Create `10MB` **MBR** partition

```
`n`
`1` or just `Enter` to use partition no `1`
`Enter` to use the default start sector
`+10M` to set this partition size to `10M`
#
# If success, you should see `Created a new partition 1 of type
# `Linux filesystem` and of size 10 MiB.

# If it asks you "Partition #X contains a YYYY signature, Do you
want to
# remove the signature", then press 'Y' to remove the previous
partition
# signature.
#
# We need to change the partition type from `Linux filesystem` to
`BIOS BOOT`.
# Before that you can press `l` to list all supported partition
types:
`t`
`4`,

# Finally, press `p` to confirm the result, it should look like
this
#
# /dev/sdX1   (ignore the sector numbers there) 10M BIOS boot
```



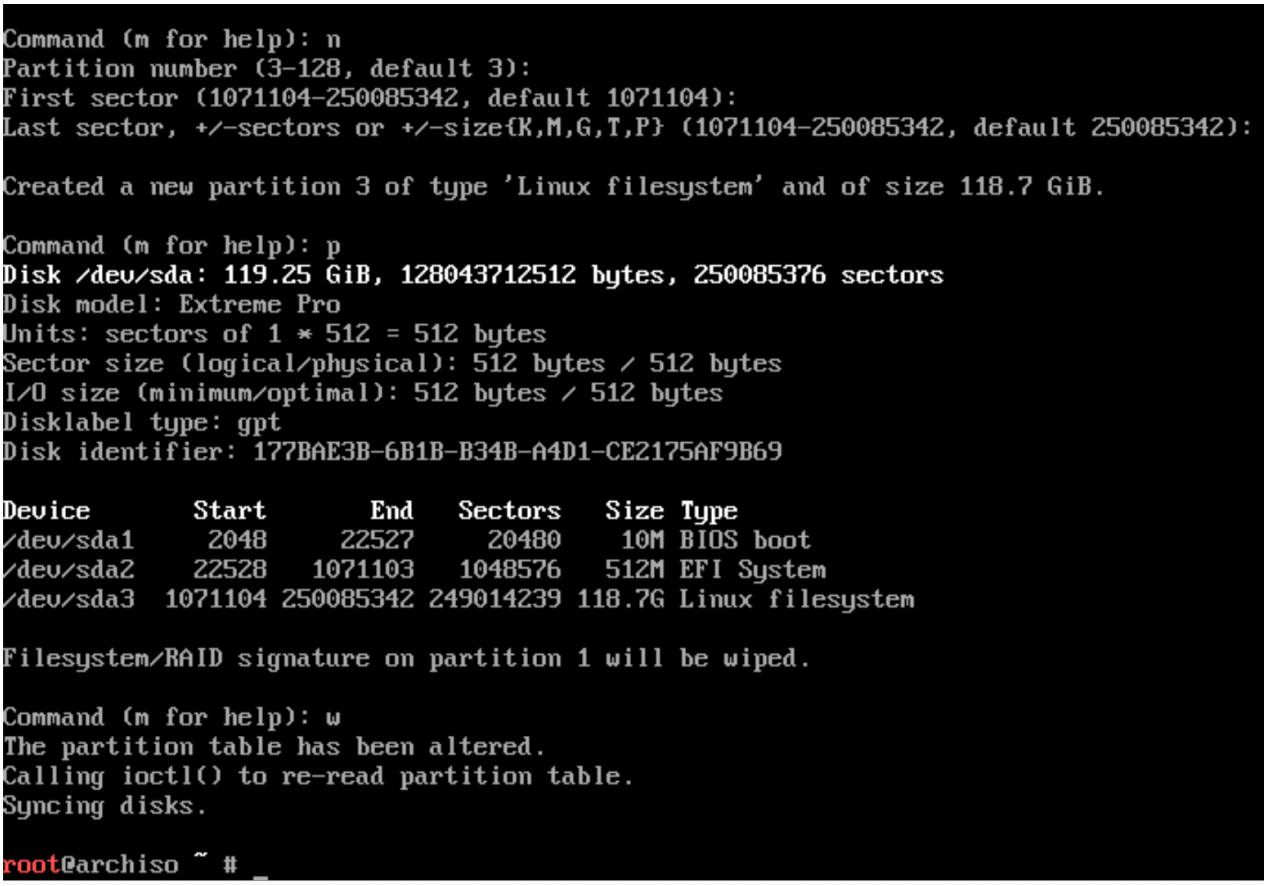
- Create `512MB` ESP (EFI System Partition) which will be mounted to `/boot` and hold the bootloader

```
`n`
`2` or just `Enter` to use partition no `2`
`Enter` to use the default start sector
`+512M` to set this partition size to `512MB`
#
# If success, you should see `Created a new partition 2 of type
# `Linux filesystem` and of size 512 MiB.

# We need to change the partition type from `Linux filesystem` to
# `EFI System`. Before that you can press `l` to list all
supported
# partition types:
`t`
`2` to make sure select the 512MB partition
`1` which means `EFI System`

# Finally, press `p` to confirm the result, it should look like
this
#
# /dev/sdX1   (ignore the sector numbers there) 10M  BIOS boot
# /dev/sdX2   (ignore the sector numbers there) 512M EFI System
```



- Finally, create the linux root partition which will be mounted to `/` and hold the entire Linux system

```
`n`
`3` or just `Enter` to use partition no `3`
`Enter` to use the default start sector
`Enter` to use all the left spaces
# If success, you should see `Created a new partition 3 of type
# `Linux filesystem` and of size XXXX GiB.

# Finally, press `p` to confirm the result, it should look like
this
#
# /dev/sdX1   (ignore the sector numbers there) 10M  BIOS boot
# /dev/sdX2   (ignore the sector numbers there) 512M EFI System
# /dev/sdX3   (ignore the sector numbers there) XXXG Linux
filesystem
```



- The last step is press `w` to write all changes into the disk!!!

```
`w`
```

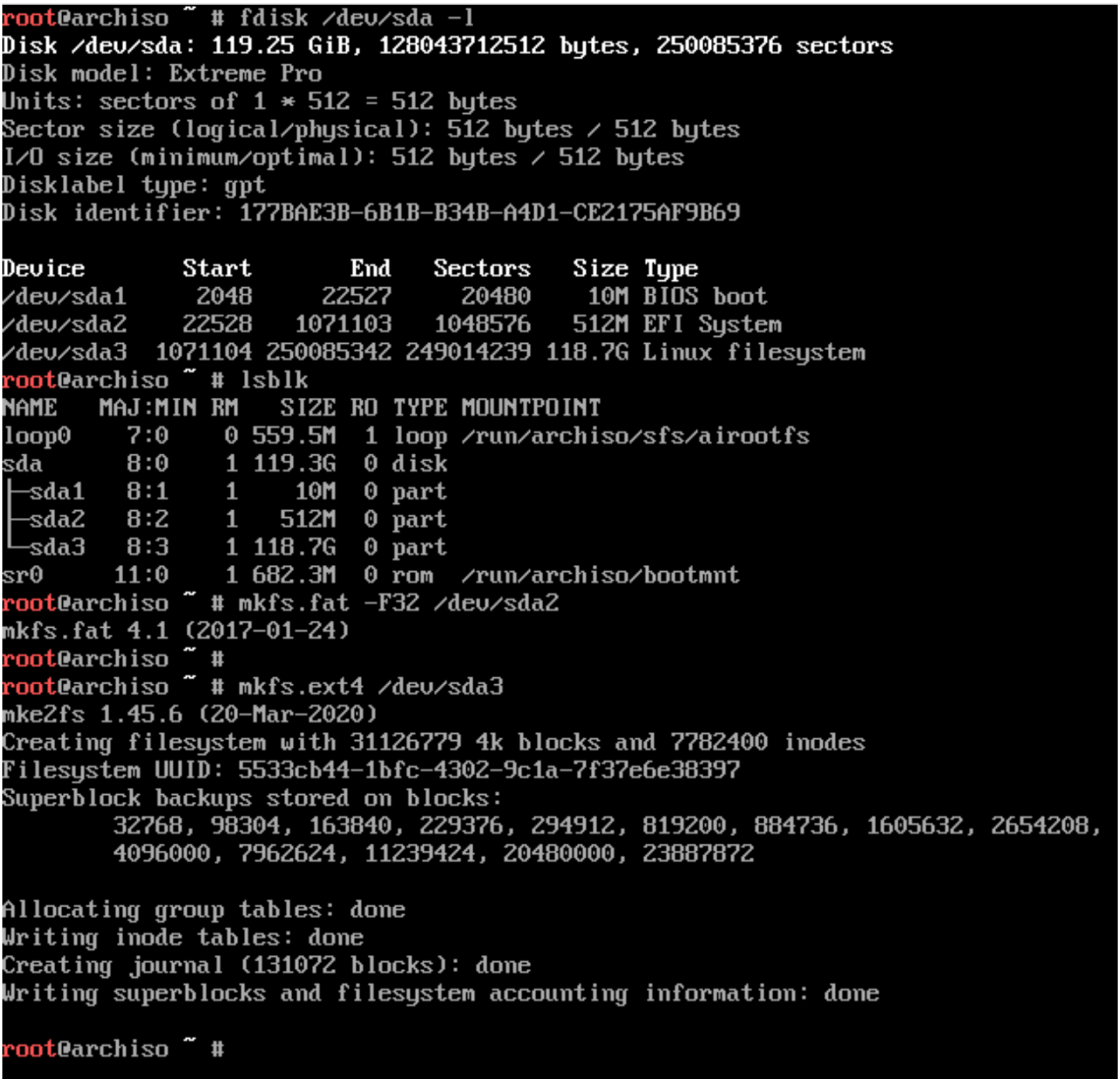
Format partitions on USB

Do not format the `/dev/sdX1` partition. This is the `BIOS/MBR` partion!!!

Do not format the `/dev/sdX1` partition. This is the `BIOS/MBR` partion!!!

Do not format the `/dev/sdX1` partition. This is the `BIOS/MBR` partion!!!

```
# List block device, can help you to type correctly.
lsblk

# The `EFI` boot partition has to be `FAT` format
mkfs.fat -F32 /dev/sdX2

# The root partition, we use `EXT4` format
mkfs.ext4 /dev/sdX3
```

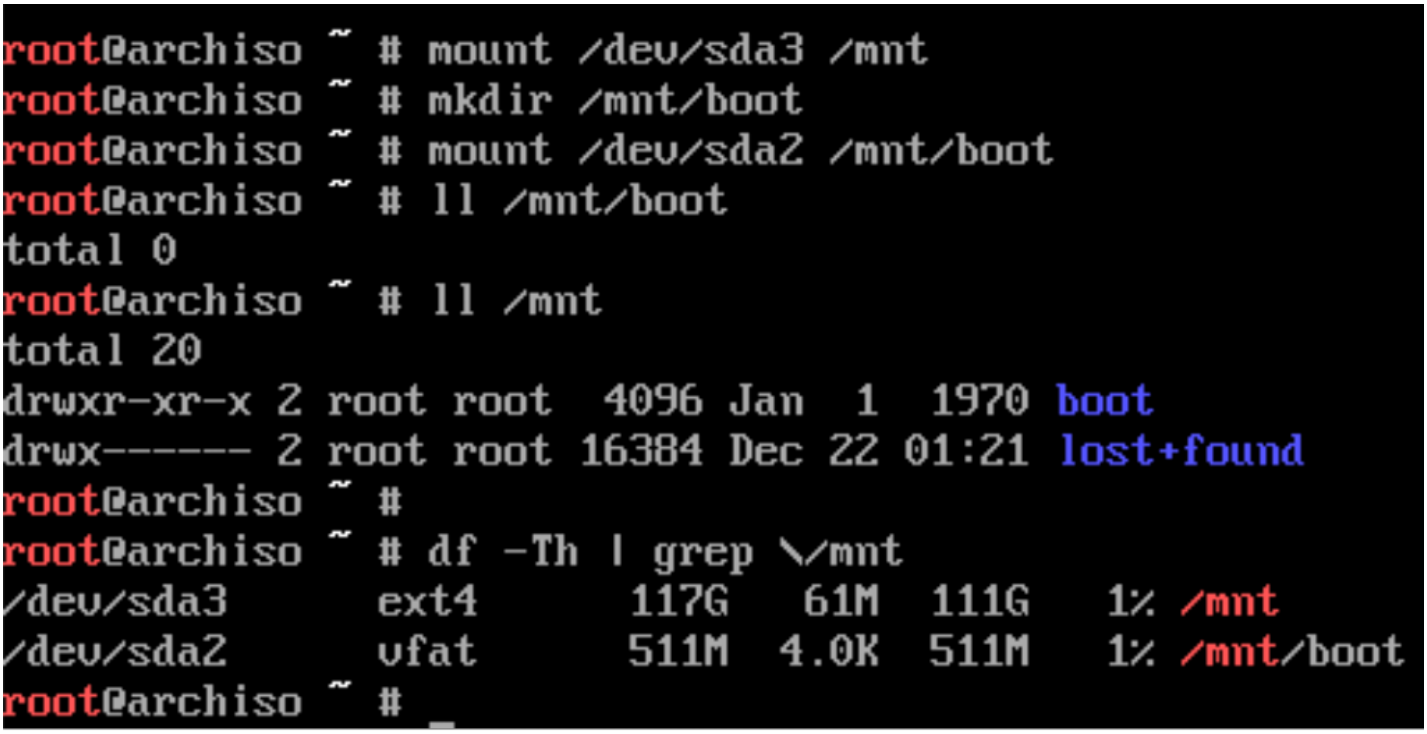


Mount partitions on USB

Before installing `Arch Liunx` to the USB, you need to mount the `Arch Linux root partition` and `Arch Linux boot partition`, then you can use `pacmanc` to install the newer Linux filesystem to the mounted folder.

You can run `fdisk -l /dev/sdX` to print the partition table again if you forgot.

```
mount /dev/sdX3 /mnt
mkdir /mnt/boot
mount /dev/sdX2 /mnt/boot

# You can `ls` to confirm
ls -lht /mnt/boot
ls -lht /mnt

# You can run `df -Th` to confirm both partitions are using the correct
format before you do the real installation.
df -Th | grep \/mnt
/dev/sdX3   ext4    /mnt
/dev/sdX2   vfat    /mnt/boot
```



Config `pacman`

- Enable the `Color` output

```
vim /etc/pacman.conf

# Search and enable `Color` line, save and exit
```

- Setup mirror

  Open mirror list to search your native country and click on it, you should be able to see the **Mirror URL**. In this tutorial, I just pick **New Zealand** as an example.

  It looks like this:

```
# Make sure to copy your country mirror URL here
http://mirror.fsmg.org.nz/archlinux/
https://mirror.fsmg.org.nz/archlinux/
http://mirror.smith.geek.nz/archlinux/
https://mirror.smith.geek.nz/archlinux/
```

  Then `vim /etc/pacman.d/mirrorlist` and replace all contents with the below settings:

```
# Make sure change to your country mirror URL!!!
# The syntax looks like below:
#
# Server = YOUR_MIRROR_URL/$repo/os/$arch

# New Zealand
Server = http://mirror.fsmg.org.nz/archlinux/$repo/os/$arch
Server = https://mirror.fsmg.org.nz/archlinux/$repo/os/$arch
Server = http://mirror.smith.geek.nz/archlinux/$repo/os/$arch
Server = https://mirror.smith.geek.nz/archlinux/$repo/os/$arch
```

Install Arch Linux to mounted folder

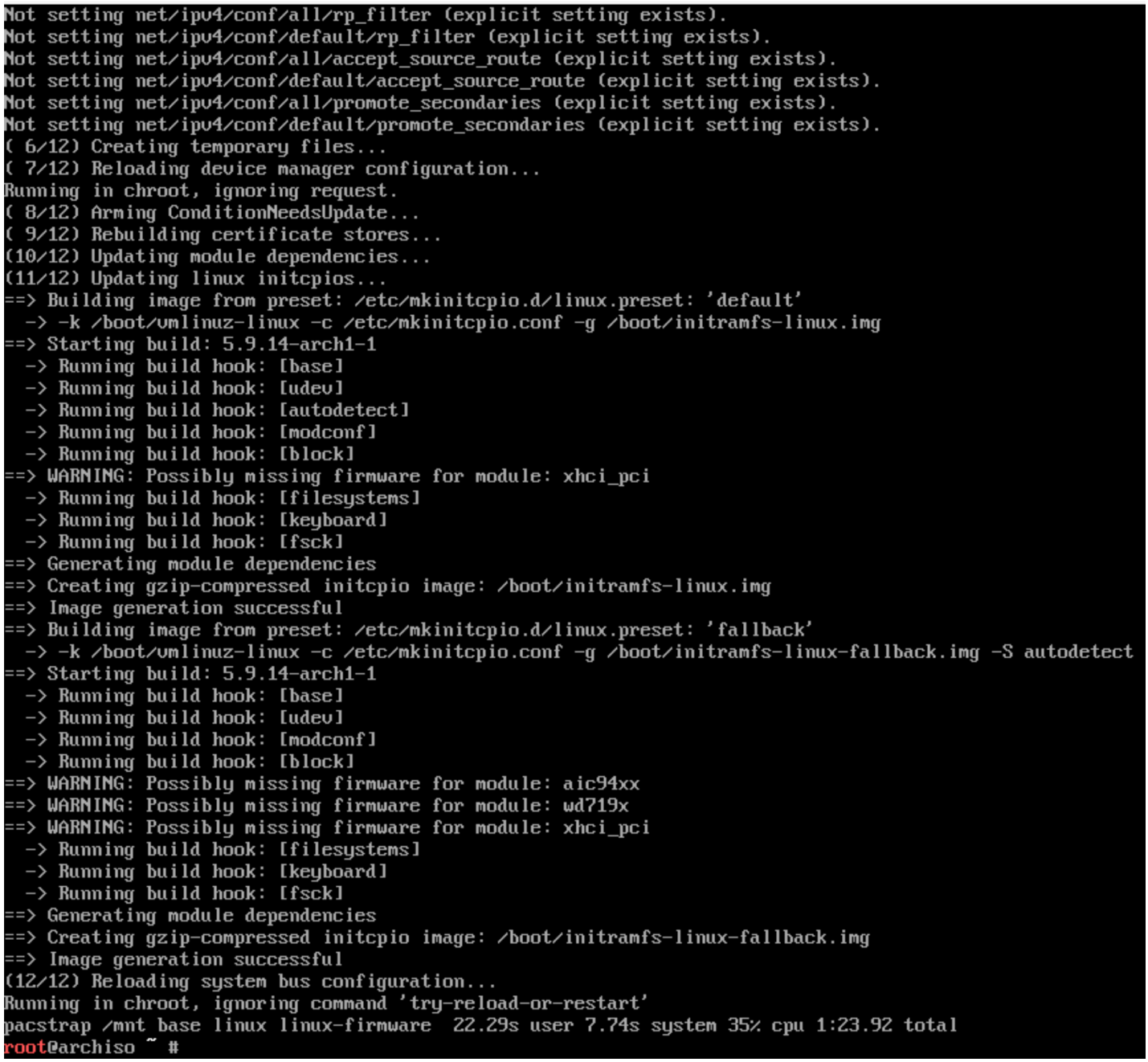
Keep in mind that:

The current `Arch Linux` you've already login is boot from the `Live ISO`, it just a `tool` for you to install another brand new `Arch Linux` to your hard drive or USB which located at `/mnt` and `/mnt/boot`.

That means there will be an entirely brand new `Arch Linux` in your `/mnt` folder!!!

Now, use the `pacstrap(8)` script to install the base package, Linux kernel and firmware for common hardware:

```
# `/mnt` is where all packages will be installed to!!!
pacstrap /mnt base linux linux-firmware
```


Not setting net/ipv4/conf/all/rp_filter (explicit setting exists).
Not setting net/ipv4/conf/default/rp_filter (explicit setting exists).
Not setting net/ipv4/conf/all/accept_source_route (explicit setting exists).
Not setting net/ipv4/conf/default/accept_source_route (explicit setting exists).
Not setting net/ipv4/conf/all/promote_secondaries (explicit setting exists).
Not setting net/ipv4/conf/default/promote_secondaries (explicit setting exists).
(6/12) Creating temporary files...
(7/12) Reloading device manager configuration...
Running in chroot, ignoring request.
(8/12) Arming ConditionNeedsUpdate...
(9/12) Rebuilding certificate stores...
(10/12) Updating module dependencies...
(11/12) Updating linux initcpios...
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux.img
==> Starting build: 5.9.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [autodetect]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'fallback'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux-fallback.img -S autodetect
==> Starting build: 5.9.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: aic94xx
==> WARNING: Possibly missing firmware for module: wd719x
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux-fallback.img
==> Image generation successful
(12/12) Reloading system bus configuration...
Running in chroot, ignoring command 'try-reload-or-restart'
pacstrap /mnt base linux linux-firmware 22.29s user 7.74s system 35% cpu 1:23.92 total
root@archiso ~ #

```
root@archiso ~ # ll /mnt
total 72
lrwxrwxrwx  1 root root     7 Sep  2 22:30 bin -> usr/bin
drwxr-xr-x  2 root root  4096 Jan  1  1970 boot
drwxr-xr-x  2 root root  4096 Dec 22 01:30 dev
drwxr-xr-x 34 root root  4096 Dec 22 01:31 etc
drwxr-xr-x  2 root root  4096 Sep  2 22:30 home
lrwxrwxrwx  1 root root     7 Sep  2 22:30 lib -> usr/lib
lrwxrwxrwx  1 root root     7 Sep  2 22:30 lib64 -> usr/lib
drwx------  2 root root 16384 Dec 22 01:21 lost+found
drwxr-xr-x  2 root root  4096 Sep  2 22:30 mnt
drwxr-xr-x  2 root root  4096 Sep  2 22:30 opt
dr-xr-xr-x  2 root root  4096 Dec 22 01:30 proc
drwxr-x---  3 root root  4096 Dec 22 01:31 root
drwxr-xr-x  2 root root  4096 Dec 22 01:30 run
lrwxrwxrwx  1 root root     7 Sep  2 22:30 sbin -> usr/bin
drwxr-xr-x  4 root root  4096 Dec 22 01:31 srv
dr-xr-xr-x  2 root root  4096 Dec 22 01:30 sys
drwxrwxrwt  2 root root  4096 Dec 22 01:30 tmp
drwxr-xr-x  8 root root  4096 Dec 22 01:31 usr
drwxr-xr-x 12 root root  4096 Dec 22 01:31 var
root@archiso ~ #
```

After that, generate the `fstab` file, the `-U` flag means use **UUID** to identify the device.

As if you're installing to the USB, then the USB will be plugged into a different machine which causes different `/dev/XXX` labels.

```
genfstab -U /mnt >> /mnt/etc/fstab
```

```
root@archiso ~ # genfstab -U /mnt >> /mnt/etc/fstab
root@archiso ~ #
root@archiso ~ # cat /mnt/etc/fstab
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/sda3
UUID=5533cb44-1bfc-4302-9c1a-7f37e6e38397       /               ext4            rw,relatime     0 1

# /dev/sda2
UUID=9F6B-FA2D          /boot           vfat            rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,short
name=mixed,utf8,errors=remount-ro       0 2
```

Base configuration after installation

Keep in mind that:

The current `Arch Linux` you've already login is boot from the `Live ISO`, it just a `tool` for you to install another brand new `Arch Linux` to your hard drive or USB which located at `/mnt` and `/mnt/boot`.

Right now, you need to config the newer `Arch Linux` you just installed, that's why you need to use `arch-chroot` to change linux root environment into the `/mnt` folder!!!

After running `arch-chroot` command, you will be inside the newly installed `Arch Linux`.

- Change root into the new `Arch Linux`

```
arch-chroot /mnt
```

- Choose timezone

```
# ln -sf /usr/share/zoneinfo/YOUR_REGION/YOUR_CITY /etc/localtime

# For example:
ln -sf /usr/share/zoneinfo/Pacific/Auckland /etc/localtime
```

- Generate `/etc/adjtime`

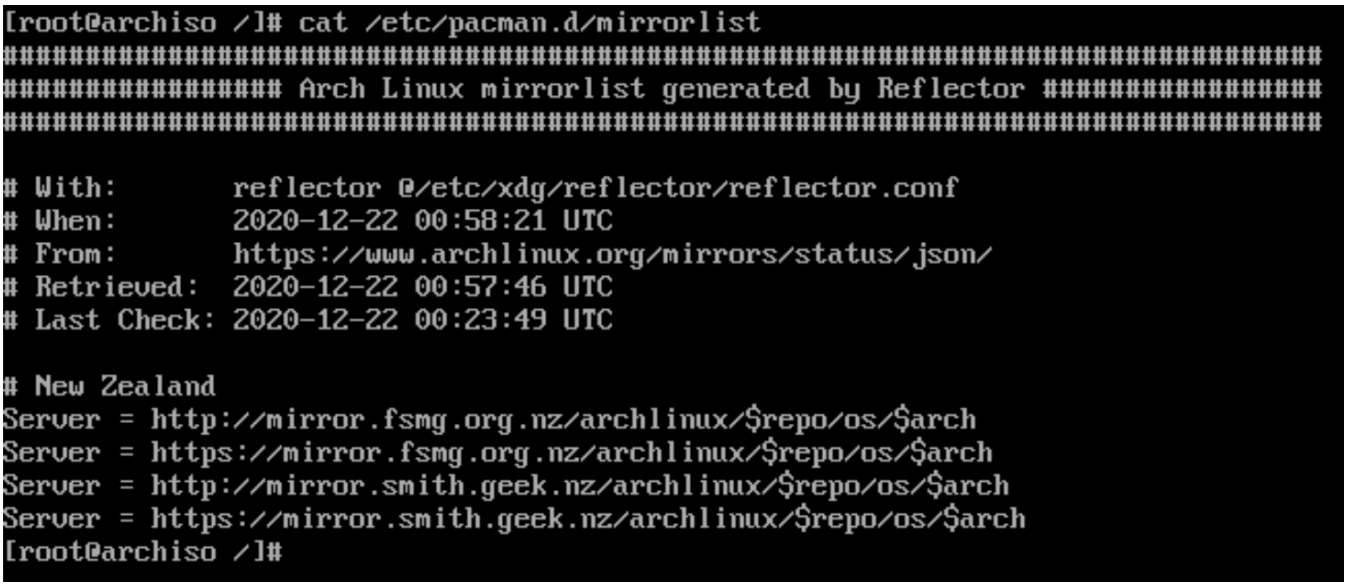
```
hwclock --systohc
```

- Sync `pacman` and install `vim`

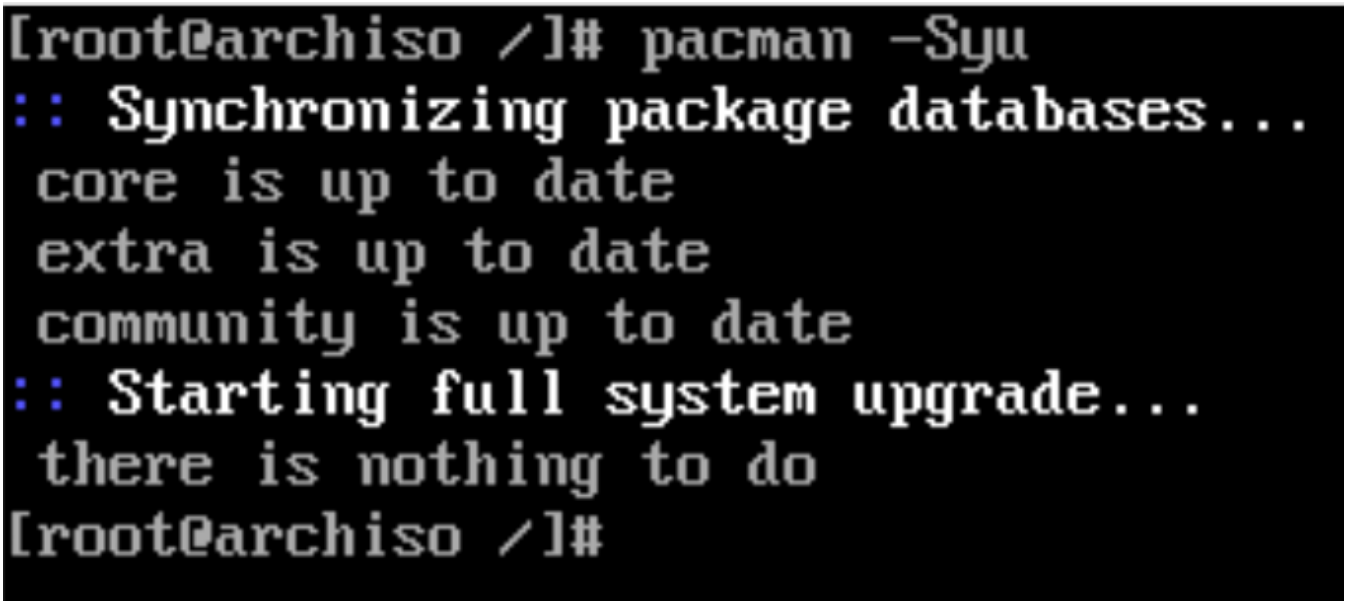
  Before we continue, we need to install `vim` for the newer installed `Arch Linux`, then we can edit any configuration file.

  In the newer installed `Arch Linux`, the `/etc/pacman.d/mirrorlist` should be copied from the `Live Arch`, we can confirm that by running:

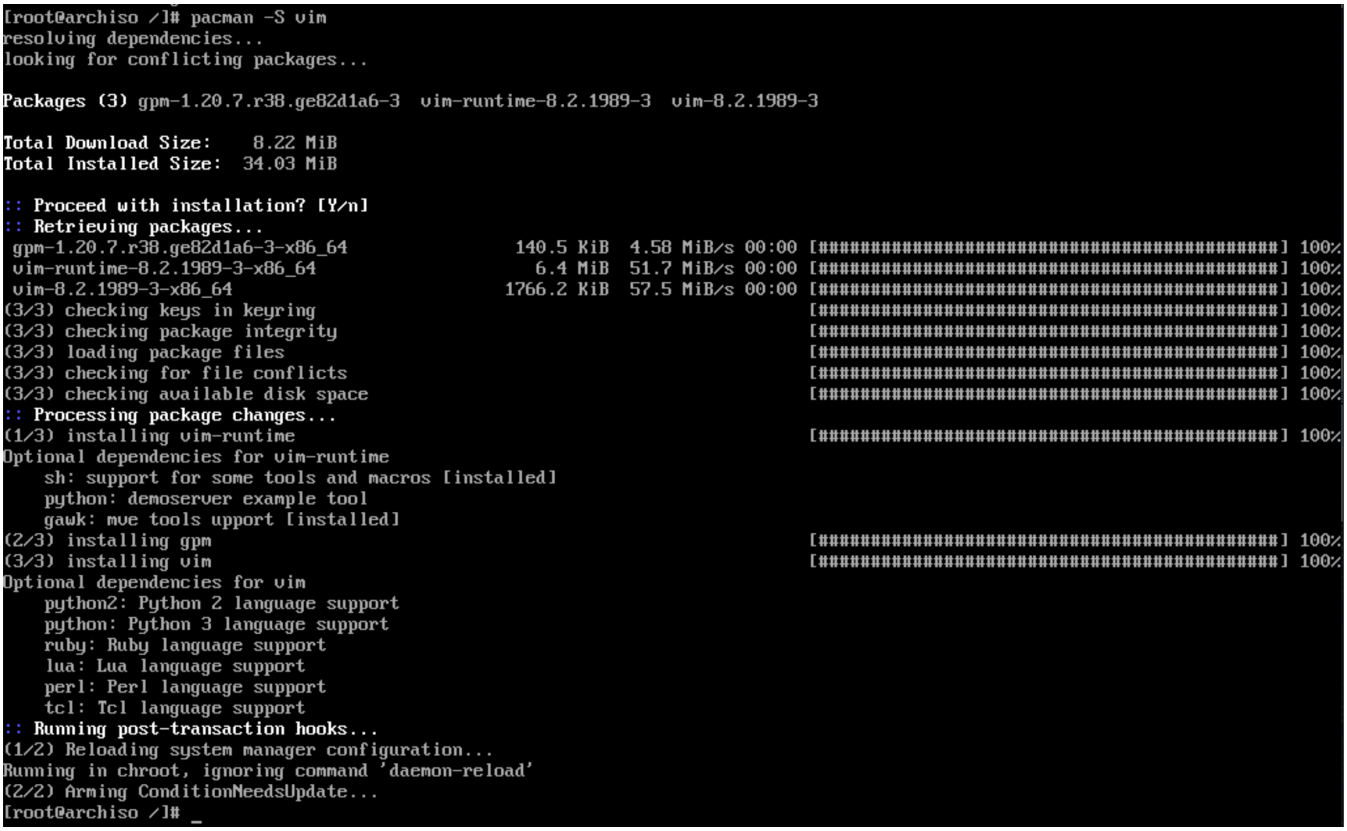
  ```
  cat /etc/pacman.d/mirrorlist
  ```



  Then sync the `pacman` database and make sure it's up-to-date:



Then, let's do `pacman -S vim`:

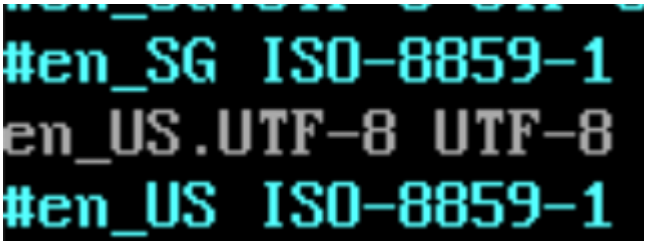


Enable the `color`:

```
vim /etc/pacman.conf

# Search and enable `Color` line, save and exit
```

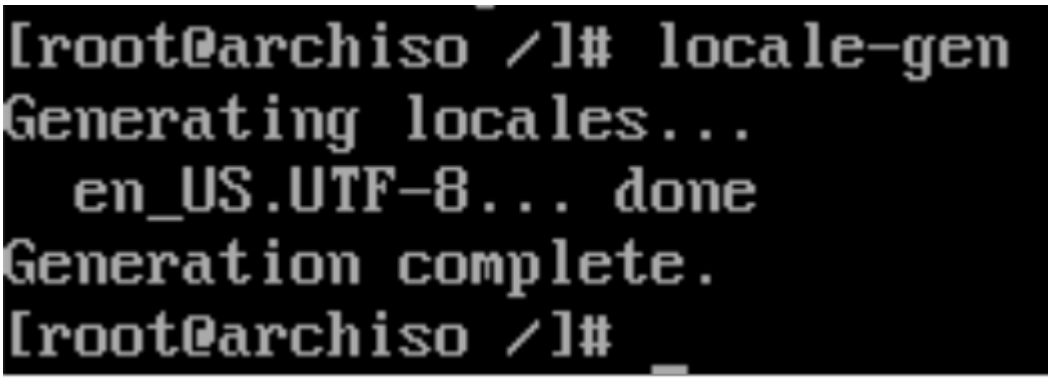
- Localization

  `vim /etc/locale.gen` and uncomment `en_US.UTF-8 UTF-8` line and other needed `locales`.



Then, genereate the locale settings:

```
locale-gen
```



`echo "LANG=en_US.UTF-8" > /etc/locale.conf`:

`vim /etc/vconsole.conf` and add your custom keybinding (if needed):

```
# Add my custom settings below (`Caps_Lock` works like `Escape`) to
# `/etc/vconsole.conf`.
# Save and exit.
KEYMAP=us
keycode 9 = Escape
keycode 66 = Escape
```

Change to `KEYMAP=colemak` if you use `colemak` keyboard layout!!!

- Hostname and host settings
  - `vim /etc/hostname`, set to your hostname.
  - `vim /etc/hosts` with the base settings like below:

```
127.0.0.1       localhost
::1             localhost
127.0.1.1       YOUR_HOSTNAME_HERE.localdomain
YOUR_HOSTNAME_HERE
```

- Set root password

```
passwd
```

Extra configuration for USB

- RAM disk image

  In order to boot the Linux Kernel persistently off of a USB device, some adjustments may be necessary to the initial RAM disk image. We need to ensure that block device support is properly loaded before any attempt at loading the filesystem. This not always the way a RAM disk image is configured in a generic Linux installation, and I suspect this may one of the failure points in other Linux USB installations out there.

  So we need to make some changes to `/etc/mkinitcpio.conf`:

```
# Editr the config
vim /etc/mkinitcpio.conf

# And ensure the `block` hook comes before the `filesystems` hook
# and directly after the `udev` hook like the following:
HOOKS=(base udev block filesystems keyboard fsck)

# Save it and exit.
```

```
##   NOTE: If you have /usr on a separate partition, you MUST include the
#    usr, fsck and shutdown hooks.
HOOKS=(base udev block filesystems keyboard fsck)
```

Now, regenerate the initial RAM disk image with the changes made:

```
mkinitcpio -p linux
```

```
[root@archiso /]# mkinitcpio -p linux
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux.img
==> Starting build: 5.9.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: aic94xx
==> WARNING: Possibly missing firmware for module: wd719x
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'fallback'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux-fallback.img -S autodetect
==> Starting build: 5.9.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: aic94xx
==> WARNING: Possibly missing firmware for module: wd719x
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux-fallback.img
==> Image generation successful
[root@archiso /]# _
```

- Network interface names

  Arch Linux's basic service manager, **`systemd`**, assigns network interfaces predictable names based on the actual device hardware. This is great for just about any other type of install, but can pose some problems for the portable USB installation we're going for.

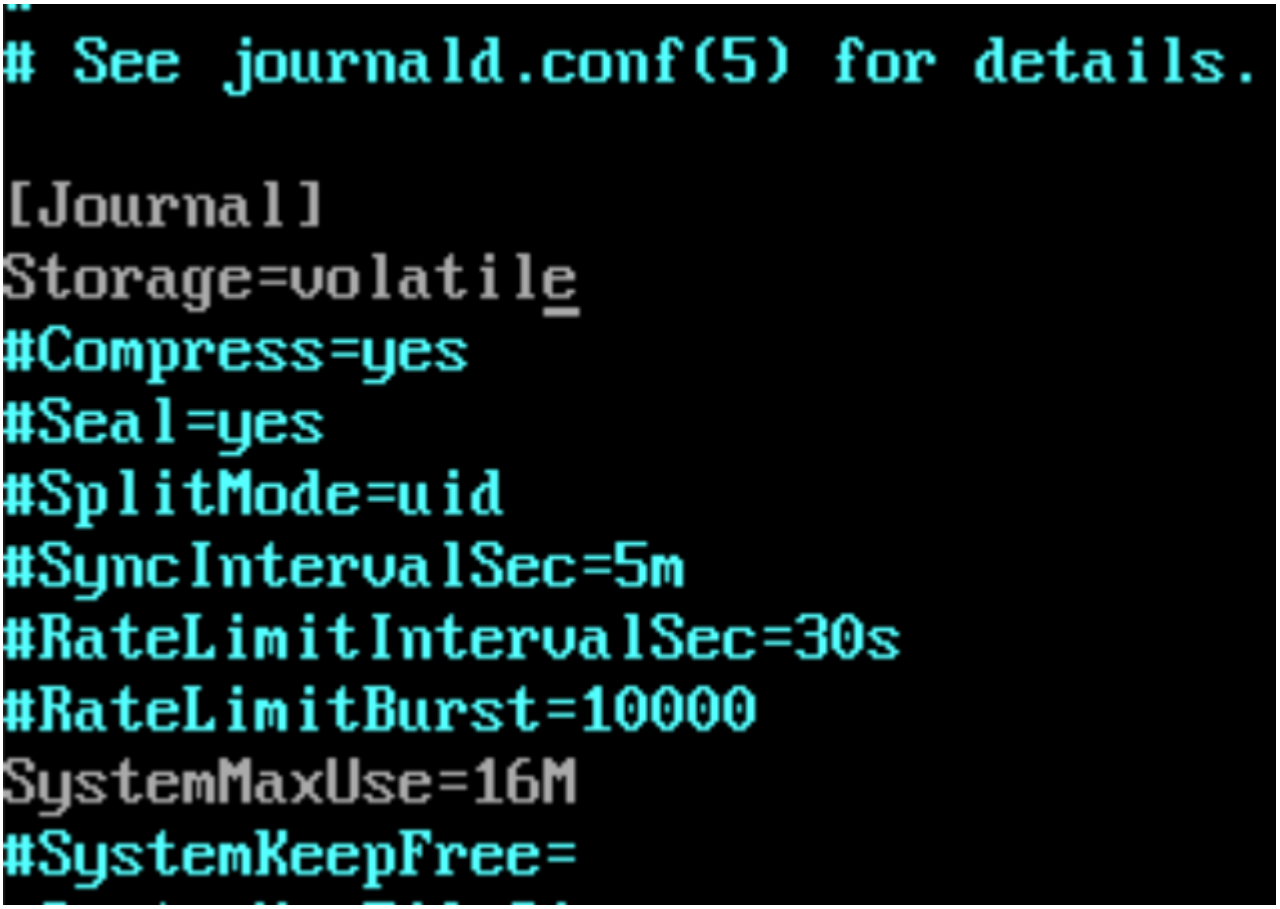
  To ensure that the ethernet and wifi interfaces will always be respectively named **`eth0`** and **`wlan0`**, revert the Arch Linux USB back to traditional device naming:

  ```
  ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules
  ```

- Journal configuration

  A default installation of Arch Linux is setup with **`systemd`** to continuously journal various information about current processes and write that data to storage on disk. For a persistent bootable installation on a flash memory device, however, we can change some options in `journald.conf` to enable journal keeping entirely in RAM (thus reducing writes to the flash device). To control where journal data is stored.

  ```
  # Editr the config
  vim /etc/systemd/journald.conf

  # To switch journal data storage to RAM, set the storage variable to
  # `volatile` by ensuring the following line is uncommented:
  Storage=volatile

  # As an additional precaution, to ensure the operating system doesn't
  # overfill RAM with journal data, set the max-use variable like
  below:
  SystemMaxUse=16M

  # Save it and exit
  ```



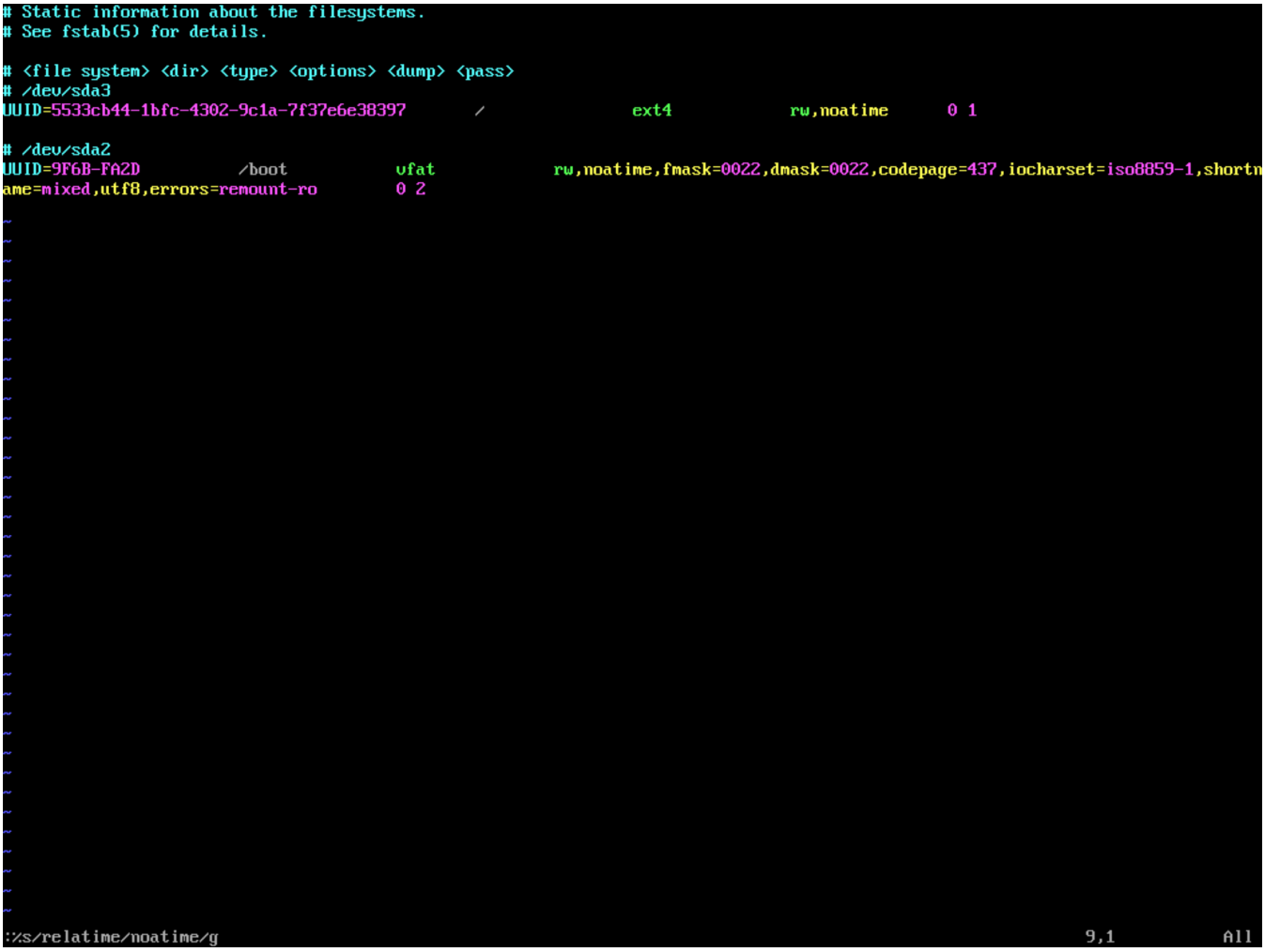
- Mount options

  Modern filesystems are able to record various metadata (last accessed, last modified, user rights, etc.) about their files. A default filesystem mount generally keeps track of as much as this information as possible. For a persistent bootable operating system on a flash memory device, however, we should limit some of this record keeping in order to reduce writes to the flash device.

  Using the **noatime** mount option in `fstab` will disable the record keeping of file access times: **no writes will occur when a file is read, only when it is modified.**

  To disable record keeping of file access times for the bootable USB:

```
vim /etc/fstab


# Replace all mount options from `relatime` to `noatime`.
# Save it and exit
```



Install `grub` to hard drive

Make sure you're in the `New Arch Linux` root environment. If not, please run `arch-chroot /mnt` to go inside it.

```
# `intel-ucode` is for the bootloader to know Intel CPU architecture,
# you need to change to your CPU one.
#
# Optionally, you can install `os-prober` if you want `grub` to detect
exists OS.
# For example, you want `grub` to handle multi OS boot situation.
pacman -S grub efibootmgr intel-ucode

# Generate then grub config
mkdir /boot/grub
grub-mkconfig > /boot/grub/grub.cfg

# Install CPU specified `grub`, you can run `uname -m` to confirm your
CPU architecture.
# For example, to install `x86_64` architecture and `EFI`.
grub-install --target=x86_64-efi --efi-directory=/boot

# After that, it will generate `/boot/EFI/arch/grubx64.efi`!!!
```

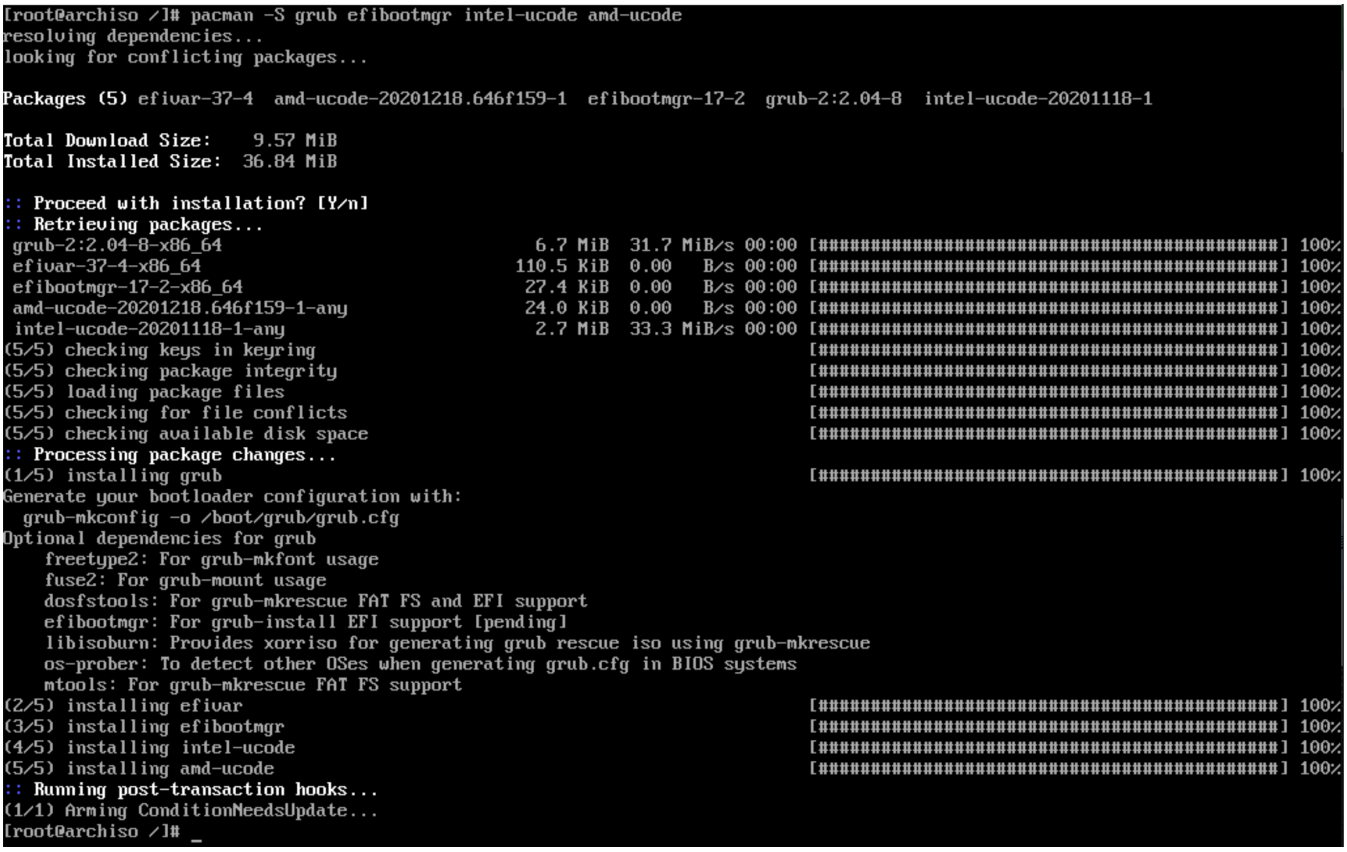
Install `grub` to USB

Make sure you're in the `New Arch Linux` root environment. If not, please run `arch-chroot /mnt` to go inside it.

To enable booting the USB in both modes (**`BIOS`**, **`UEFI`**), two bootloaders will need to be installed.

- Install the following packages:

```
# `intel-ucode` and ` amd-ucode ` is for the bootloader to know Intel
CPU
# architecture, you need to change to your CPU one.
#
# As we install to USB, then  we needs to have `microcode` for both
# manufacturer processors
#
# Optionally, you can install `os-prober` if you want `grub` to
detect exists OS.
# For example, you want `grub` to handle multi OS boot situation.
pacman -S grub efibootmgr intel-ucode amd-ucode
```



- Setup **`bootloader`**:
  - View the current block devices to determine the target USB device:

    ```
    lsblk
    ```

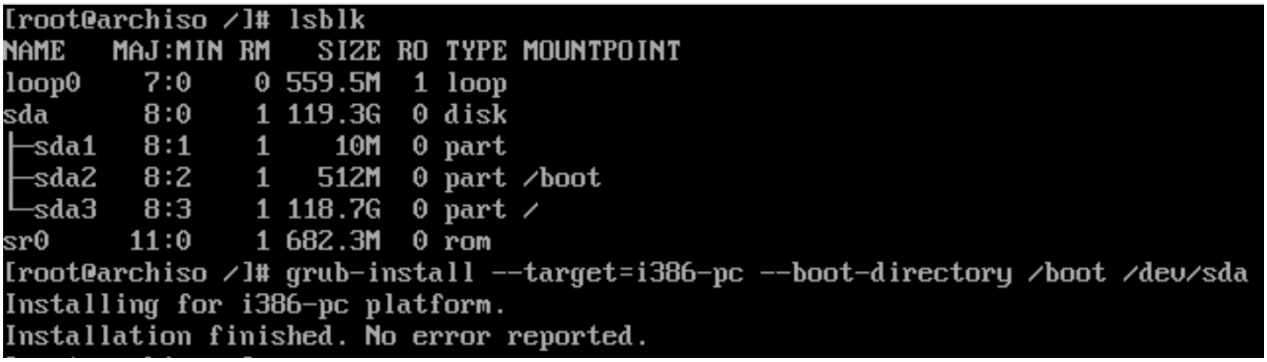
    Make sure that `/dev/sdX` block device is the USB you're installing to !!!

    All the command below, the `/dev/sdX` below means the USB driver itself, not to any partition. So please DO NOT add any number at the end!!!

    ```
    # `/dev/sdX` is correct
    # `/dev/sdXn` is NOT correct
    ```

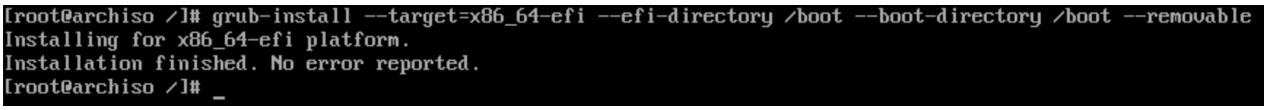
  - Setup **`GRUB`** for **`MBR/BIOS`** booting mode (replace the `X` to your real device letter)

```
grub-install --target=i386-pc --boot-directory /boot /dev/sdX
```



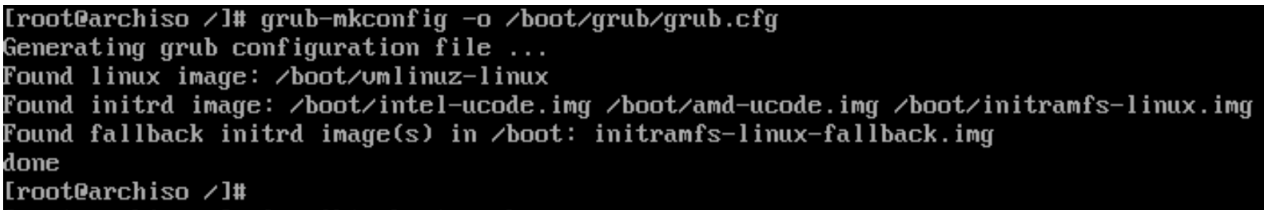
- Setup **GRUB** for **UEFI** booting mode

```
# Install CPU specified `grub`, you can run `uname -m` to confirm
your CPU architecture.
# Install `x86_64` architecture and `EFI`.
grub-install --target=x86_64-efi --efi-directory /boot --boot-
directory /boot --removable
```



- Generate a **GRUB** configuration:

```
grub-mkconfig -o /boot/grub/grub.cfg

# After that, it will generate something important below:
# /boot/EFI/BOOT/BOOTX64.EFI
# /boot/grub/x86_64-efi/
# /boot/grub/i386-pc/
```



```
[root@archiso /]# ls -lht /boot/EFI/BOOT/
total 128K
-rwxr-xr-x 1 root root 128K Dec 22 15:50 BOOTX64.EFI
[root@archiso /]# _
```

```
[root@archiso /]# ls -lht /boot/grub
total 64K
-rwxr-xr-x 1 root root 5.1K Dec 22 15:51 grub.cfg
drwxr-xr-x 2 root root  20K Dec 22 15:50 x86_64-efi
drwxr-xr-x 2 root root 4.0K Dec 22 15:50 locale
drwxr-xr-x 2 root root 4.0K Dec 22 15:47 fonts
-rwxr-xr-x 1 root root 1.0K Dec 22 15:47 grubenv
drwxr-xr-x 2 root root  20K Dec 22 15:47 i386-pc
drwxr-xr-x 3 root root 4.0K Dec 22 15:47 themes
[root@archiso /]#
```

Recommended package to install

All the packages below are optional but recommended which can make your life a bit easier before you install any DE (`Desktop Environment`) or WM (`Window Manager`).

Make sure you're in the `New Arch Linux` root environment. If not, please run `arch-chroot /mnt` to go inside it.

- Networking
  - `netctl` - A CLI and profile-based network manager.
  - `ifplugd` - a daemon which will automatically configure your Ethernet device when a cable is plugged in and automatically unconfigure it if the cable is pulled.
  - And the based WIFI support CLI.

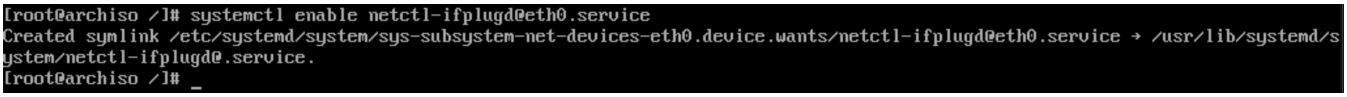
```
pacman -S netctl ifplugd iw wpa_supplicant dhcpcd
```

Copy the example ethernet profile to `/etc/netctl/`:

```
cp /etc/netctl/examples/ethernet-dhcp /etc/netctl/eth0-dhcp
```

Enable `ifplugd` to automatically connect to any available wired network:

```
systemctl enable netctl-ifplugd@eth0.service
```



Enable network time synchronization:

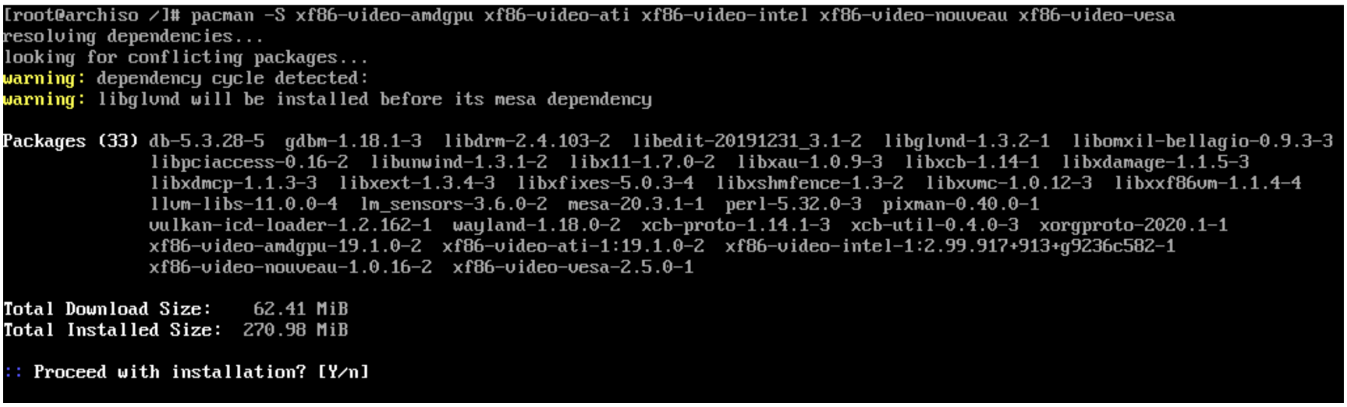
```
systemctl enable systemd-timesyncd.service

# You can check you time setting by command `timedatectl`
timedatectl
#    Local time: Wed 2014-09-24 21:19:26 CST
#  Universal time: Wed 2014-09-24 13:19:26 UTC
#        RTC time: Wed 2014-09-24 13:19:26
#        Timezone: Asia/Shanghai (CST, +0800) // time zone you setted
#     NTP enabled: yes // if you enable systemd-timesysncd success
here will be yes, otherwise you need use `systemctl status systemd-
timesyncd.service` to check it
#NTP synchronized: yes
# RTC in local TZ: no
#      DST active: n/a
```

For the `WIFI` configuration, please go to `Configuration` chapter.

- Video drivers

  To support most common `GPUs`, install all five basic open source video drivers:

  ```
  pacman -S xf86-video-amdgpu xf86-video-ati xf86-video-intel xf86-video-nouveau xf86-video-vesa
  ```



- Touchpad support

  Install support for standard notebook touchpads:

  ```
  pacman -S xf86-input-synaptics
  ```

- Battery support

  Install support for checking battery charge and state:

  ```
  pacman -S acpi tlp
  ```

  Enable the service

  ```
  systemctl enable tlp.service
  ```

- Build tools if you needed

```
pacman -S base-devel man
```

- Basic editor and shell if you needed

```
pacman -S vim fish
```

- Basic ssh tool if you needed

```
pacman -S openssh
```

- `bat` is a enhancement for `cat` written in `Rust`.

  It adds some default features below:

  - Line number
  - Color
  - Git integration, can show git status
  - Works like `less` you can do `vim` searching
  - Automatic paging

```
pacman --sync --refresh bat
```

usage:

```
bat FILE_NAME

# Pipe case
curl -s https://sh.rustup.rs | bat
```

- `procs` is a replacement for `ps` written in `Rust`.

  ```
  pacman --sync --refresh procs
  ```

  Optionally, you can add `alias ps="procs"` (for `bash`) or `abbr ps "pros" (for fish`) to your shell configuration file.

  Some use cases:

  ```
  # Query `vim` related process
  procs vim

  # Query `vim` or `alacritty` related process
  procs --or vim alacritty

  # Query `vim` or `alacritty` and ascending sort by PID
  procs --or --sorta PID vim alacritty

  # Query `vim` or `alacritty` and descending sort by memory
  procs --or --sortd VmRss vim alacritty

  # Query `vim` or `alacritty` related process in watch mode (1s refresh rate)
  procs --or --watch vim alacritty
  ```

  Also, you can custom the config to control which informat you want to show:

  `vim ~/.config/procs/config.toml` with the following settings:

```
# For more detail setting information, plz visit
https://github.com/dalance/procs

[[columns]]
# The column kind to show
kind = "Pid"
# Use which color to show
style = "BrightYellow|Yellow"
# This column can be search for as numeric parameter passed
# to `procs` command?
numeric_search = true
# This column can be search for as non-numeric parameter passed
# to `procs` command?
nonnumeric_search = false
# Alignment, "Left, Center, Right". "Left" by default.j
# align = "Right"

[[columns]]
kind = "User"
style = "BrightMagenta|Magenta"
numeric_search = false
nonnumeric_search = true
align = "Right"

[[columns]]
kind = "Separator"
style = "BrightWhite|White"
color_075 = "BrightWhitee"
numeric_search = false
nonnumeric_search = false
align = "Center"

[[columns]]
kind = "Tty"
style = "BrightWhite|White"
numeric_search = false
nonnumeric_search = false
```

```
align = "Center"

[[columns]]
kind = "Threads"
style = "BrightWhite|White"
numeric_search = false
nonnumeric_search = false
align = "Center"

[[columns]]
kind = "TcpPort"
style = "BrightCyan|Cyan"
numeric_search = true
nonnumeric_search = false
align = "Right"

[[columns]]
kind = "VmRss"
style = "BrightGreen|Green"
numeric_search = false
nonnumeric_search = false
align = "Right"

# [[columns]]
# kind = "UsageCpu"
# style = "BrightGreen|Green"
# numeric_search = false
# nonnumeric_search = false
# align = "Center"
#
# [[columns]]
# kind = "UsageMem"
# style = "BrightGreen|Green"
# numeric_search = false
# nonnumeric_search = false
# align = "Center"
```

```
[[columns]]
kind = "Separator"
style = "BrightWhite|White"
numeric_search = false
nonnumeric_search = false
align = "Center"

[[columns]]
kind = "Command"
style = "BrightWhite|White"
numeric_search = false
nonnumeric_search = true
align = "Left"

[search]
numeric_search = "Exact"
nonnumeric_search = "Partial"
logic = "And"

[display]
show_self = false
show_thread = false
show_thread_in_tree = true
cut_to_terminal = true
cut_to_pager = false
cut_to_pipe = false
color_mode = "Always"

[sort]
column = 0
order = "Ascending"
# order = "Descending"

[docker]
path = "unix:///var/run/docker.sock"
```

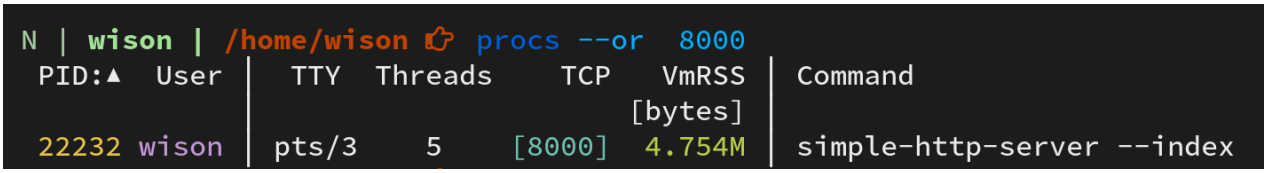
```
[pager]
mode = "Auto"
```

The custom config above show with the follow colums:

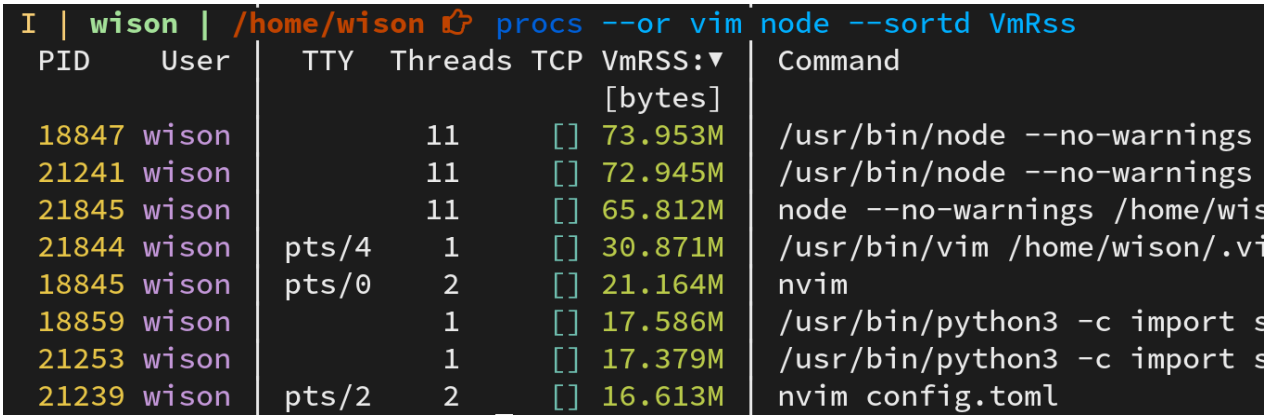
`PID, User, TTY, Threads, TCP, VmRSS, Command`

Here is the running example:

- Search by TCP port:



- `Or` search by name and sort (desc) by memory usage:



<++>

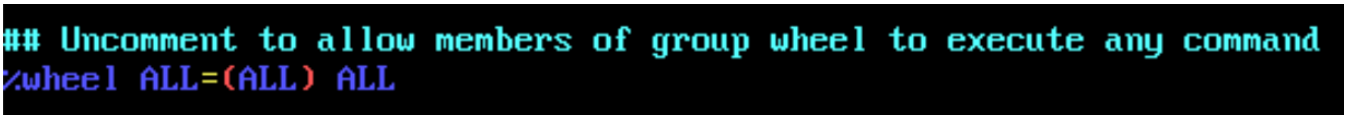
Finish installation

Below are the final steps to finish the `Arch Linux` installation.

Make sure you're in the `New Arch Linux` root environment. If not, please run `arch-chroot /mnt` to go inside it.

- Add a new user and assign it to the `sudo` group

```
# Add new user
useradd -m -G wheel YOUR_USER_NAME

# Set password
passwd YOUR_USER_NAME

# Install `sudo`
pacman -S sudo

# Enable `wheel` group for `sudo` command
visudo

# Enable the below line:
# %wheel ALL=(ALL) ALL
#
# Save and exit.
```



- Exit and shutdown

```
# Exit current Arch root.
exit

# Unmount all mounted folders and sync
umount /mnt/boot /mnt && sync

# Shutdown, all done:)
shutdown -h now
```

Arch configuration guide

Before starting, recommended that login with your new user (not the `root` user) to finish all the steps below.

You should know about Getty

What is `Getty`?

After reboot, you will see a default terminal login prompt which prints with **tty1**, that's the `getty`.

In Arch Linux, `agetty` is the default `getty`, it prompts for a login name and invokes the `/bin/login` command.

You can press `Ctrl+Alt+F1 ~ F6` to switch between **tty1 ~ tty6**.

The **`tty7`** (`Ctrl+Alt+F7`) is for the `X` which means your `DE`(Desktop Environment) or `WM`(Window Manager).

More information from here

Tips:

- Sometimes, when you do a wrong configuration to your `DE` or `WM`, even the `DM` (Display Manager which works like a graphical login prompt), then you can switch another **tty** and login to fix it, very handy.
- How to change **tty** fonts:

```
# All fonts stay in this folder
ls -lht /usr/share/kbd/consolefonts

# Set the TTY font
setfont FONT_FILE_NAME_HERE
```

Prepare your preferred editor Configuration

Before to edit a lot of configuration files, you better to prepare your personal preferred editor and shell environment before continuing.

For example, if you prefer the pre-installed `vim`, then you need to install `gvim` to enable the `clipboard` feature which makes your life a bit easier.

```
# It will ask you to remove the `vim-minimal` as a conflict, just says
`Y`
sudo pacman --sync --refresh gvim
```

Or you can install `neovim`:

```
sudo pacman --sync --refresh neovim python-pynvim

mkdir ~/.config/nvim
touch ~/.config/nvim/init.vim
```

Let's do it:)

WIFI Configuration

Right now, the newer `Arch Linux` can connect via **`ethernet`** NIC, but how about **`wireless`** NIC?

- Copy the WIFI config from example and named it as `wlan0-dhcp`

  ```
  cd /etc/netctl
  sudo cp -rvf examples/wireless-wpa ./wlan0-dhcp
  ```

- Edit `wlan0-dhcp` to change your WIFI `SSID` and `KEY` (encrypted password in HEX string)

```
sudo vim wlan0-dhcp

# Change the `ESSID` to your WIFI `SSID`

# Go to the bottom, run the command below to read the
# `wpa_passphrase` result into current file.
:r !wpa_passphrase YOUR_SSID_HERE YOUR_WIFI_PASSWORD_TEXT_HERE | grep psk

# After that, you should get something like below:
# #psk="xxxxx"
# psk="XXXXXX" // That's encrypted WIFI PASS in HEX string
#
# So you need to copy that "XXXX" hex string and make it looks like below
# to set the `Key` value.
# Make sure the value start with `\"` and then follow your HEX string
Key=\"XXXXXX

# Finally, enable the `Hidden=yes` line if your SSID is hidden.
# Then save and exit
```

Pay attention:

For connecting to the `5G` network, you need to match 2 conditions below:

- *You have to use unencrypted (plaintext) password to fill the `Key` field.*
- *The `5G` network needs to pick the `channel` to `40`.*

Otherwise, it won't work!!!

- Start connecting to WIFI

  Any file located at `/etc/netctl` folder is call `profile`, you can run the `netctl start PROFILE_NAME` to start any profile-based network configuration:

```
sudo netctl start wlan0-dhcp
```

If start fail, then run `sudo netctl status wlan0-dhcp` to read the detail error.

- Connect to WIFI when computer boots

  You can create a `systemd` service to connect to WIFI automatic when computer boots

  ```
  sudo netctl reenable wlan0-dhcp
  ```

  But this is not recommended if you're installing the `Arch Linux` to **USB**, as not all computers always have the `wlan0`. In the case which doesn't have the `wlan0` NIC, then the booting process will keep waiting before time out, it will waste a couple of seconds.

  For solving this, you can add a script to start `wlan0-dhcp` profile into a script which will be added to **`lightdm`**, that will be perfect.

- Run the script to list all available networks

  - Install `node` if you don't have it yet:

    ```
    sudo pacman --sync --refresh nodejs-lts-erbium
    ```

  - Create the `~/scripts/scan-ssid.js` with the following content:

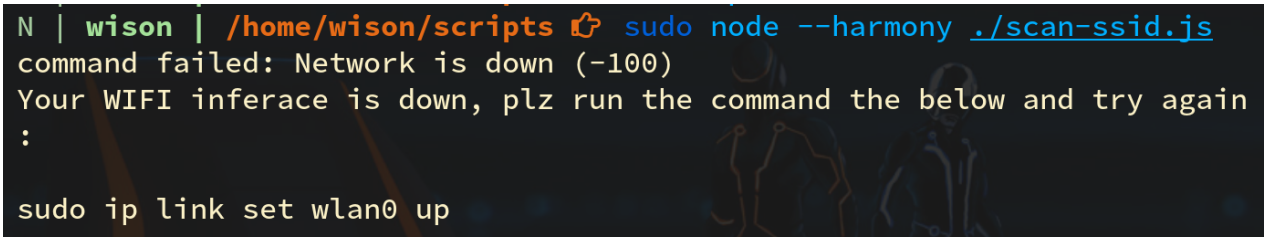
```
const {execSync} = require('child_process')

// Run the `iw` command to do a WIFI scan
const cmd = `iw dev wlan0 scan | grep -e "freq:" -e "SSID:"`
let cmdResult = ``
try {
    cmdResult = execSync(cmd)
        .toString()
        .replace(/\t/g, '')
    // console.log(`cmdResult: ${cmdResult}`)
}
catch (error) {
    if (error.message && error.message.indexOf('Network is down')
!== -1) {
        console.log(`Your WIFI inferace is down, plz run the
command the below and try again: \n\nsudo ip link set wlan0
up\n`)
    }

    console.error(error)
    return;
}



// Format the result
const resultList = []

let lastWifiInfo = {freq: '', ssid: ''}
const tempArr = cmdResult.split(`\n`)
    .forEach((tempString, index, arrRef) => {
        const isSsid = index % 2 == 1

        if (isSsid) {
            let fixedSsid = tempString.replace(`SSID: `,
``).trim()
            lastWifiInfo.ssid = Boolean(fixedSsid == ``) ?
`Unknown` : fixedSsid
```

```
            resultList.push(lastWifiInfo)
        } else {
            let freqStr = tempString.replace(`freq: `,
``).trim()
            let is5G = Boolean(parseInt(freqStr, 10) > 5100)
            // console.log(`freqStr: ${freqStr}, is5G: ${is5G}`)

            lastWifiInfo = {freq: '', ssid: ''}
            lastWifiInfo.freq = is5G ? `5G` : `2.4G`
        }
    })

let resultTable = `\n[ Available WIFI networks ]` +
    `\n-------------------------------------------\n` +
    resultList.map(wifiInfo => `${wifiInfo.ssid} -
${wifiInfo.freq}`)
        .join(`\n`) + `\n`

console.log(resultTable)
```

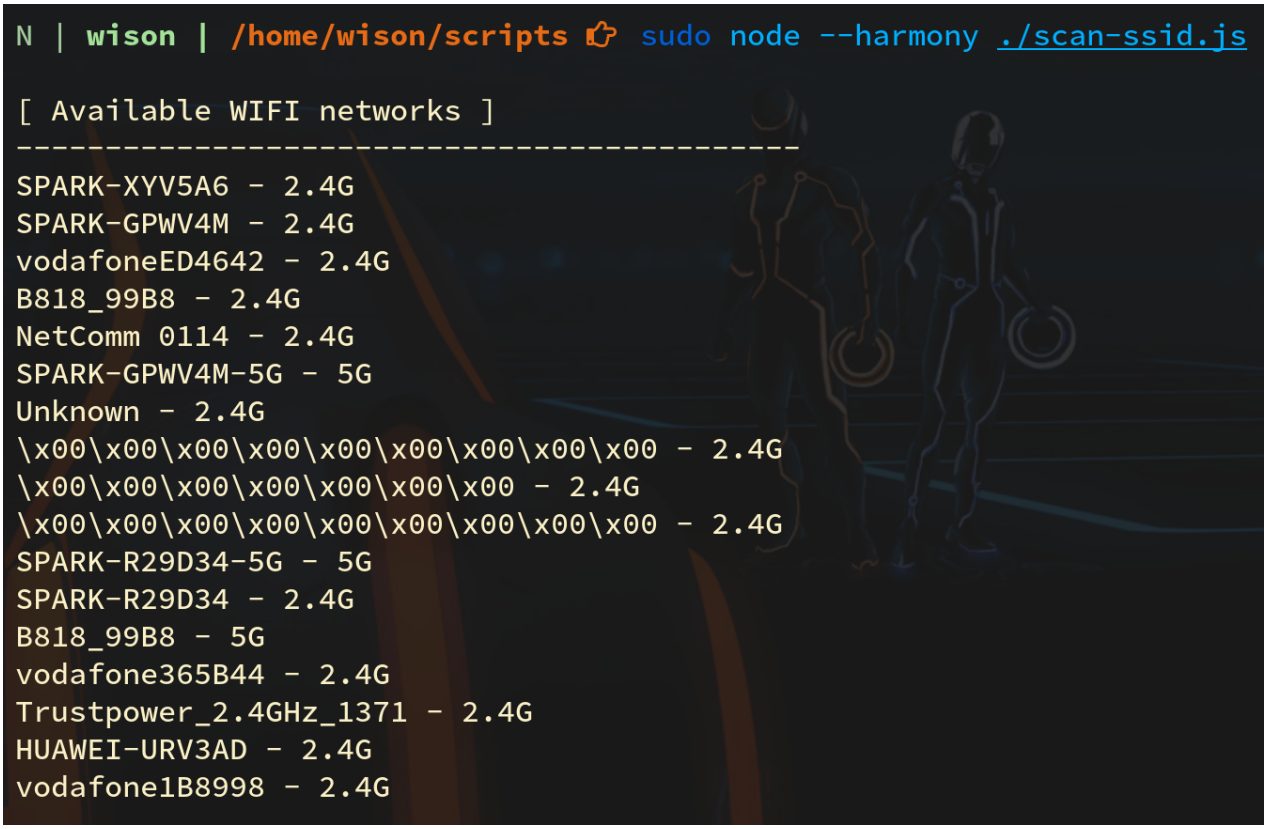
- Run the script like below:

```
sudo node --harmony ~/scripts/scan-ssid.js
```

If you didn't up the `wlan0` interface, it will print out the error like below:



Then just run `sudo ip link set wlan0 up`, wait for a few seconds, then re-run the script command again. And it should print out like below:



Make your boot faster

By default, `grub` will wait for around **5** second before select the default boot option.

But we can change it in `/etc/default/grub`.

`sudo vim /etc/default/grub` to change some settings to reduce the timeout

```
GRUB_TIMEOUT=0
GRUB_TIMEOUT_STYLE=hidden
```

Save and exit.

Then run the command below to re-generate the grub configuration file:

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Now, reboot to take effect.

A tips for changing the resolution for the `GRUB`:

You can change the following settings to `/etc/default/grub`

```
# Try 1024x768x32 first, if fail, then fallback to auto resolution
GRUB_GFXMODE=1024x768x32,auto
```

Install or update kernel

First you need to check the current linux kernel you can install by running:

```
pacman --sync --search linux | grep core/linux

# [ The current kernel installed ]

# core/linux 5.10.5.arch1-1 [installed]
# core/linux-api-headers 5.8-1 [installed]
# core/linux-docs 5.10.5.arch1-1
# core/linux-firmware 20201218.646f159-1 [installed]
# core/linux-headers 5.10.5.arch1-1 [installed]


# [ The LTS (Long Term Support) kernel ]

# core/linux-lts 5.4.87-1
# core/linux-lts-docs 5.4.87-1
# core/linux-lts-headers 5.4.87-1
```

If you need to install driver (webcam driver, usb driver), then you have to install `headers`. Otherwise, you don't need that.

After you confirm you ready need that, then pick the one you like to install:

```
# Install the latest kernel and headers
sudo pacman --sync --refresh linux linux-headers

# Install the latest LTS kernel and headers
sudo pacman --sync --refresh linux-lts linux-headers-lts
```

Optionally, you can list all installed kernels in the **`grub`** bootloader UI, then you can pick which one you want to boot into.

This just optaional, you don't need to do that if don't need it

`sudo vim /etc/default/grub` with the following settings:

```
GRUB_DISABLE_SUBMENU=y
GRUB_DEFAULT=saved
GRUB_SAVEDEFAULT=true
```

Then re-generate the grub configuration:

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Reboot.

Install `yay` to enable AUR

What is AUR?

AUR stands for Arch User Repository. It is a community-driven repository for Arch-based Linux distributions users. It contains package descriptions named **PKGBUILDs** that allow you to compile a package from source with `makepkg` and then install it via `pacman` (package manager in Arch Linux).

How to enable **AUR** installation:

```
mkdir ~/temp && cd ~/temp

pacman -S --needed git base-devel
git clone https://aur.archlinux.org/yay.git
cd yay
makepkg -si

cd ~/temp && rm -rf yay
```

If you want to change or add any mirror server to `/etc/pacman.d/mirrorlist`, you can go to here

Install X implementation

The `X` Window System (`X11`, or simply `X`) is a windowing system for bitmap displays, common on Unix-like operating systems. It provides the basic framework

for a GUI environment:

- Drawing
- Moving windows
- Interact with mouse and keyboard

It's `Client, Server` architecture

How to install a `X` implementation?

```
sudo pacman -S xorg xorg-server
```

More information from here

Install `ligthDM` as Dislpay Manager

- What is `Dislpay Manager`?

  A display manager, or login manager, is typically a graphical user interface that is displayed at the end of the boot process in place of the default shell.

- We will install one of the implementation which is called: **LightDM**

  LightDM is a cross-desktop **D**isplay **M**anager (also known as a **L**ogin **M**anager). Its key features are:

  - Cross-desktop - supports different desktop technologies.
  - Supports different display technologies (X, Mir, Wayland ...).
  - Lightweight - low memory usage and high performance.
  - Supports guest sessions.
  - Supports remote login (incoming - XDMCP, VNC, outgoing - XDMCP, pluggable).
  - Comprehensive test suite.
  - Low code complexity.

  Here has more detail information.

- Installation

```
# `lightdm-webkit2-greeter` is one of the lightdm greeter which will explain below
sudo pacman -S lightdm lightdm-webkit2-greeter
```

- Change the default `greeter`

  A `greeter` just like a graphical login prompt, you can install any one you like.

  By default, all installed `gretter` are located in `/usr/share/xgreeters`

```
ls -lth /usr/share/xgreeters/
# lightdm-webkit2-greeter.desktop
```

  By default, `lightdm` will use `lightdm-gtk-greeter` but you can change it any time in `/etc/lightdm/lightdm.conf`

  If you add new `greeter`, then get the name in `/usr/share/xgreeters` and change in `/etc/lightdm/lightdm.conf` like below:

```
[Seat:*]
# ...
# For example, we just installed this one
greeter-session=lightdm-webkit2-greeter
# ...
```

- Enable the `lightdm` service when booting

```
sudo systemctl enable lightdm.service
```

- Restart the `lightdm` service when booting

After you change the greeter or another settings, you need to restart the service to take effect. Pay attention that this command will logout the curernt session, make sure save all docs before you do that

```
sudo systemctl restart lightdm.service
```

If you want to change setting very often and don't want to be foreced to logout, then you can open another **`tty`** (Ctrl+Alt+F1 ~ F6) and run the command above, then back the **`lightdm`** will refresh in the **`tty7`** which more convenience.

- Install new theme for the `lightdm-webkit2-greeter`

  It's super easy to add a different theme based on `lightdm-webkit2-greeter`, for example, install this one:

  ```
  yay -Sy lightdm-webkit-theme-aether
  ```

  After a theme be installed, it saves in `/usr/share/lightdm/themes`

  ```
  ls -lht /usr/share/lightdm/themes

  # lightdm-webkit-theme-aether
  ```

  So, you need to change the theme name in `/etc/lightdm/lightdm-webkit2-greeter.conf` like below:

  ```
  webkit_theme = lightdm-webkit-theme-aether
  ```

  Save it and logout to take effect.

- Troubleshooting

  If you can't start the `lightdm` or `lightdm-webkit2-greeter` (which includes can't see the webkit2 themes), plz double check all configurations to see whether you put the wrong spelling there, as around 90% chance is that:)

If yes, then switch another `tty` to fix it and restart `lightdm` again:

```
sudo systemctl restart lightdm.service
```

If you make sure NO wrong spelling in the configuration file, then switch to another `tty` and run the command below to see whether some helpful error to figure out what's happening there:

```
/usr/bin/lightdm-webkit2-greeter --help
```

My personal very bad luck experience, it causes by missing the correct version of `icu` package to be installed. I think that's because I installed `lightdm-webkit-theme-aether` via `yay` which always use the latest package, but my `Arch Linux` use another older package. That's why no matter what I had tried, it still doesn't work. After I run `sudo pacmac -Syc` to upgrade all installed package. It works...:)

Fix resolution issue

In some high `DPI` screens (e.g. Apple Retina Screen), you will see a very high resolution but with tiny small fonts which you even can see what's that.

For that case, you can add the settings below to `~/.Xresources`:

```
# ------------------------------------------------------------------------
------
# The settings below will affect the screen DPI (Dots Per Inch)
#
# Based on the wiki (https://wiki.archlinux.org/index.php/HiDPI "X
Resources"
# section, it says:
#
# For `Xft.dpi`, using the integer multiples 96 usually works best. For
example:
#
# 96  - 100% scaling
# 144 - 150% scaling
# 192 - 200% scaling
# 288 - 300% scaling
# ------------------------------------------------------------------------
------
# Xft.dpi: 96
Xft.dpi: 192
# Xft.dpi: 144
# Xft.dpi: 288
```

Install GTK themes

After installing `X` implementation, it's optional to install different `GTK themes`. Those themes can make the GUI application looks more nice.

`GTK` (The GIMP Toolkit) was initially made by the `GNU` Project for GIMP, but it is now a very popular toolkit with bindings for many languages.

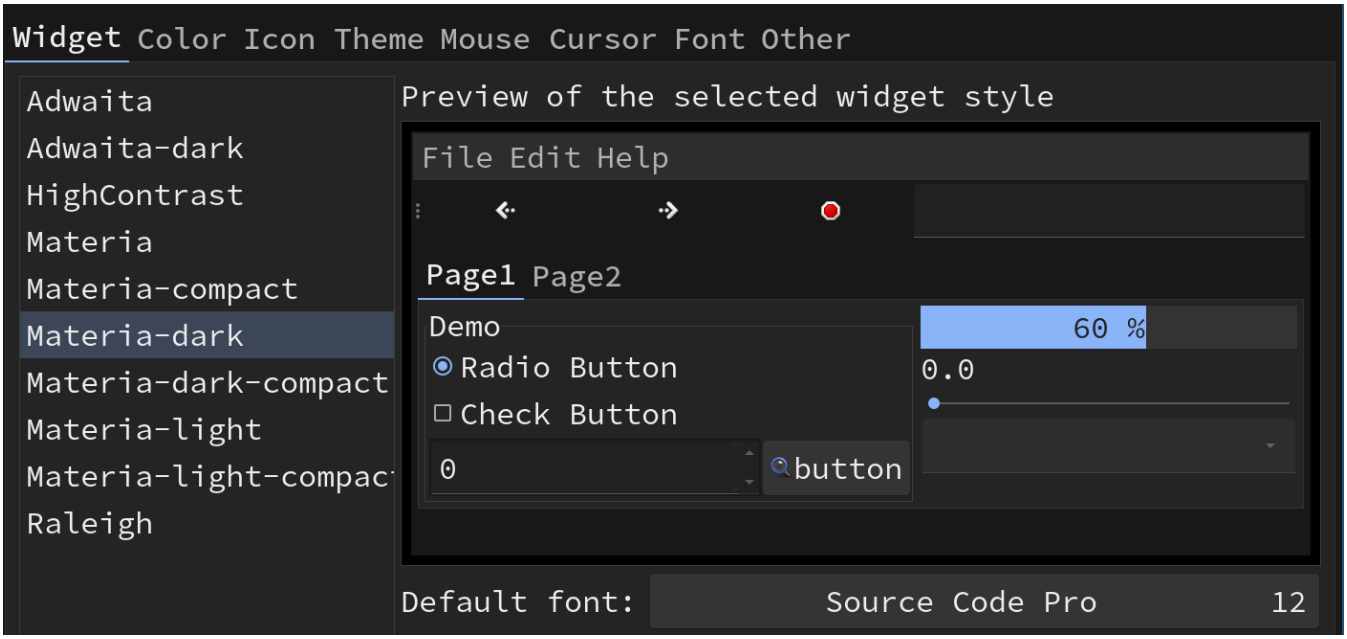
- Install themes and configuration tools

  ```
  sudo pacman -Sy lxappearance materia-gtk-theme
  ```

  After installing, all the themes are located in `/usr/share/themes/`

- Run the configuration tool to pick the theme you like

  ```
  lxappearance
  ```



The `Materia-dark` + `Source Code Pro` font would be a nice chocie.

You can find more GTK themes at here.

Install Google Chrome

- Install via `yay`

```
pacman -S --needed git base-devel
mkdir ~/temp && cd ~/temp
git clone https://aur.archlinux.org/yay.git
cd yay
makepkg -si
cd ~/temp && rm -rf yay

yay -S google-chrome
```

- Apply the selected theme in `lxappearance`:
  - Open `chrome` with this url: `chrome://settings/?search=theme`
  - Then choose `Use GTK+`

- Install `xdg-utils` and set `Chrome` as the default browser

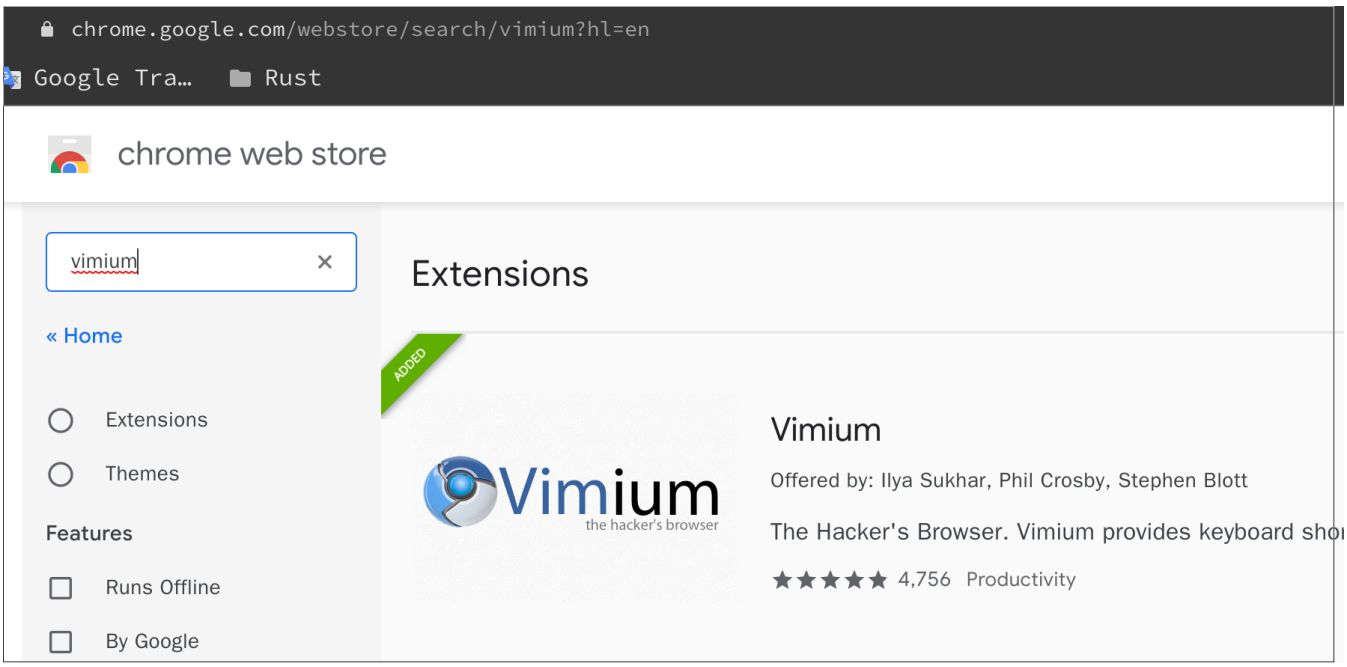
```
sudo pacman -Sy xdg-utils

# Set `chrome` as the default browser
xdg-settings set default-web-browser google-chrome.desktop

# Query to confirm
xdg-settings get default-web-browser
```

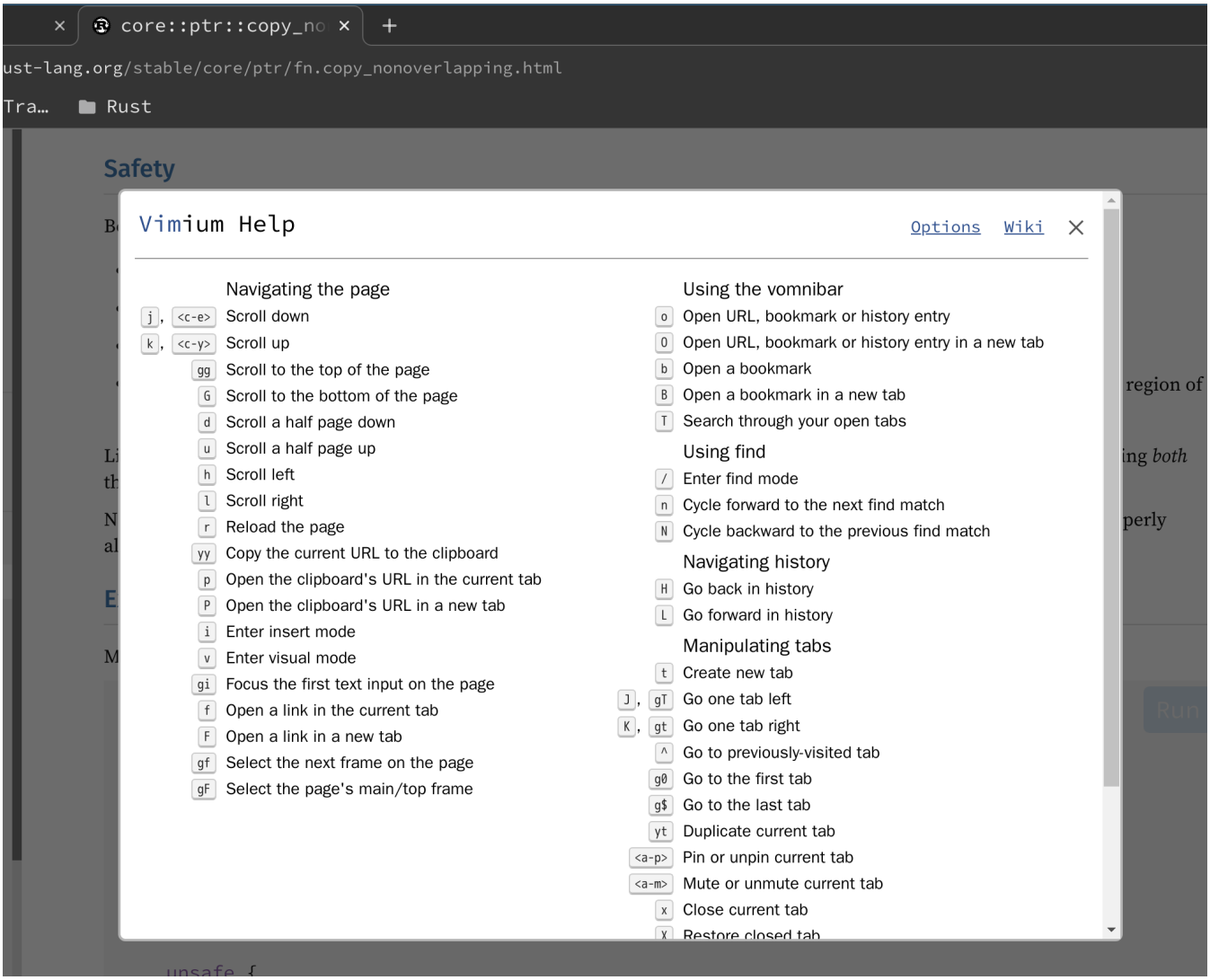
Tips: All installed GUI application links located in `/usr/share/applications/`.

Useful `vim` plugins

- `vimium`

  Open `Chrome web store` and search `vimium` and install it, enjoy the super convenience.





- **`firenvim`**

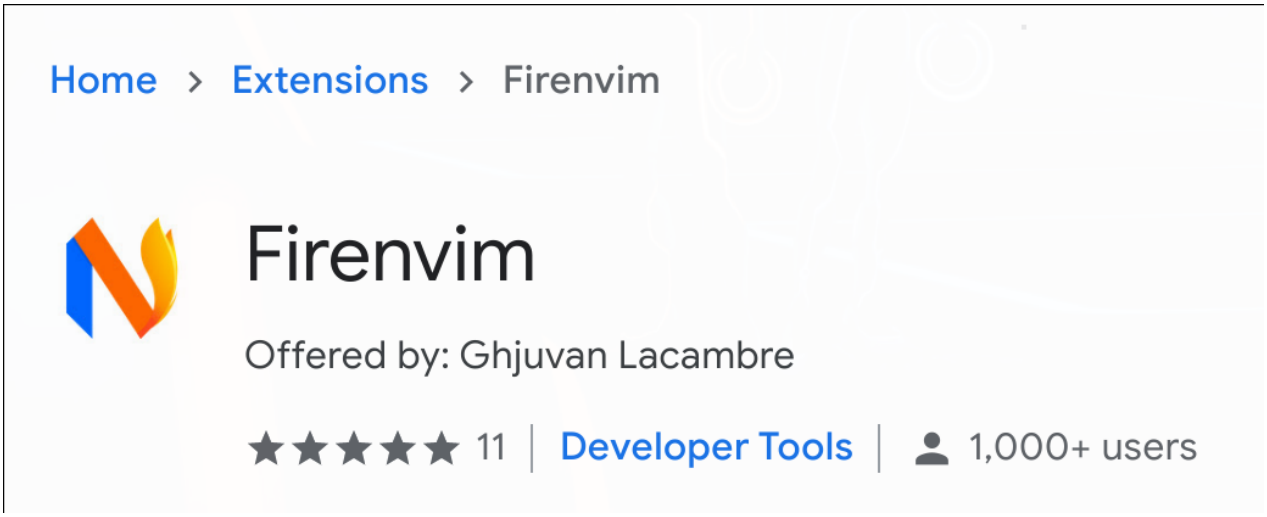
  `firenvim` allows you to use `nvim` inside the browser!!!

  - Install `firenvim`

    ```
    Plug 'glacambre/firenvim', { 'do': { _ -> firenvim#install(0) } }
    ```

  - Install `firenvim` chrome plugin

    Open `Chrome web store` and search `firenvim` and install it.



- `firenvim` settings:

```
" ------------------------------------------------------------------
-------------
" 'glacambre/firenvim'
" ------------------------------------------------------------------
-------------
let g:firenvim_config = {
    \ 'globalSettings': {
        \ 'alt': 'all',
    \  },
    \ 'localSettings': {
        \ '.*': {
            \ 'cmdline': 'neovim',
            \ 'content': 'text',
            \ 'priority': 0,
            \ 'selector': 'textarea',
            \ 'takeover': 'always',
        \ },
    \ }
\ }

" Disable `firenvim` for the particular webiste
let fc = g:firenvim_config['localSettings']
" let fc['https?://[^/]+\.co\.uk/'] = { 'takeover': 'never',
'priority': 1 }
let fc['https?://twitter.com/'] = { 'takeover': 'never',
'priority': 1 }
let fc['https?://twitter.tv/'] = { 'takeover': 'never',
'priority': 1 }


" Change `firenvim` file type to enable syntax highlight, `coc`
works perfectly
" after this settings!!!
autocmd BufEnter github.com_*.txt set filetype=markdown
autocmd BufEnter txti.es_*.txt set filetype=typescript
```

```
" Increase the font size to solve the `text too small` issue
function! s:IsFirenvimActive(event) abort
    if !exists('*nvim_get_chan_info')
        return 0
    endif
    let l:ui = nvim_get_chan_info(a:event.chan)
    return has_key(l:ui, 'client') && has_key(l:ui.client,
'name') &&
      \ l:ui.client.name =~? 'Firenvim'
endfunction

function! OnUIEnter(event) abort
    if s:IsFirenvimActive(a:event)
        " Disable the status bar
        set laststatus=0

        " Increase the font size
        set guifont=SauceCodePro\ Nerd\ Font:h18
    endif
endfunction

autocmd UIEnter * call OnUIEnter(deepcopy(v:event))
```

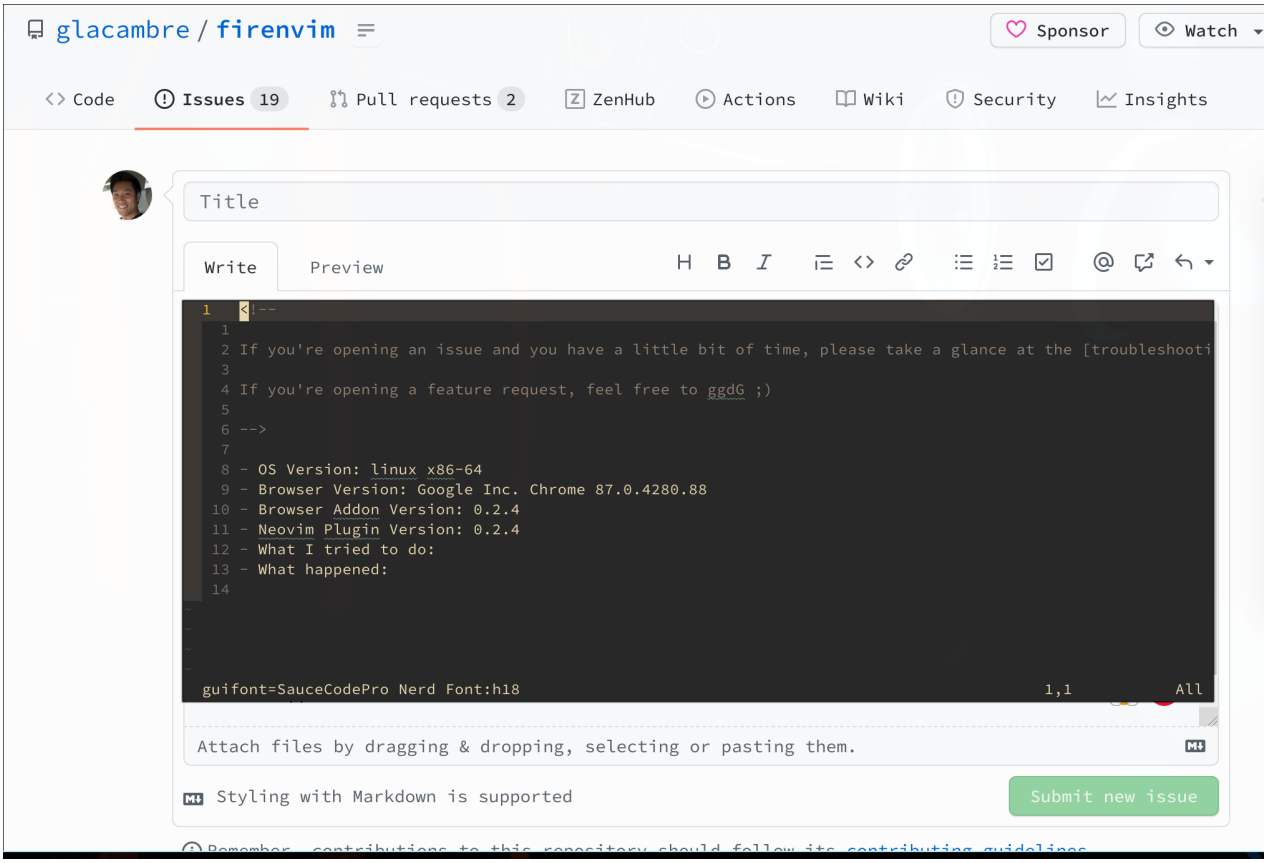
As you can see above, you need to run the `set guifont` command to fix the `text too small` issue when running `firenvim` inside `Chrome`. The important thing is that:

- This will work for `firenvim`:

  ```
  set guifont=SauceCodePro\ Nerd\ Font:h18
  ```

- This will NOT work for `firenvim`:

  ```
  set guifont=SauceCodePro\ Nerd\ Font\ 18
  ```



The screenshot above shows the correct font setting when running `set guifont?` inside the `firenvim`.

Install `i3` Window Manager

A window manager (`WM`) is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface (GUI). It can be part of a desktop environment (`DE`) or be used standalone.

We will install a `Tiling` window manager which calls `i3`.

- How to install

```
sudo pacman -Sy i3 dmenu

# It will ask you what selection you want like below:
#
# 1) i3-gaps 2) i3-wm 3) i3block 4) i3lock 5) i3status
#
# Enter a selection (default=al): // Just enter to access all of
them!!!
```

Then we need on more comopnent from `AUR`:

```
yay -Sy i3exit
```

Component	Description
`i3-gaps`	Provides a grap between different tiling windows.
`i3-wm`	The core component.
`i3block`	Define blocks for your i3bar status line.
`i3lock`	mproved screenlocker based upon XCB and PAM.
`i3status`	Generates status bar to use with i3bar, dzen2 or xmobar.
`i3exit`	A easy to do `shutdown`, `reboot`, `lock` in i3 environment.

- Let `lightdm` to start `i3`

  After finishing install the `i3`, the new `X` session already been added to `/usr/share/xessions`.

  What you need to do just add that `i3.desktop` to `/etc/lightdm/lightdm.conf` like below:

```
user-session=i3

# Logout to take effect
i3exit logout
```

- How to run some scripts or programms when `i3` reload?

  Add the setting below to your `~/.config/i3/config`

  ```
  # Run the command without delay
  exec_always YOUR_COMMAND_HERE

  # Sleep 1 second then run the command
  exec_always sleep 1; YOUR_COMMAND_HERE
  ```

change wallpaper

```
sudo pacman -S feh
```

Download wallpaper you like and then add the following setting to `~/.config/i3/config`

```
exec_always feh --bg--file YOUR_WALLPAPER_FULL_PATH_FILE_NAME
```

Restart `i3` then you should be able to see the new wallpaper.

icon font support

```
sudo pacman -S ttf-font-awesome
```

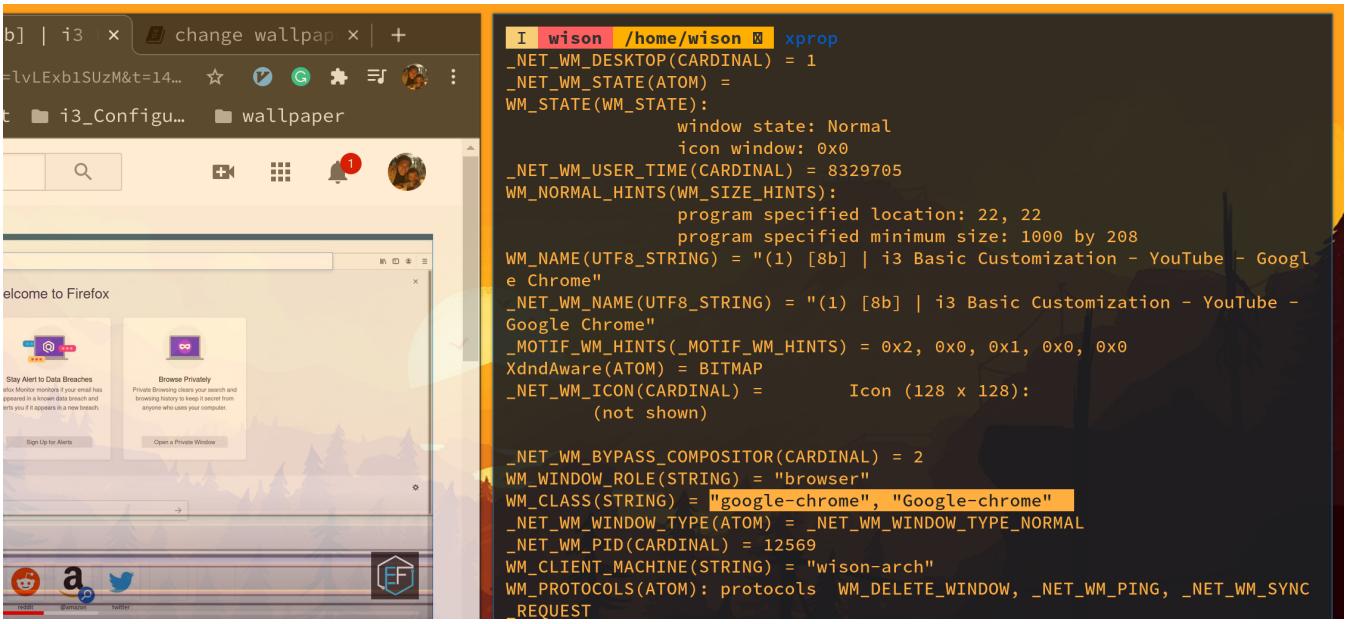
After that, open `https://fontawesome.com/cheatsheet` and copy any icon, then you can paste into any configuration file directly. You need to relogin to take effect.

window customization

- How to get the `Window Class`

  Open the (application) window you want to get the class string name, and then open your terminal app side by side, then run `xprop`.

Right now, you can see the cursor become a `cross`, then click on the window, then you will get the result like below:



The highlight part is the `window class` string value, you can pick any one of them and assign to the `i3` configuration file to apply the window customization:

```
for_window [class="(?i)google-chrome"] border pixel 1
# for_window [class="Skype"] floating enable border normal
# for_window [class="(?i)virtualbox"] floating enable border normal
```

- Apply color setting to window

  Replace all colors you want and reload `i3` to take effect.

```
# Window colors
set $bg_color            #E66B17
set $ibg_color           #551E22
set $border_color        #E66B17
set $text_color          #FFFFFF
set $itext_color         #000000
set $ubg_color           #000000

#                        border           background          text
indicator
client.focused           $border_color    $bg_color
$text_color     $bg_color
client.focused_inactive $ibg_color       $ibg_color
$itext_color    $ibg_color
client.unfocused         $ibg_color       $ibg_color
$itext_color    $ibg_color
client.urgent            $ubg_color       $ubg_color
$text_color     $ubg_color
# client.placeholder      #000000 #0c0c0c #ffffff #000000   #0c0c0c
```

bar customization

Preview:

- Copy the template to your home folder like below:

```
cp -rvf /etc/i3status.conf ~/.config/i3/i3status.conf
```

- Indicate the config file in `~/.config/i3/config`

```
status_command  i3status --config ~/.config/i3/i3status.conf
```

That means run the `i3status --config ~/.config/i3/i3status.conf` command and get back the standard output as the status bar content.

- Here is the `i3status.config` sample with comment

  You need to restart `i3` to take effect after changing this file.

  The icon below you can copy from `https://fontawesome.com/cheatsheet`

```
general {
        # `false` to use the bar color by default
        colors = false
        # All commands refresh interval in seconds
        interval = 2
}

order += "wireless _first_"
order += "ethernet _first_"
order += "battery 0"
order += "disk /"
order += "memory"
order += "volume master"
order += "tztime local"

wireless _first_ {
        format_up = " %essid"
        format_down = ""
}

ethernet _first_ {
        format_up = "d(%speed)"
        format_down = ""
}

battery 0 {
        format = " %status %percentage"
        format_down = ""
        status_bat = ""
        status_chr = ""
        status_full = " "
}

disk "/" {
        # format = " %avail"
        format = " [ %free / %total ]"
        prefix_type = "custom"
```

```
}

memory {
        format = " %used / %total"
        memory_used_method = "memavailable"
}

volume master {
    format = " %volume"
    format_muted = ""
    device = "default"
    mixer = "Master"
    mixer_idx = 0
}

tztime local {
        format = " %Y-%m-%d %H:%M:%S "
}
```

- Extra `bar` customization with comment

```
#
================================================================================

# i3 status control
#
# Start i3bar to display a workspace bar (plus the system information
i3status
# finds out, if available)
#
================================================================================

set $bar_bg_color           #1C1C1CD9
set $bar_text_color         #FFFFFFCC
set $bar_itext_color        #616161
set $bar_ws_bg_color        #551E22
set $bar_ws_bg_color_2      #E66B17
set $bar_separator_color    #E66B17

bar {
    # Run the `i3status` command and get back the standard output as
the bar content
    status_command  i3status --config ~/.config/i3/i3status.conf

    # Bar transparent effect
    # Even enabled this flag, you still need to apply the
"transparency hex" value to
    # the bar color for getting the real transparency effect. You can
find the hex code
    # from here:
    # https://gist.github.com/lopspower/03fb1cc0ac9f32ef38f4
    i3bar_command i3bar --transparency

    # Stay on the top/bottom
    position    top

    # Render to the primary screen only
    output  primary
```

```
    # Hide the appliation icon tray area (which always stay on the
most-right)
    tray_output none
    # tray_output primary

    #
    font pango:SourceCodePro 10
    # font pango:monospace 10

    # The separator
    # separator_symbol    "|"
    separator_symbol    " 🔋 "

    # Workspace
    workspace_buttons   yes
    workspace_min_width 25
    # strip_workspace_numbers yes
    # binding_mode_indicator no
    colors {
        background  $bar_bg_color
        statusline  $bar_text_color
        separator   $bar_separator_color

        focused_workspace  $bar_ws_bg_color_2   $bar_ws_bg_color_2
$bar_text_color     $bar_ws_bg_color_2
        active_workspace   $bar_ws_bg_color_2   $bar_ws_bg_color_2
$bar_itext_color    $bar_ws_bg_color_2
        inactive_workspace $bar_ws_bg_color     $bar_ws_bg_color
$bar_itext_color    $bar_ws_bg_color
        urgent_workspace   $bar_ws_bg_color     $bar_ws_bg_color
$bar_text_color     $bar_ws_bg_color
    }
}
```

bar customization with i3blocks

`i3status` is simple, but you can get more flexible status output with `i3blocks`, as `i3blocks` allows you to run different script/program for each part you want to display on the status bar.

For example:

You can customize the output with different status and icon based on the `acpi -b` program result:

```
# Run `acpi -b` when charging
Battery 0: Charging, 72%, 00:31:20 until charged

# Run `acpi -b` when discharging
Battery 0: Discharging, 71%, 03:36:05 remaining
```

So, you can use diferent awesome font icon to show different status.

Let's step-by-step to make that happen.

- Use `i3blocks` as the `status line command`

  Copy the configuration template to your home folder like below:

  ```
  cp -rvf /etc/i3blocks.conf ~/.config/i3
  ```

  `vim ~/.config/i3/config` and change the line below:

  ```
  bar {
      # Run the `i3block` command and get back the standard
      # output as the bar content
      status_command  i3blocks -c ~/.config/i3/i3blocks.conf
  }
  ```

  Reload `i3`, the new status content should be display on the right-top on your screen.

- Configuration

  `i3blocks.conf` is super simple, plz have a look:

  ```
  [test]
  # You can define any property you want, and the property will become
  the
  # environment variable in the `command` line
  my_var="(You look great today:)"

  # The command you want to run, the result will show to the status bar
  command=echo "OMG $my_var"

  # Command running interval settings:
  # `onece`: only run once, then never run
  # `x`: run per `x` second to refresh the conent
  interval=once
  ```

  That's it, super simple!!!:)

Bar customization with `polybar`

- Install

  ```
  yay -S polybar
  ```

- Copy the default config to your home folder like below:

  ```
  mkdir ~/.config/polybar
  cp -rvf /usr/share/doc/polybar/config  ./polybar/
  ```

  Also, make sure to change the `[bar/example]` to `[bar/i3-bar]` in `~/.config/polybar/config`.

- Create `~/scripts/restart-polybar.sh` with the following content:

```
#!/usr/bin/bash

killall polybar

while pgrep -x > /dev/null; do sleep 1; done

polybar i3-bar
```

- Make some changes to `~/.config/i3/config`
  - Add this:

```
# Load polybar
exec_always --no-startup-id ~/scripts/restart-polybar.sh &
```

  - Remove all related settings about the `i3 status bar`

```
#
# ###########################################################

# i3 status control
#
# Start i3bar to display a workspace bar (plus the system
information i3status
# finds out, if available)
#
# ###########################################################

bar {
    # ......
}
```

i3 config sample with comment

Here is the fully comment `i3` configuration sample file:

```
#
# ============================================================================

# Set `$mod` to `Mod4`(WinKey or CmdKey)
#
# ============================================================================

set $mod Mod4
set $alt Mod1

#
# ============================================================================

# This font is widely installed, provides lots of unicode glyphs, right-
to-left
# text rendering and scalability on retina/hidpi displays (thanks to
pango).
# font pango:DejaVu Sans Mono 8
#
# ============================================================================

font pango:SauceCodePro Nerd Font 11

#
# ============================================================================

# Start up programs
#
# ============================================================================


# Set keyboard repeast rate: 200ms for the first delay, 40Hz for repeat
rate
exec --no-startup-id xset r rate 200 40

# xss-lock grabs a logind suspend inhibit lock and will use i3lock to
lock the
# screen before suspend. Use loginctl lock-session to lock your screen.
# exec --no-startup-id xss-lock --transfer-sleep-lock -- i3lock --nofork

# NetworkManager is the most popular way to manage wireless networks on
Linux,
# and nm-applet is a desktop environment-independent system tray GUI for
it.
# exec --no-startup-id nm-applet

# Load my custom keymapping
exec_always sleep 1; xmodmap ~/.Xmodmap

# Load compton renderer. After `compton` runs, `xsetroot -solid 'color'`
# won't work!!!
```

```
exec_always compton --active-opacity 0.99 --inactive-opacity 0.9

# Load polybar
# exec_always --no-startup-id ~/scripts/restart-polybar.sh &

# Load chinese input method
# exec_always fcitx

# Enable notification server
exec --no-startup-id dunst -config ~/.config/dunst/dunstrc

# Load wallpaper
exec_always feh --bg-scale ~/Photos/wallpaper/tron-2.png
# exec_always feh --bg-scale ~/Photos/wallpaper/arch-2.png

#
================================================================================

# Audio and volume control
#
================================================================================

# Use pactl to adjust volume in PulseAudio.
set $refresh_i3status killall -SIGUSR1 i3status
bindsym XF86AudioRaiseVolume exec --no-startup-id pactl set-sink-volume
@DEFAULT_SINK@ +5% && $refresh_i3status
bindsym XF86AudioLowerVolume exec --no-startup-id pactl set-sink-volume
@DEFAULT_SINK@ -5% && $refresh_i3status
bindsym XF86AudioMute exec --no-startup-id pactl set-sink-mute
@DEFAULT_SINK@ toggle && $refresh_i3status
bindsym XF86AudioMicMute exec --no-startup-id pactl set-source-mute
@DEFAULT_SOURCE@ toggle && $refresh_i3status


#
================================================================================

# Screen brightness control
#
================================================================================

bindsym XF86MonBrightnessUp exec --no-startup-id ~/scripts/mac-light-
controller Screen +
bindsym XF86MonBrightnessDown exec --no-startup-id ~/scripts/mac-light-
controller Screen -
bindsym XF86KbdBrightnessUp exec --no-startup-id ~/scripts/mac-light-
controller Keyboard +
bindsym XF86KbdBrightnessDown exec --no-startup-id ~/scripts/mac-light-
controller Keyboard -
```

```
#
# ============================================================================

# Special key mapping
#
# ============================================================================


# `cmd+c` -> `ctrl+c` (Copy)
# bindsym --release $mod+c exec --no-startup-id xdotool key --
clearmodifiers ctrl+c

# `cmd+v` -> `ctrl+v` (Paste)
# bindsym --release $mod+v exec --no-startup-id xdotool key --
clearmodifiers ctrl+v

# `cmd+f` -> `ctrl+f` (Find)
# bindsym --release $mod+f exec --no-startup-id xdotool key --
clearmodifiers ctrl+f

# `cmd+t` -> `ctrl+t` (Open new tab)
# bindsym --release $mod+t exec --no-startup-id xdotool key --
clearmodifiers ctrl+t

# `cmd+w` -> `ctrl+w` (Close current tab)
# bindsym --release $mod+w exec --no-startup-id xdotool key --
clearmodifiers ctrl+w


#
# ============================================================================

# Window Navigation (Vim style)
#
# ============================================================================

set $up k
set $down j
set $left h
set $right l

# Change window focus by HJKL
bindsym $mod+$left focus left
bindsym $mod+$down focus down
bindsym $mod+$up focus up
bindsym $mod+$right focus right

# Change window focus by arrow key
bindsym $mod+Left focus left
bindsym $mod+Down focus down
bindsym $mod+Up focus up
```

```
bindsym $mod+Right focus right

# Move focused window by HJKL
bindsym $mod+Shift+$left move left
bindsym $mod+Shift+$down move down
bindsym $mod+Shift+$up move up
bindsym $mod+Shift+$right move right

# Move focused window by arrow key
bindsym $mod+Shift+Left move left
bindsym $mod+Shift+Down move down
bindsym $mod+Shift+Up move up
bindsym $mod+Shift+Right move right

# kill focused window
bindsym $mod+q kill


#
================================================================================

# Resize window
#
================================================================================

# resize window (you can also use the mouse for that)
set $resize_unit 5
mode "resize" {
        # These bindings trigger as soon as you enter the resize mode

        # Pressing left will shrink the window’s width.
        # Pressing right will grow the window’s width.
        # Pressing up will shrink the window’s height.
        # Pressing down will grow the window’s height.
        bindsym $left       resize shrink width $resize_unit px or
$resize_unit ppt
        bindsym $down       resize grow height $resize_unit px or
$resize_unit ppt
        bindsym $up         resize shrink height $resize_unit px or
$resize_unit ppt
        bindsym $right      resize grow width $resize_unit px or
$resize_unit ppt

        # back to normal: Enter or Escape
        bindsym Return mode "default"
        bindsym Escape mode "default"
}
bindsym Shift+$mod+r mode "resize"


#
```

```
# ============================================================================

# Layout control
#
# ============================================================================



# Toggle fullscreen on current focused window
# bindsym Control+$mod+f fullscreen toggle
bindsym $mod+f fullscreen toggle

# use Mouse+$mod to drag floating windows to their wanted position
floating_modifier $mod

# toggle tiling / floating
bindsym $mod+Shift+space floating toggle

# change container layout (stacked, tabbed, toggle split)
# bindsym $mod+s layout stacking
# bindsym $mod+w layout tabbed
# bindsym $mod+e layout toggle split
bindsym $mod+e layout toggle tabbed split

# split in horizontal orientation
# bindsym $mod+h split h

# split toggle between splitv and splith
bindsym $mod+v split toggle

# Open window Mode: Open specified layout of windows based on the number
mode "Open multiple windows in the specific layout" {
        # Pressing left will shrink the window’s width.
        # Pressing right will grow the window’s width.
        # Pressing up will shrink the window’s height.
        # Pressing down will grow the window’s height.
        # bindsym $left       resize shrink width $resize_unit px or
$resize_unit ppt
        # bindsym $down       resize grow height $resize_unit px or
$resize_unit ppt
        # bindsym $up         resize shrink height $resize_unit px or
$resize_unit ppt
        # bindsym $right      resize grow width $resize_unit px or
$resize_unit ppt
        bindsym 2  exec ~/scripts/2w.sh; mode "default"
        bindsym 3  exec ~/scripts/3w.sh; mode "default"

        # back to normal: Enter or Escape
        bindsym Return mode "default"
        bindsym Escape mode "default"
}
bindsym Shift+$mod+o mode "Open multiple windows in the specific layout"
```

```
#
# ============================================================================

# Workspace control
#
# ============================================================================


# Workspace name 1 ~ 10
set $ws1 "1"
set $ws2 "2"
set $ws3 "3"
set $ws4 "4"
set $ws5 "5"
set $ws6 "6"
set $ws7 "7"
set $ws8 "8"
set $ws9 "9"
# set $ws10 "10"

# Switch to workspace
bindsym $mod+1 workspace number $ws1
bindsym $mod+2 workspace number $ws2
bindsym $mod+3 workspace number $ws3
bindsym $mod+4 workspace number $ws4
bindsym $mod+5 workspace number $ws5
bindsym $mod+6 workspace number $ws6
bindsym $mod+7 workspace number $ws7
bindsym $mod+8 workspace number $ws8
bindsym $mod+9 workspace number $ws9
# bindsym $mod+0 workspace number $ws10

# Move focused window to particular workspace
bindsym $mod+Shift+1 move container to workspace number $ws1
bindsym $mod+Shift+2 move container to workspace number $ws2
bindsym $mod+Shift+3 move container to workspace number $ws3
bindsym $mod+Shift+4 move container to workspace number $ws4
bindsym $mod+Shift+5 move container to workspace number $ws5
bindsym $mod+Shift+6 move container to workspace number $ws6
bindsym $mod+Shift+7 move container to workspace number $ws7
bindsym $mod+Shift+8 move container to workspace number $ws8
bindsym $mod+Shift+9 move container to workspace number $ws9
bindsym $mod+Shift+0 move container to workspace number $ws10



#
# ============================================================================
```

```
# Program launch shortcuts
#
# ================================================================================



# Terminal
bindsym $mod+Return exec alacritty

# Browser
set $browser google-chrome-stable
bindsym $mod+b exec $browser

# Open file manager
# bindsym $mod+f exec "pcmanfm --new-win $(pwd) &"

# Open ranger
bindsym $mod+r exec "alacritty --command ranger"

# Take screenshot
bindsym --release $alt+$mod+p exec "scrot -s
~/Screenshots/screenshot-$(date +%F_%T).png -e 'xclip -selection c -t
image/png < $f'"
bindsym Control+$mod+p exec "scrot ~/Screenshots/screenshot-$(date
+%F_%T).png -e 'xclip -selection c -t image/png < $f'"

# start dmenu (a program launcher)
#
# There also is the (new) i3-dmenu-desktop which only displays
applications
# shipping a .desktop file. It is a wrapper around dmenu, so you need
that
# installed.
# bindsym $mod+d exec --no-startup-id i3-dmenu-desktop
# bindsym $mod+d exec dmenu_run
# bindsym $mod+d exec dmenu_run
bindsym $alt+space exec --no-startup-id i3-dmenu-desktop

# reload the configuration file
# bindsym $mod+Shift+c reload
# restart i3 inplace (preserves your layout/session, can be used to
upgrade i3)
# bindsym $mod+Shift+r restart
# exit i3 (logs you out of your X session)
bindsym $mod+Shift+e exec "i3-nagbar -t warning -m 'You pressed the exit
shortcut. Do you really want to exit i3? This will end your X session.' -
B 'Yes, exit i3' 'i3-msg exit'"

# Toggle `screenkey`
bindsym $mod+s exec --no-startup-id ~/scripts/toggle-screenkey.sh

# Launch WxWork
```

```
bindsym $mod+w exec --no-startup-id ~/scripts/wechat_work.sh

# Zoom
bindsym $mod+z exec --no-startup-id ~/scripts/startzoom.sh

#
# ================================================================================

# Force some launch programs open in floating mode or control window open
style
#
# ================================================================================


# Make sure dialog window not outside of screen!
floating_maximum_size 1920 x 1080

for_window [class="(?i)google-chrome"] border pixel 1
for_window [class="(?i)vlc"] border pixel 1
# for_window [class="(?i)4kvideodownloader-bin"] floating enable border
pixel 1
for_window [class="(?i)VirtualBox Machine"] floating enable
for_window [class="(?i)VirtualBox Manager"] floating enable
for_window [title="^4K Video Downloader"] floating enable
for_window [title="SimpleScreenRecord"] floating enable border pixel 1
for_window [class="(?i)wxwork.exe"] floating enable border pixel 1
for_window [class="(?i)slack"] floating enable border pixel 1
for_window [title="^Firewall Configuration"] floating enable
for_window [title="^glxgears"] floating enable
for_window [class="(?i)zoom"] floating enable

# for_window [window_type="dialog"] floating enbale border pixel 1
# for_window [class="Skype"] floating enable border normal
# for_window [class="(?i)virtualbox"] floating enable border normal


#
# ================================================================================

# Display styles control
#
# ================================================================================


# Thin border
for_window [class="^.*"] border 1pixel
new_window 1pixel

# With gaps
gaps inner 10
```

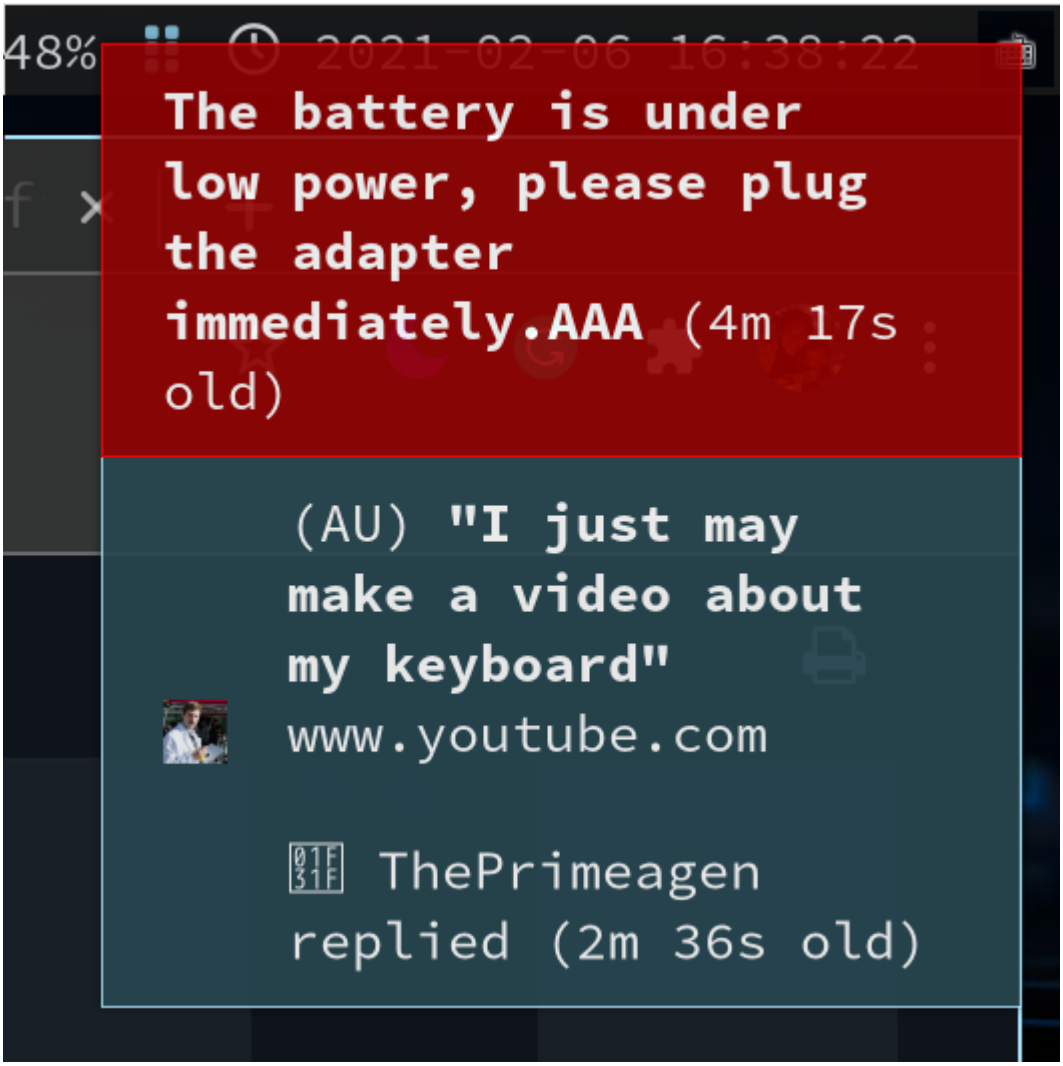
```
# Window colors

# Light blue
set $bg_color            #ACE6FE
set $ibg_color           #014775

set $border_color        #ACE6FE
set $text_color          #1C1C1C
set $itext_color         #FFFFFF
set $ubg_color           #000000

#                        border           background          text
indicator
client.focused           $border_color    $bg_color           $text_color
$bg_color
client.focused_inactive $ibg_color       $ibg_color          $itext_color
$ibg_color
client.unfocused         $ibg_color       $ibg_color          $itext_color
$ibg_color
client.urgent            $ubg_color       $ubg_color          $text_color
$ubg_color
# client.placeholder      #000000 #0c0c0c #ffffff #000000   #0c0c0c


#
================================================================================

# i3 status control
#
# Start i3bar to display a workspace bar (plus the system information
i3status
# finds out, if available)
#
================================================================================

set $bar_bg_color           #1C1C1CD9
set $bar_text_color         #FFFFFFCC
set $bar_itext_color        #000000CC
set $bar_ws_text_color      #1C1C1CD9
set $bar_ws_bg_color        #274650
set $bar_ws_bg_color_2      #7ab4cb
set $bar_separator_color    #7ab4cb

bar {
    # Run the `i3block` command and get back the standard output as the
bar content
    status_command i3status -c ~/.config/i3/i3status.conf
    # status_command  i3blocks -c ~/.config/i3/i3blocks.conf

    # Bar transparent effect
    # Even enabled this flag, you still need to apply the "transparency
```

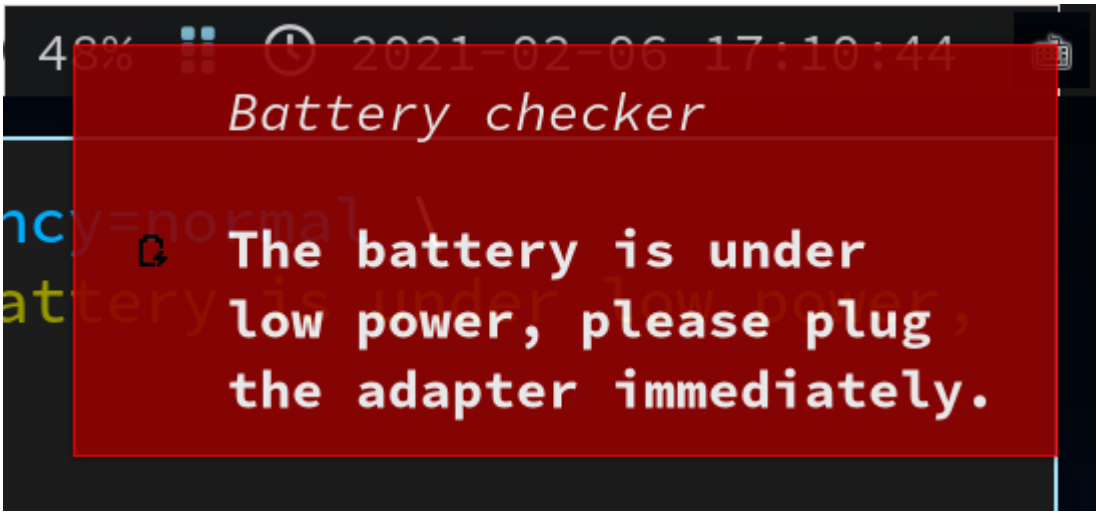
```
hex" value to
    # the bar color for getting the real transparency effect. You can
find the hex code
    # from here:
    # https://gist.github.com/lopspower/03fb1cc0ac9f32ef38f4
    i3bar_command i3bar --transparency

    # Stay on the top/bottom
    position    top

    # Render to the primary screen only
    output  primary

    # Hide the appliation icon tray area (which always stay on the most-
right)
    # tray_output none
    tray_output primary

    #
    font pango:SauceCodePro Nerd Font 10
    # font pango:monospace 10

    # The separator
    # separator_symbol    "|"
    separator_symbol    " ⋮ "
    # separator_symbol    " ⣿ "

    # Workspace
    workspace_buttons   yes
    workspace_min_width 25
    # strip_workspace_numbers yes
    # binding_mode_indicator no
    colors {
        background  $bar_bg_color
        statusline  $bar_text_color
        separator   $bar_separator_color

        focused_workspace  $bar_ws_bg_color_2   $bar_ws_bg_color_2
$bar_ws_text_color  $bar_ws_bg_color_2
        active_workspace   $bar_ws_bg_color_2   $bar_ws_bg_color_2
$bar_itext_color    $bar_ws_bg_color_2
        inactive_workspace $bar_ws_bg_color     $bar_ws_bg_color
$bar_itext_color    $bar_ws_bg_color
        urgent_workspace   $bar_ws_bg_color     $bar_ws_bg_color
$bar_text_color     $bar_ws_bg_color
    }
}


#
================================================================================
```

```
# System control
#
================================================================================

set $mode_system System (l) Lock, (o) Logout, (r) Reboot, (Shift+s)
Shutdown
mode "$mode_system" {
    bindsym l exec --no-startup-id i3exit lock, mode "default"
    bindsym o exec --no-startup-id i3exit logout, mode "default"
    bindsym r exec --no-startup-id i3exit reboot, mode "default"
    bindsym Shift+s exec --no-startup-id i3exit shutdown, mode "default"
    bindsym Return mode "default"
    bindsym Escape mode "default"
}

bindsym $mod+0 mode "$mode_system"


#
================================================================================

# Unknow purpose yet
#
================================================================================


# move the currently focused window to the scratchpad
bindsym $mod+Shift+minus move scratchpad

# Show the next scratchpad window or hide the focused scratchpad window.
# If there are multiple scratchpad windows, this command cycles through
them.
bindsym $mod+minus scratchpad show


#########################################################################
# automatically start i3-config-wizard to offer the user to create a
# keysym-based config which used their favorite modifier (alt or windows)
#
# i3-config-wizard will not launch if there already is a config file
# in ~/.config/i3/config (or $XDG_CONFIG_HOME/i3/config if set) or
# ~/.i3/config.
#
# Please remove the following exec line:
#########################################################################
exec i3-config-wizard
```

Notification

Preview:





`Desktop Notification` allows user send custom notifcation to the `DE/WM`. It needs two parts:

- Notifcation Server

  `dunst` is a customizable and lightweight notification-daemon.

  ```
  sudo pacmanc --sync --refresh dunst
  ```

- Notifcation client (program)

  `notify-send` is a built-in program to send desktop notifications.

Custom configiration

- Copy the template to your home folder like below:

  ```
  cp -rvf /usr/share/dunst ~/.config
  ```

- Lanuch it in `i3` session

  `vim ~/.config/i3/config` with the following settings:

  ```
  # Enable notification server
  exec --no-startup-id dunst
  ```

- Here is the `~/.config/dunst/dunstrc` sample with comment

  You need to kill the exists process and restart `i3` like below to take effect after changing this file.

  ```
  killall dunst && i3-msg restart
  ```

```
# The geometry of the window:
#   [{width}]x{height}[+/-{x}+/-{y}]
# The geometry of the message window.
# The height is measured in number of notifications everything else
# in pixels.  If the width is omitted but the height is given
# ("-geometry x2"), the message window expands over the whole screen
# (dmenu-like).  If width is 0, the window expands to the longest
# message displayed.  A positive x is measured from the left, a
# negative from the right side of the screen.  Y is measured from
# the top and down respectively.
# The width can be negative.  In this case the actual width is the
# screen width minus the width defined in within the geometry option.
geometry = "500x5-20+20"

# Show how many messages are currently hidden (because of geometry).
indicate_hidden = yes

# Shrink window if it's smaller than the width.  Will be ignored if
# width is 0.
shrink = true

# The transparency of the window.  Range: [0; 100].
# This option will only work if a compositing window manager is
# present (e.g. xcompmgr, compiz, etc.).
transparency = 10

# Padding between text and separator.
padding = 15

# Horizontal padding.
horizontal_padding = 30

# The height of the entire notification.  If the height is smaller
# than the font height and padding combined, it will be raised
# to the font height and padding.
notification_height = 0
```

```
# Draw a line of "separator_height" pixel height between two
# notifications.
# Set to 0 to disable.
separator_height = 1

# Padding between text and separator.
padding = 15

# Horizontal padding.
horizontal_padding = 30

# Defines width in pixels of frame around the notification window.
# Set to 0 to disable.
frame_width = 1

# Defines color of the frame around the notification window.
frame_color = "#ACE6FE"

# Font settings
font = SourceCodePro 10

# The spacing between lines.  If the height is smaller than the
# font height, it will get raised to the font height.
line_height = 0

# Browser for opening urls in context menu.
browser = /usr/bin/google-chrome-stable

# Paths to default icons.
# Icons are set in the option icon_path. Status and devices icons are
needed.
# By default, Dunst looks for the `gnome-icon-theme` icons. For
example, to use
# `adwaita-icon-theme` (gnome-icon-theme's successor), instead:
icon_path =
/usr/share/icons/Adwaita/16x16/status/:/usr/share/icons/Adwaita/16x16/
```

- Test the notification:
  - Use `notify-send`:

```
# Low  urgency level
notify-send --urgency=low \
    "The battery is under low power, please plug the adapter
immediately."

# Normal urgency level
notify-send "The battery is under low power, please plug the
adapter immediately."
notify-send --urgency=normal \
    "The battery is under low power, please plug the adapter
immediately."

# Critical urgency level
notify-send --urgency=critical \
    "The battery is under low power, please plug the adapter
immediately."

# Use icon
notify-send  --icon=battery-level-0-charging-symbolic.symbolic \
    "The battery is under low power, please plug the adapter
immediately."
```

  - Use `dunstify`:

```
dunstify --appname="Battery checker" \
         --urgency=critical --icon=battery-level-0-charging-
symbolic.symbolic \
         "The battery is under low power, please plug the adapter
immediately."
```

i3 notification config sample

Here is the fully comment `notification` configuration sample file:

```
[global]
    ### Display ###

    # Which monitor should the notifications be displayed on.
    monitor = 0

    # Display notification on focused monitor.  Possible modes are:
    #   mouse: follow mouse pointer
    #   keyboard: follow window with keyboard focus
    #   none: don't follow anything
    #
    # "keyboard" needs a window manager that exports the
    # _NET_ACTIVE_WINDOW property.
    # This should be the case for almost all modern window managers.
    #
    # If this option is set to mouse or keyboard, the monitor option
    # will be ignored.
    follow = mouse

    # The geometry of the window:
    #   [{width}]x{height}[+/-{x}+/-{y}]
    # The geometry of the message window.
    # The height is measured in number of notifications everything else
    # in pixels.  If the width is omitted but the height is given
    # ("-geometry x2"), the message window expands over the whole screen
    # (dmenu-like).  If width is 0, the window expands to the longest
    # message displayed.  A positive x is measured from the left, a
    # negative from the right side of the screen.  Y is measured from
    # the top and down respectively.
    # The width can be negative.  In this case the actual width is the
    # screen width minus the width defined in within the geometry option.
    geometry = "500x5-20+20"

    # Show how many messages are currently hidden (because of geometry).
    indicate_hidden = yes

    # Shrink window if it's smaller than the width.  Will be ignored if
    # width is 0.
    shrink = true

    # The transparency of the window.  Range: [0; 100].
    # This option will only work if a compositing window manager is
    # present (e.g. xcompmgr, compiz, etc.).
    transparency = 10

    # The height of the entire notification.  If the height is smaller
    # than the font height and padding combined, it will be raised
    # to the font height and padding.
    notification_height = 0

    # Draw a line of "separator_height" pixel height between two
```

```
    # notifications.
    # Set to 0 to disable.
    separator_height = 1

    # Padding between text and separator.
    padding = 15

    # Horizontal padding.
    horizontal_padding = 30

    # Defines width in pixels of frame around the notification window.
    # Set to 0 to disable.
    frame_width = 1

    # Defines color of the frame around the notification window.
    frame_color = "#ACE6FE"

    # Define a color for the separator.
    # possible values are:
    #  * auto: dunst tries to find a color fitting to the background;
    #  * foreground: use the same color as the foreground;
    #  * frame: use the same color as the frame;
    #  * anything else will be interpreted as a X color.
    separator_color = frame

    # Sort messages by urgency.
    sort = yes

    # Don't remove messages, if the user is idle (no mouse or keyboard
input)
    # for longer than idle_threshold seconds.
    # Set to 0 to disable.
    # A client can set the 'transient' hint to bypass this. See the rules
    # section for how to disable this if necessary
    idle_threshold = 120

    ### Text ###

    font = SauceCodePro Nerd Font 10

    # The spacing between lines.  If the height is smaller than the
    # font height, it will get raised to the font height.
    line_height = 0

    # Possible values are:
    # full: Allow a small subset of html markup in notifications:
    #        <b>bold</b>
    #        <i>italic</i>
    #        <s>strikethrough</s>
    #        <u>underline</u>
    #
```

```
    #        For a complete reference see
    #        <https://developer.gnome.org/pango/stable/pango-
Markup.html>.
    #
    # strip: This setting is provided for compatibility with some broken
    #        clients that send markup even though it's not enabled on the
    #        server. Dunst will try to strip the markup but the parsing
is
    #        simplistic so using this option outside of matching rules
for
    #        specific applications *IS GREATLY DISCOURAGED*.
    #
    # no:    Disable markup parsing, incoming notifications will be
treated as
    #        plain text. Dunst will not advertise that it has the body-
markup
    #        capability if this is set as a global setting.
    #
    # It's important to note that markup inside the format option will be
parsed
    # regardless of what this is set to.
    markup = full

    # The format of the message.  Possible variables are:
    #   %a  appname
    #   %s  summary
    #   %b  body
    #   %i  iconname (including its path)
    #   %I  iconname (without its path)
    #   %p  progress value if set ([  0%] to [100%]) or nothing
    #   %n  progress value if set without any extra characters
    #   %%  Literal %
    # Markup is allowed
    # format = "<b>%s</b>\n%b"
    format = "<i>%a</i>\n\n<b>%s</b>\n%b"

    # Alignment of message text.
    # Possible values are "left", "center" and "right".
    alignment = left

    # Vertical alignment of message text and icon.
    # Possible values are "top", "center" and "bottom".
    vertical_alignment = center

    # Show age of message if message is older than show_age_threshold
    # seconds.
    # Set to -1 to disable.
    show_age_threshold = 60

    # Split notifications into multiple lines if they don't fit into
    # geometry.
```

```
    word_wrap = yes

    # When word_wrap is set to no, specify where to make an ellipsis in
long lines.
    # Possible values are "start", "middle" and "end".
    ellipsize = middle

    # Ignore newlines '\n' in notifications.
    ignore_newline = no

    # Stack together notifications with the same content
    stack_duplicates = true

    # Hide the count of stacked notifications with the same content
    hide_duplicate_count = false

    # Display indicators for URLs (U) and actions (A).
    show_indicators = yes

    ### Icons ###

    # Align icons left/right/off
    icon_position = left

    # Scale small icons up to this size, set to 0 to disable. Helpful
    # for e.g. small files or high-dpi screens. In case of conflict,
    # max_icon_size takes precedence over this.
    min_icon_size = 0

    # Scale larger icons down to this size, set to 0 to disable
    max_icon_size = 32

    # Paths to default icons.
    icon_path =
/usr/share/icons/Adwaita/16x16/status/:/usr/share/icons/Adwaita/16x16/devi


    ### History ###

    # Should a notification popped up from history be sticky or timeout
    # as if it would normally do.
    sticky_history = yes

    # Maximum amount of notifications kept in history
    history_length = 20

    ### Misc/Advanced ###

    # dmenu path.
    dmenu = /usr/bin/dmenu -p dunst:
```

```
    # Browser for opening urls in context menu.
    browser = /usr/bin/google-chrome-stable

    # Always run rule-defined scripts, even if the notification is
suppressed
    always_run_script = true

    # Define the title of the windows spawned by dunst
    title = Dunst

    # Define the class of the windows spawned by dunst
    class = Dunst

    # Print a notification on startup.
    # This is mainly for error detection, since dbus (re-)starts dunst
    # automatically after a crash.
    startup_notification = false

    # Manage dunst's desire for talking
    # Can be one of the following values:
    #  crit: Critical features. Dunst aborts
    #  warn: Only non-fatal warnings
    #  mesg: Important Messages
    #  info: all unimportant stuff
    # debug: all less than unimportant stuff
    verbosity = mesg

    # Define the corner radius of the notification window
    # in pixel size. If the radius is 0, you have no rounded
    # corners.
    # The radius will be automatically lowered if it exceeds half of the
    # notification height to avoid clipping text and/or icons.
    corner_radius = 0

    # Ignore the dbus closeNotification message.
    # Useful to enforce the timeout set by dunst configuration. Without
this
    # parameter, an application may close the notification sent before
the
    # user defined timeout.
    ignore_dbusclose = false

    ### Legacy

    # Use the Xinerama extension instead of RandR for multi-monitor
support.
    # This setting is provided for compatibility with older nVidia
drivers that
    # do not support RandR and using it on systems that support RandR is
highly
    # discouraged.
```

```
    #
    # By enabling this setting dunst will not be able to detect when a
monitor
    # is connected or disconnected which might break follow mode if the
screen
    # layout changes.
    force_xinerama = false

    ### mouse

    # Defines list of actions for each mouse event
    # Possible values are:
    # * none: Don't do anything.
    # * do_action: If the notification has exactly one action, or one is
marked as default,
    #              invoke it. If there are multiple and no default, open
the context menu.
    # * close_current: Close current notification.
    # * close_all: Close all notifications.
    # These values can be strung together for each mouse event, and
    # will be executed in sequence.
    mouse_left_click = close_current
    mouse_middle_click = do_action, close_current
    mouse_right_click = close_all

# Experimental features that may or may not work correctly. Do not expect
them
# to have a consistent behaviour across releases.
[experimental]
    # Calculate the dpi to use on a per-monitor basis.
    # If this setting is enabled the Xft.dpi value will be ignored and
instead
    # dunst will attempt to calculate an appropriate dpi value for each
monitor
    # using the resolution and physical size. This might be useful in
setups
    # where there are multiple screens with very different dpi values.
    per_monitor_dpi = false

[shortcuts]

    # Shortcuts are specified as [modifier+][modifier+]...key
    # Available modifiers are "ctrl", "mod1" (the alt-key), "mod2",
    # "mod3" and "mod4" (windows-key).
    # Xev might be helpful to find names for keys.

    # Close notification.
    close = ctrl+space

    # Close all notifications.
    close_all = ctrl+shift+space
```

```
    # Redisplay last message(s).
    # On the US keyboard layout "grave" is normally above TAB and left
    # of "1". Make sure this key actually exists on your keyboard layout,
    # e.g. check output of 'xmodmap -pke'
    history = ctrl+grave

    # Context menu.
    context = ctrl+shift+period

[urgency_low]
    # IMPORTANT: colors have to be defined in quotation marks.
    # Otherwise the "#" and following would be interpreted as a comment.
    background = "#222222"
    foreground = "#888888"
    timeout = 10
    # Icon for notifications with low urgency, uncomment to enable
    #icon = /path/to/icon

[urgency_normal]
    background = "#274650"
    foreground = "#ffffff"
    timeout = 10
    # Icon for notifications with normal urgency, uncomment to enable
    #icon = /path/to/icon

[urgency_critical]
    background = "#900000"
    foreground = "#ffffff"
    frame_color = "#ff0000"
    timeout = 0
    # Icon for notifications with critical urgency, uncomment to enable
    #icon = /path/to/icon

# Every section that isn't one of the above is interpreted as a rules to
# override settings for certain messages.
#
# Messages can be matched by
#    appname (discouraged, see desktop_entry)
#    body
#    category
#    desktop_entry
#    icon
#    match_transient
#    msg_urgency
#    stack_tag
#    summary
#
# and you can override the
#    background
#    foreground
```

```
#    format
#    frame_color
#    fullscreen
#    new_icon
#    set_stack_tag
#    set_transient
#    timeout
#    urgency
#
# Shell-like globbing will get expanded.
#
# Instead of the appname filter, it's recommended to use the
desktop_entry filter.
# GLib based applications export their desktop-entry name. In comparison
to the appname,
# the desktop-entry won't get localized.
#
# SCRIPTING
# You can specify a script that gets run when the rule matches by
# setting the "script" option.
# The script will be called as follows:
#   script appname summary body icon urgency
# where urgency can be "LOW", "NORMAL" or "CRITICAL".
#
# NOTE: if you don't want a notification to be displayed, set the format
# to "".
# NOTE: It might be helpful to run dunst -print in a terminal in order
# to find fitting options for rules.

# Disable the transient hint so that idle_threshold cannot be bypassed
from the
# client
#[transient_disable]
#    match_transient = yes
#    set_transient = no
#
# Make the handling of transient notifications more strict by making them
not
# be placed in history.
#[transient_history_ignore]
#    match_transient = yes
#    history_ignore = yes

# fullscreen values
# show: show the notifications, regardless if there is a fullscreen
window opened
# delay: displays the new notification, if there is no fullscreen window
active
#        If the notification is already drawn, it won't get undrawn.
# pushback: same as delay, but when switching into fullscreen, the
notification will get
```

```
#           withdrawn from screen again and will get delayed like a new
notification
#[fullscreen_delay_everything]
#    fullscreen = delay
#[fullscreen_show_critical]
#    msg_urgency = critical
#    fullscreen = show

#[espeak]
#    summary = "*"
#    script = dunst_espeak.sh

#[script-test]
#    summary = "*script*"
#    script = dunst_test.sh

#[ignore]
#    # This notification will not be displayed
#    summary = "foobar"
#    format = ""

#[history-ignore]
#    # This notification will not be saved in history
#    summary = "foobar"
#    history_ignore = yes

#[skip-display]
#    # This notification will not be displayed, but will be included in
the history
#    summary = "foobar"
#    skip_display = yes

#[signed_on]
#    appname = Pidgin
#    summary = "*signed on*"
#    urgency = low
#
#[signed_off]
#    appname = Pidgin
#    summary = *signed off*
#    urgency = low
#
#[says]
#    appname = Pidgin
#    summary = *says*
#    urgency = critical
#
#[twitter]
#    appname = Pidgin
#    summary = *twitter.com*
#    urgency = normal
```

```
#
#[stack-volumes]
#    appname = "some_volume_notifiers"
#    set_stack_tag = "volume"
#
# vim: ft=cfg
```

How to setup low-battery notifications

- `vim ~/scripts/check-battery.sh` with the following content:

```
#!/bin/bash

# Because `dunstify` needs the `dbus-daemon` to send the
notification,
# that's why we need to have the below settings.
#
# `DBUS_SESSION_BUS_ADDRESS` helps the `dbus-daemon` to use the
existing
# running session rather than fork the new one!!!
#
# Otherwise, you will see a lot of forked `dbus-daemon` processes
under the
# `CGroup: /system.slice/cronie.service` when checking the service
like below:
#
# `systemctl status cronie.service`
export DISPLAY=:0
export DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"

#Check parameters
if [[ -z "$1" || "$1" -lt 0 || "$1" -gt 100 ]]; then
    echo "Usage: checkbat <warning threshold in percentage>"
    exit 1
fi

#Check if battery is present
if [[ -f /sys/class/power_supply/BAT0/capacity ]]; then
    #Poll status and current capacity
    LEVEL=$(cat /sys/class/power_supply/BAT0/capacity)
    STATUS=$(cat /sys/class/power_supply/BAT0/status)
    echo "Battery level at ${LEVEL}% $STATUS"

    #Check if the capacity is below threshold and discharging
    if [[ "$LEVEL" -lt "$1" && "$STATUS" == "Discharging" ]]; then
        dunstify --appname="Battery checker" \
                 --urgency=critical \
                 --icon=battery-level-0-charging-symbolic.symbolic \
```

```
                "Battery at ${LEVEL}%, Please connect charger
immediate!"
    fi
else
    echo "No battery!"
fi
```

- Install `cronie`

```
sudo pacman --sync --refresh cronie
```

- Add repeatable task to `cronine`

  Run `EDITOR=vim crontab -e` with the following settings:

```
*/1 * * * * ~/scripts/check-battery.sh 8
```

  It means run the `~/scripts/check-battery.sh 8` command for every minute with the `8%` threshold notification settings.

- Enable `cronie` service

```
sudo systemctl enable --now cronie.service
```

- Service log

  If you run `systemctl status cronie.service` to check the service status, it will print out the log like below:

```
● cronie.service - Periodic Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/cronie.service; enabled;
vendor preset: disabled)
     Active: active (running) since Sat 2021-02-06 20:46:55 NZDT;
44min ago
   Main PID: 722 (crond)
      Tasks: 1 (limit: 19013)
     Memory: 1.1M
     CGroup: /system.slice/cronie.service
             └─722 /usr/bin/crond -n

Feb 06 21:29:01 wison-arch CROND[5367]: (wison) CMDOUT (Unable to
send notification: Error spawning command line “dbus-launch --
autolaunch=61a4f82>
Feb 06 21:29:01 wison-arch CROND[5367]: pam_unix(crond:session):
session closed for user wison
Feb 06 21:30:01 wison-arch crond[5531]: pam_unix(crond:session):
session opened for user wison(uid=1000) by (uid=0)
Feb 06 21:30:01 wison-arch CROND[5532]: (wison) CMD (~/scripts/check-
battery.sh 60)
Feb 06 21:30:01 wison-arch CROND[5531]: (wison) CMDOUT (Battery level
at 30% Discharging)
Feb 06 21:30:01 wison-arch CROND[5531]: pam_unix(crond:session):
session closed for user wison
Feb 06 21:31:01 wison-arch crond[5610]: pam_unix(crond:session):
session opened for user wison(uid=1000) by (uid=0)
Feb 06 21:31:01 wison-arch CROND[5611]: (wison) CMD (~/scripts/check-
battery.sh 60)
Feb 06 21:31:01 wison-arch CROND[5610]: (wison) CMDOUT (Battery level
at 30% Discharging)
Feb 06 21:31:01 wison-arch CROND[5610]: pam_unix(crond:session):
session closed for user wison
```

How to open multiple window in the particular layout

You can run `i3-msg` command in terminal, it's the same way you call command in the `i3` config file.

Let's use terminal script to open multiple windows in the particular layout:)

- Write some scripts to run `i3-msg` to open windows and change layout
  - `vim ~/scripts/2w.sh` with the following contents:

```
#!/usr/bin/fish

# Open 2 windows in the particular layout below:
#  --------   --------
# |        | |        |
# |        | |        |
# |        | |        |
# |        | |        |
# |        | |        |
# |        | |        |
#  --------   --------

i3-msg --quiet --type command "split horizontal; exec alacritty;
exec alacritty" &

# The `sleep` call here is important, as the `i3-msg` needs a bit
of time to execute
# before the layout gets changed!!!
sleep 0.2

i3-msg --quiet --type command "focus left" &
```

- `vim ~/scripts/3w.sh` with the following contents:

```
#!/usr/bin/fish

# Open 3 windows in the particular layout below:
#  --------   --------
# |        | |        |
# |        | |        |
# |        |  --------
# |        |  --------
# |        | |        |
# |        | |        |
#  --------   --------

i3-msg --quiet --type command "split horizontal; exec alacritty;
exec alacritty" &

# The `sleep` call here is important, as the `i3-msg` needs a bit
of time to execute
# before the layout gets changed!!!
sleep 0.2

i3-msg --quiet --type command "split vertical; exec alacritty" &

sleep 0.1

i3-msg --quiet --type command "focus left" &
``
```

- Map the key binding to run the scripts created above

  `vim ~/.config/i3/config` with the following settings:

```
# Open window Mode: Open specified layout of windows based on the
number
mode "Open multiple windows in the specific layout" {
        # Pressing left will shrink the window’s width.
        # Pressing right will grow the window’s width.
        # Pressing up will shrink the window’s height.
        # Pressing down will grow the window’s height.
        # bindsym $left       resize shrink width $resize_unit px or
$resize_unit ppt
        # bindsym $down       resize grow height $resize_unit px or
$resize_unit ppt
        # bindsym $up         resize shrink height $resize_unit px or
$resize_unit ppt
        # bindsym $right      resize grow width $resize_unit px or
$resize_unit ppt
        bindsym 2  exec ~/scripts/2w.sh; mode "default"
        bindsym 3  exec ~/scripts/3w.sh; mode "default"

        # back to normal: Enter or Escape
        bindsym Return mode "default"
        bindsym Escape mode "default"
}
bindsym Shift+$mod+o mode "Open multiple windows in the specific
layout"
```

Run `i3-msg reload` to reload.

- Test it

  Go to any `i3` workspace and press `Shift+$mod+o`, then you can see the tips below to show on left-top area which is saying that you're in the particular mode at this moment:

  ```
  Open multiple windows in the specific layout
  ```

  And press `3`, then 3 windows should open like this:

Special keybinding

You need to install the utilities below:

- `xmodmap`: A utility for modifying keymaps and pointer button mappings in Xorg.
- `xev`: A utility which we need it to see the keycode for each key.

After that, run the commands below to generate a copy with the current `keyboard key code -> key symbol` mapping.

```
xmodmap -pke > ~/.Xmodmap
```

Inside that file, it contains the keybinding with the syntax like below:

```
keycode   9 = Escape NoSymbol Escape
```

Then you can search `keycode` or `key symbol` in that file and change the mapping. Just in case, if you will switch to different keyboard very often, then you should use

`keycode` to set your own keybinding!!!

How to know the `keycode`? Just run `xev` and press any key, it prints out the hardware `keycode`.

Here is the some `keycode` you might interest in:

```
133 - Left Super/Win key, Left Command key (Apple Keyboard)
64  - Left Alt
37  - Left Control
66  - Caps_Lock
9   - Escape
```

Let's do a real example for mapping `Caps_Lock` to `Escape`

`vim ~/.Xmodmap` then find the `keycode 66` and replace the settings like below:

```
keycode  66 = Escape NoSymbol Escape
```

Save and exit.

How to load the `~/.Xmodmap` when `i3` reload?

`vim ~/.config/i3/config` and add the settings below:

```
# It will reload `~/.Xmodmap` every time when `i3` reload
exec_always sleep 1; xmodmap ~/.Xmodmap
```

And pay attention on that: Even the `Caps_Lock` key already mapped to `Escape`, but the caplock functionality still works!!!

So always tap `Caps_Lock` twice to make sure it doesn't switch uppercase mode!!!:)

Advanced keybinding

In `Arch Linux`, it uses the key combination below by default:

- `Control + c`: Copy
- `Control + v`: Paste
- `Control + f`: Find or search
- `Control + t`: Open new tab
- `Control + w`: Close current tab

But if you're a Mac user, then you will very difficult to deal with those settings. Is that possible to change that?

YES, you can use `xdotool` combine with `i3` keybinding to do that:

`vim ~/.config/i3/config` and add the following settings:

```
#
# ================================================================================

# Special key mapping
#
# ================================================================================


# `cmd+c` -> `ctrl+c` (Copy)
bindsym --release $mod+c exec --no-startup-id xdotool key --
clearmodifiers ctrl+c

# `cmd+v` -> `ctrl+v` (Paste)
bindsym --release $mod+v exec --no-startup-id xdotool key --
clearmodifiers ctrl+v

# `cmd+f` -> `ctrl+f` (Find)
bindsym --release $mod+f exec --no-startup-id xdotool key --
clearmodifiers ctrl+f

# `cmd+t` -> `ctrl+t` (Open new tab)
bindsym --release $mod+t exec --no-startup-id xdotool key --
clearmodifiers ctrl+t

# `cmd+w` -> `ctrl+w` (Close current tab)
bindsym --release $mod+w exec --no-startup-id xdotool key --
clearmodifiers ctrl+w
```

Switching keyboard in real-time

If you have multiple keyboards, how do you switch them in `Arch Linux` in real-time?

For example, you have different keybaords below:

- Apple keybaord (embedded in your MBP) or separated Apple wireless keyboard
- Varmilo mechanical keyboard

And you want to switch between in real-time, how to do that?

Let's make some requirements for the switching task below:

- No matter switch to which keyboard, the `i3` configuration settings below no need to be changed:

```
#
# ============================================================================

# Set `$mod` to `Mod4`(WinKey or CmdKey)
#
# ============================================================================

set $mod Mod4
set $alt Mod1
```

- **`Apple keybaord`**
  - Use `Super_L`(Left Command) key as the `i3` $Mod key
  - Use `Caps Lock` key as `Escape` key
  - Use `Alt_L`(Left Alt) key as Mod1

- **`Varmilo mechanical keybaord`**

    - Exchange the `Super_L` key and `Alt_L` key, as they're in a different position with `Apple Keyboard` and you don't like to change your pressing habit
    - Use `Super_L` (Left Command) key as the `i3` $Mod key
    - Use `Caps Lock` key as `Escape` key
    - Use `Alt_L` (Left Alt) key as Mod1

Let's do it.

- **`Apple keybaord`**
    - Create `~/scripts/Xmodmap-for-apple-keyboard` from current keybinding:

      ```
      xmodmap -pke > ~/scripts/Xmodmap-for-apple-keyboard
      ```

    - Open `~/scripts/Xmodmap-for-apple-keyboard` and add some settings below:
        - Add to the beginning of the file:

          ```
          clear lock
          clear mod1
          clear mod4
          ```

        - Add to the bottom of the file:

          ```
          add mod1 = Alt_L
          add mod4 = Super_L
          ```

        - Edit the setting like below:

          ```
          keycode  66 = Escape NoSymbol Escape
          ```

    - Create `~/scripts/change-keybinding-for-apple-keyboard.sh` with the following settings:

```
#!/bin/bash
cp -rvf ~/scripts/Xmodmap-for-apple-keyboard ~/.Xmodmap
i3-msg restart
```

- Make it executable

```
chmod +x ~/scripts/change-keybinding-for-apple-keyboard.sh
```

- **`Apple keybaord` for `colemak` keyboard layout**
  - Set to `colemak` layout before you continue:

    ```
    setxkbmap us colemak
    ```

    Right now, your keyboard key mapping should be load as `colemak`.
  - Create `~/scripts/Xmodmap-colemak-for-apple-keyboard` from current keybinding:

    ```
    xmodmap -pke > ~/scripts/Xmodmap-colemak-for-apple-keyboard
    ```
  - Open `~/scripts/Xmodmap-for-apple-keyboard` and add some settings below:
    - Add to the beginning of the file:

      ```
      clear lock
      clear mod1
      clear mod4
      ```
    - Add to the bottom of the file:

      ```
      add mod1 = Alt_L
      add mod4 = Super_L
      ```
    - Edit the setting like below:

```
keycode  66 = Escape NoSymbol Escape
```

- Create `~/scripts/change-keybinding-for-apple-keyboard-colemak.sh` with the following settings:

```
#!/bin/bash
cp -rvf ~/scripts/Xmodmap-colemak-for-apple-keyboard ~/.Xmodmap
i3-msg restart
```

- Make it executable

```
chmod +x ~/scripts/change-keybinding-for-apple-keyboard-
colemak.sh
```

- **`Varmilo mechanical keybaord`**
  - Create `~/scripts/Xmodmap-for-varmilo-keyboard` from current keybinding:

```
xmodmap -pke > ~/scripts/Xmodmap-for-varmilo-keyboard
```

  - Open `~/scripts/Xmodmap-for-varmilo-keyboard` and add some settings below:
    - Add to the beginning of the file:

```
clear lock
clear mod1
clear mod4
```

    - Add to the bottom of the file:

```
add mod1 = Alt_L
add mod4 = Super_L
```

    - Edit the setting like below:

```
keycode  66 = Escape NoSymbol Escape
keycode  64 = Super_L NoSymbol Super_L
keycode 133 = Alt_L Meta_L Alt_L Meta_L
```

- Create `~/scripts/change-keybinding-for-varmilo-keyboard.sh` with the following settings:

```
#!/bin/bash
cp -rvf ~/scripts/Xmodmap-for-varmilo-keyboard ~/.Xmodmap
i3-msg restart
```

- Make it executable

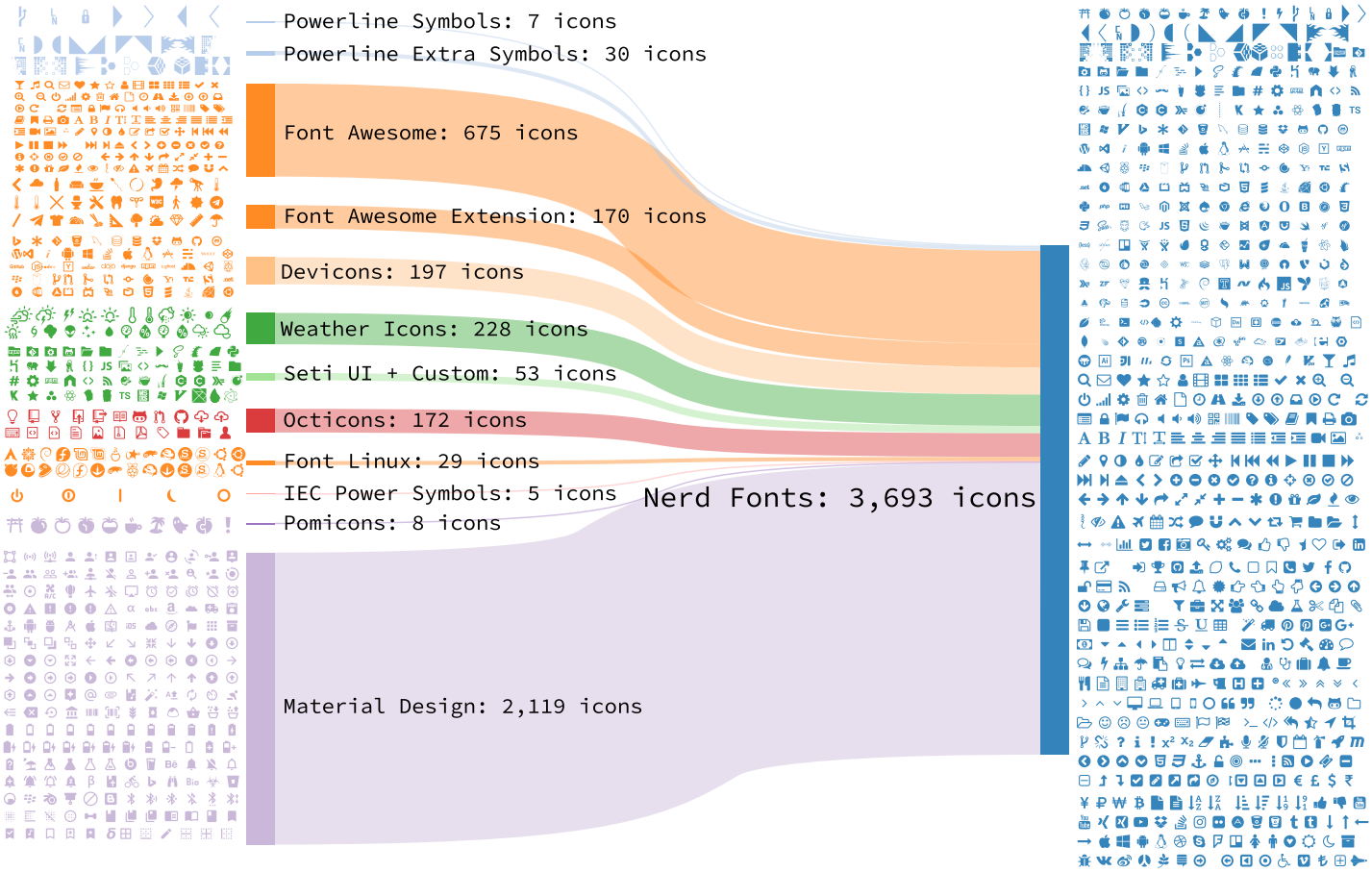
```
chmod +x ~/scripts/change-keybinding-for-varmilo-keyboard.sh
```

After that, you can switch to any keyboard at anytime you want in real-time.

```
# For Apple keybaord
~/scripts/change-keybinding-for-apple-keyboard.sh

# For Apple keybaord colemak
~/scripts/change-keybinding-for-apple-keyboard-colemak.sh

# For Varmilo keyboard
~/scripts/change-keybinding-for-varmilo-keyboard.sh
```

Install `NerdFont`

`Nerd Fonts` is a project that patches developer targeted fonts with a high number of glyphs (icons). Basically, it adds all popular icons into the popular fonts which call `patched font` which include the original font and all supported icons. Then you can use any of those `patched font` to show any supported icons below:



Installation

It has 7 optional ways to install **`Nerd Fonts`**. But for `Arch`, you can install via `AUR`. Let's take the `Source Code Pro` at an example:

```
yay -S nerd-fonts-source-code-pro
```

After installing, you can run the command below to show the installed location:

```
yay --query --list nerd-fonts-source-code-pro

# /usr/share/fonts/TTF/Sauce Code Pro Black Italic Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Black Italic Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Black Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Black Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Bold Italic Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Bold Italic Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Bold Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Bold Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Italic Nerd Font
Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Italic Nerd Font
Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Italic Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Italic Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Light Italic Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Light Italic Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Light Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Light Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Medium Italic Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Medium Italic Nerd Font
Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Medium Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Medium Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Nerd Font Complete Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Nerd Font Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Italic Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Italic Nerd Font
Complete.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Nerd Font Complete
Mono.ttf
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Nerd Font Complete.ttf
```

Setup correct font name

As you can see that, the font name is **NOT** the `Source Code Pro`!!! So you have to run the command below to know the **Real font name** which you can fill into any application's font configuration file:

```
fc-list | grep Pro

# /usr/share/fonts/TTF/Sauce Code Pro Light Italic Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Light Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Black Italic Nerd Font
Complete.ttf: SauceCodePro Nerd Font:style=Black Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Semibold,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Bold Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Bold
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Italic Nerd Font
Complete.ttf: SauceCodePro Nerd Font:style=Semibold Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Italic Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Italic
# /usr/share/fonts/TTF/Sauce Code Pro Light Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Light,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Regular
# /usr/share/fonts/TTF/Sauce Code Pro Italic Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Italic
# /usr/share/fonts/TTF/Sauce Code Pro Bold Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Bold
# /usr/share/fonts/TTF/Sauce Code Pro Medium Italic Nerd Font
Complete.ttf: SauceCodePro Nerd Font:style=Medium Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Medium Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Medium,Regular
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Italic Nerd Font
Complete.ttf: SauceCodePro Nerd Font:style=ExtraLight Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Bold Italic Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Bold Italic
# /usr/share/fonts/TTF/Sauce Code Pro Medium Italic Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Medium Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Light Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Light,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Bold Italic Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Bold Italic
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=ExtraLight,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Black Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Black,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Black Italic Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Black Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=ExtraLight,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Medium Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Medium,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Black Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Black,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Italic Nerd Font Complete
Mono.ttf: SauceCodePro Nerd Font Mono:style=Semibold Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro ExtraLight Italic Nerd Font
```

```
Complete Mono.ttf: SauceCodePro Nerd Font Mono:style=ExtraLight
Italic,Italic
# /usr/share/fonts/TTF/Sauce Code Pro Nerd Font Complete Mono.ttf:
SauceCodePro Nerd Font Mono:style=Regular
# /usr/share/fonts/TTF/Sauce Code Pro Semibold Nerd Font Complete.ttf:
SauceCodePro Nerd Font:style=Semibold,Regular
# /usr/share/fonts/TTF/Sauce Code Pro Light Italic Nerd Font
Complete.ttf: SauceCodePro Nerd Font:style=Light Italic,Italic
```

Right now, you should notice that the font name is either `SauceCodePro Nerd Font Mono` or `SauceCodePro Nerd Font`. So fill them into your application font configuration file.

For example, put it into the ** `Alacritty` configuration file:

```
  # Bold italic font face
  bold_italic:
    # Font family
    #
    # If the bold italic family is not specified, it will fall back to
the
    # value specified for the normal font.
    family: "SauceCodePro Nerd Font"

    # The `style` can be specified to pick a specific face.
    style: Bold Italic
```

Copy icons from `NerdFont` cheatsheet

After setting the correct font name, then you can copy and paste the icons from the **NerdFont cheatsheet** :

Just type any searching keyword there, place the mouse on top of the one you like, it will show up a popup menu on the right-top corner. So, click `Icon` to copy to clipbobard, then you can paste to anywhere to have that icon.

`Vim` support

Add the settings below to your `vimrc` or `init.vim` (for `neovim`):

```
#
Plug 'ryanoasis/vim-devicons'

# Set the `guifont` to your patched font name with the size
set guifont=SauceCodePro\ Nerd\ Font\ 11

# Enable this line if you use `vim-airline`
let g:airline_powerline_fonts = 1
```

Then run `:PlugInstall` and restart `vim` to take effect.

Install Ranger

- Install

```
sudo pacman -S ranger

# During the installation process, it will list another optional
packages you might needed:

# Optional dependencies for ranger
#   atool: for previews of archives
#   elinks: for previews of html pages
#   ffmpegthumbnailer: for video previews
#   highlight: for syntax highlighting of code
#   libcaca: for ASCII-art image previews
#   lynx: for previews of html pages
#   mediainfo: for viewing information about media files
#   odt2txt: for OpenDocument texts
#   perl-image-exiftool: for viewing information about media files
#   poppler: for pdf previews
#   python-chardet: in case of encoding detection problems
#   sudo: to use the "run as root"-feature
#   transmission-cli: for viewing bittorrent information
#   ueberzug: for previews of images
#   w3m: for previews of images and html pages

# You just pick what you want. For example:

sudo pacman -S ueberzug highlight mediainfo sudo poppler
```

- Generate your own configuration

You need to run the command below to generate a copy of the `ranger` rc configuration, it will saved to your `~/.config/ranger/rc.conf`.

```
ranger --copy-config=rc
```

- The normal daily keybinding you need to know (just in case if you're new to `ranger`)

Keybinding	Description
`Basic`	*Basic file and folder operations*
yy	Copy current file or folder, then you can to to any folder and paste it
dd	Cut current file or folder, then you can to to any folder and paste it
dD	Delete the file same with `rm -rf`, need to press `enter` to confirm
pp	paste the last copied file or folder to current folder
yp	(Yank full path), copy the full path with filename to clipboard
yd	(Yank dir), copy the full path WITHOUT filename to clipboard
yn	(Yank name), copy only the filename to clipboard
cw	(Change word), rename current file or folder
I	(Insert), rename, cursor position will be stopped at the beginning of the filename. 
a	(append), rename, cursor position will be stopped at the end of the filename. 

Keybinding	Description
A	(append at the end), rename, cursor position will be stopped at the end of the extension name. 
cw	(Change word), rename current file or folder
oc	Order/Sort by change time (latest to older)
oC	Order/Sort by change time (older to latest)
ot	Order/Sort by type
oT	Order/Sort by type (reversed)
on	Order/Sort by name
oN	Order/Sort by name (reversed)
Control+h	Toggle show hidden file
:mkdir xx	Create new folder
:touch xx	Create empty file
`Select`	*Select/Mark mode*
va	Select all or unselect all
V	Toggle select mode, move your cursor to select file in select mode
Space	Toggle select mode for single file
`Change folder`	*Change folder quickly*
gh	Go home (to `$HOME`)
ge	Go to `/etc`
gu	Go to `/usr`
gv	Go to `/var`
`Shell`	*Shell related operations*
du	Run `du` (disk usage) command on current foler and sort the result by size
#	Type your command to run in shell with current path, wait for press `enter` to return.

Keybinding	Description
Shift+s	Open terminal with current folder/path. When you done, type `exit` to back to `ranger`.

- Add customization configration if you needed

  Here are some very useful settings maybe can help you, `vim ~/.config/ranger/rc.conf` and pick what you want to try. You should search `set xxxx` and replace to the below settings.

```
# Only show 2 columns with 1:2 width
set column_ratios 2,4

# Change color themes
set colorscheme jungle

# Show both borders, looks comfortable for some people
# set draw_borders separators
# set draw_borders outline
# set draw_borders none
set draw_borders both

# Sync the folder name to window title
set update_title true

# Display `~` related to your home folder, save some display space
set tilde_in_titlebar true

# Show the relative line number. If you use this feature in `vim`,
then
# you will like it, it can help u fast jump to the specified line
set line_numbers relative

# Just add any `go to XXX` keybinding for yourself, For example:
map gd cd ~/Downloads
# map gr cd ~/Rust
# map gs cd ~/Screenshots
# map gb cd ~/my-shell/backup
# map gp cd ~/xxx/yyy/zzz/...very deep path/YOUR_PROJECT_FOLDER

# Replace the default `du` command, as it doesn't sort by size by
default
# map du shell -p du --max-depth=1 -h --apparent-size
map du shell -p du --max-depth=1 -h --apparent-size | sort -rh
```

More powerful features:

- Add `fuzzy finder` to `ranger`
  - First you need to install `fuzzy finder` by running `sudo pacman -S fzf`
  - Add `~/.config/ranger/commands.py` with the following settings

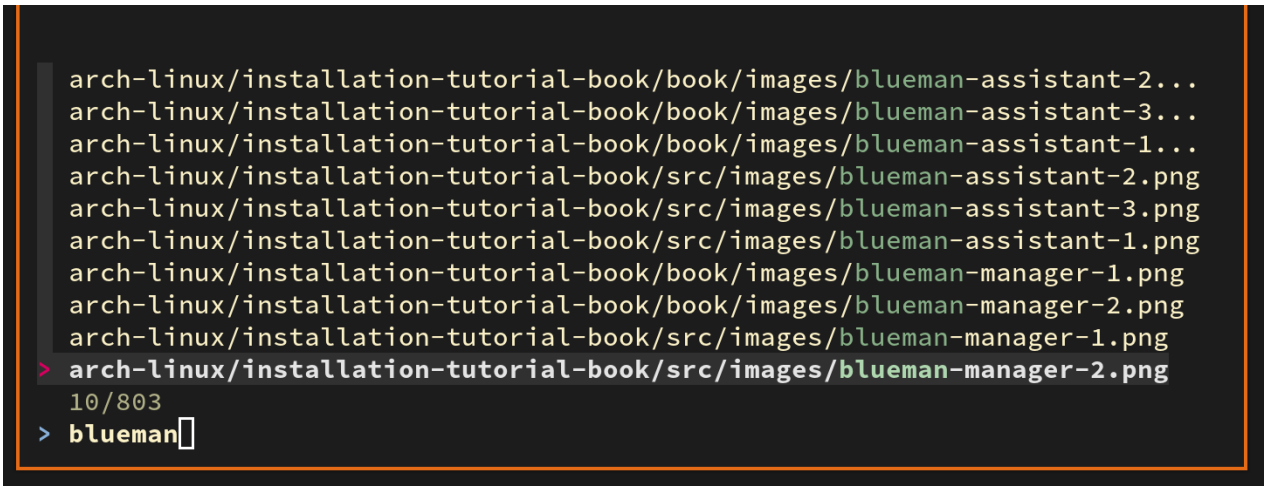
```
from ranger.api.commands import Command

class fzf_select(Command):
    """
    :fzf_select

    Find a file using fzf.

    With a prefix argument select only directories.

    See: https://github.com/junegunn/fzf
    """
    def execute(self):
        import subprocess
        import os.path
        if self.quantifier:
            # match only directories
            command = "find -L . \( -path '*/\.*' -o -fstype
'dev' -o -fstype 'proc' \) -prune \
            -o -type d -print 2> /dev/null | sed 1d | cut -
b3- | fzf +m"

        else:
            # match files and directories
            command = "find -L . \( -path '*/\.*' -o -fstype
'dev' -o -fstype 'proc' \) -prune \
            -o -print 2> /dev/null | sed 1d | cut -b3- | fzf
+m"

        fzf = self.fm.execute_command(command,

universal_newlines=True,

stdout=subprocess.PIPE)
        stdout, stderr = fzf.communicate()
        if fzf.returncode == 0:
            fzf_file = os.path.abspath(stdout.rstrip('\n'))
```

```
            if os.path.isdir(fzf_file):
                self.fm.cd(fzf_file)
            else:
                self.fm.select_file(fzf_file)
```

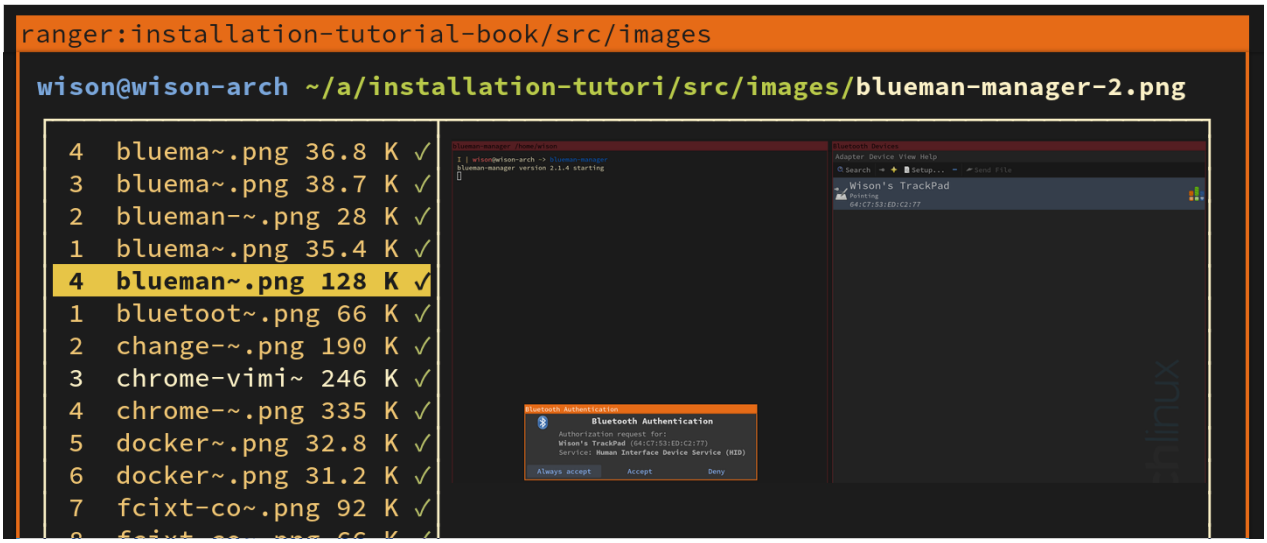
- Add keybinding to `~/.config/ranger/rc.conf`

```
map <c-f> fzf_select
```

After that, open `ranger` and press `Control + f` and type any part of the file name, you got crazy fast search result:



press `Enter` to go to the selected file:

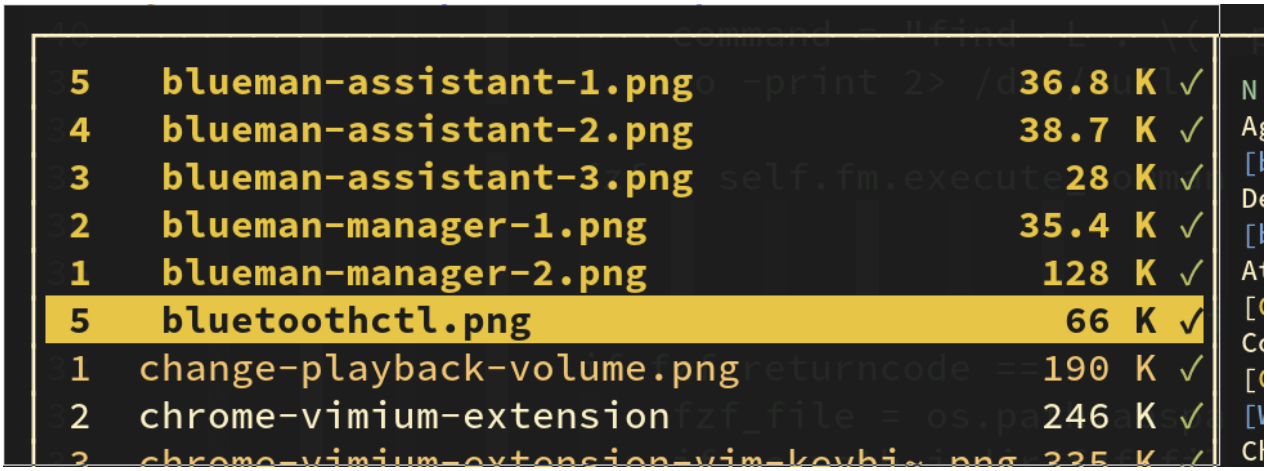


- Fast rename multiple selected files

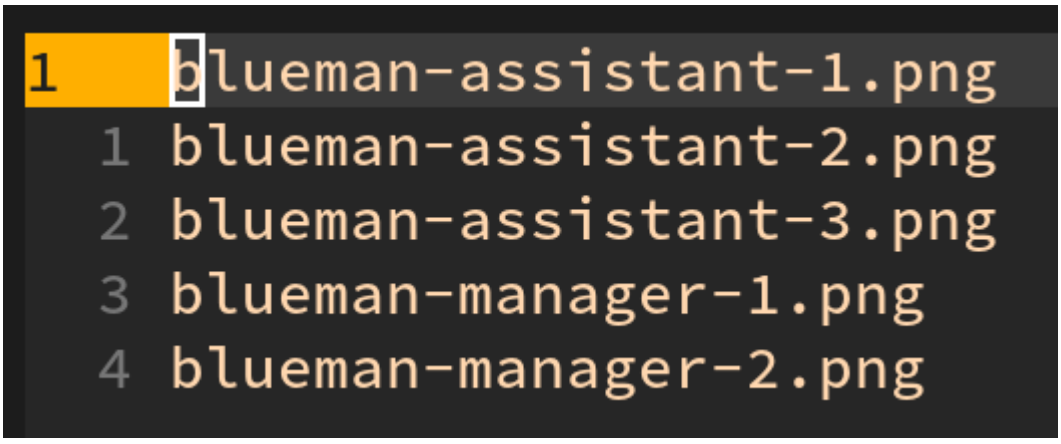
`vim ~/.config/ranger/rc.conf`, and then replace the below keybinding to overwrite the default one:

```
map cw eval fm.execute_console("bulkrename") if
fm.thisdir.marked_items else fm.open_console("rename ")
```

After that, select all the files you want to rename to together like below:



Then press `cw`, it will show your selected file name in your default editor. Change it and save it, done.



Everything related to `neovim`

- Installation

```
sudo pacman --sync --refresh neovim python-pynvim

mkdir ~/.config/nvim
touch ~/.config/nvim/init.vim
```

- Auto install `vim-plug` when open `nvim`

  `nvim ~/.config/nvim/init.vim` and add the following settings:

```
" ===
" === Auto load for first time uses
" ===
if empty(glob('~/.config/nvim/autoload/plug.vim'))
        silent !curl -fLo ~/.config/nvim/autoload/plug.vim --create-
dirs
                                    \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
        autocmd VimEnter * PlugInstall --sync | source $MYVIMRC
endif
```

- `Ranger` plugin

  `nvim ~/.config/nvim/init.vim` with the following settings:

```
Plug 'kevinhwang91/rnvimr'

" ----------------------------------------------------------------------
---------
" 'rrnvim'
" ----------------------------------------------------------------------
---------
" Make Ranger to be hidden after picking a file
let g:rnvimr_enable_picker = 1

" Hide the files included in gitignore
let g:rnvimr_hide_gitignore = 1

" Fullscreen for initial layout
let g:rnvimr_layout = {
            \ 'relative': 'editor',
            \ 'width': &columns,
            \ 'height': &lines - 2,
            \ 'col': 0,
            \ 'row': 0,
            \ 'style': 'minimal'
            \ }

" Open key mapping
nnoremap <leader>r :RnvimrToggle<CR>
```

- `Coc` troubleshooting
  - `TypeScript`: If you see the error below when you open any `TypeScript` file:

    ```
    [tsserver 2307] [E] Cannot find module XXXXXXXX
    ```

    That means you should already installed the `coc-deno` extention. It's very easy to fix it:

  - Run `:CocList extentions` command
  - Inside the extentions list, search `coc-deno`, then press `tab` on it and choose `disable`.

About GPU

- Identify your graphics card:

```
lspci -v | grep -A1 -e VGA -e 3D

# It will print something like this:
#
# VGA compatible controller: Intel Corproation Crystal ell Integrated
Grahpics Controller (rev 08) (prog-if 00 [VGA controller])
# Subsystem: Apple Inc. Device 0147
```

  So you can know that's a GPU integrated inside Intel CPU and 'Subsystem' show you the specific model for you to find the driver

- Install the video driver

```
# Driver - `xf86-video-BRAND_NAME_HERE`
# OpenGL - `lib32-mesa` if that's Intel
# OpenGL (multilib) - `mesa` if that's Intel
sudo pacman -S xf86-video-intel mesa mesa-demos
sudo pacman -Sy
```

  here is the full list for all GPU brands.

  After that, you can run the utilities below to check your OpenGL ability:

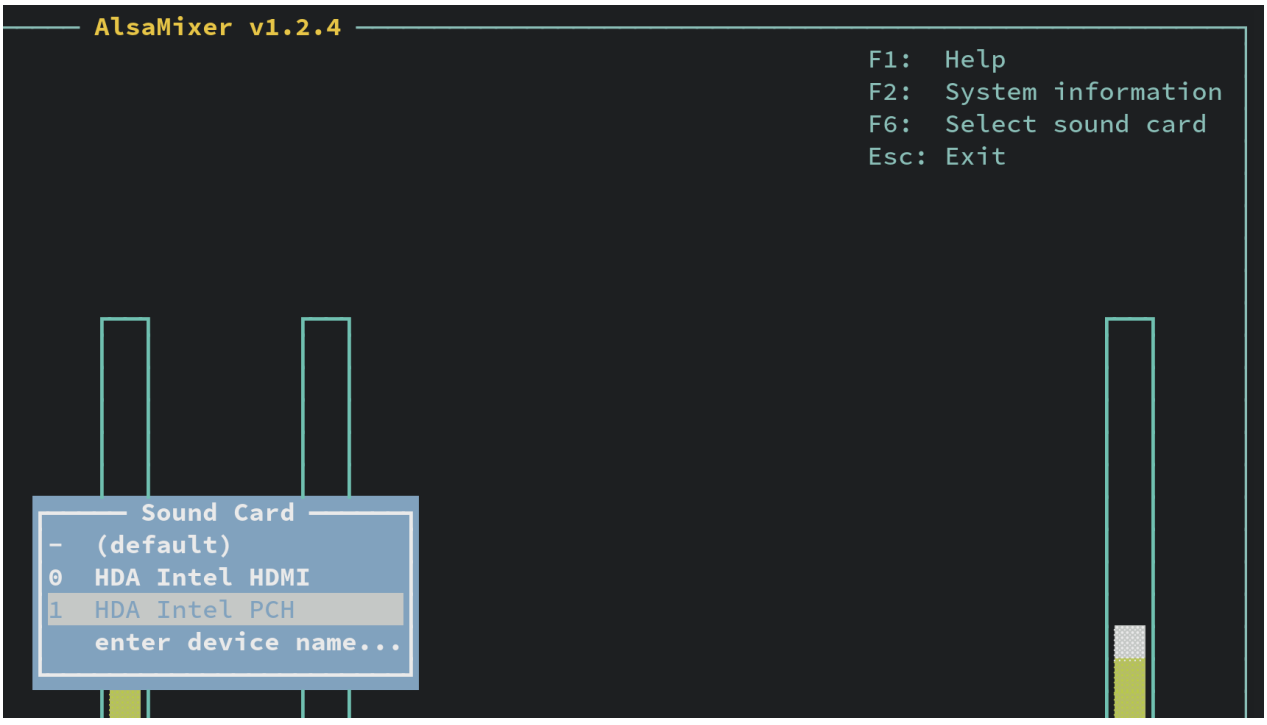
```
# Check the OpenGL version
glxinfo | grep "OpenGL version"

# Test the OpenGL render framerate
glxgears
```

Fix sound issue

- Install the packages below:

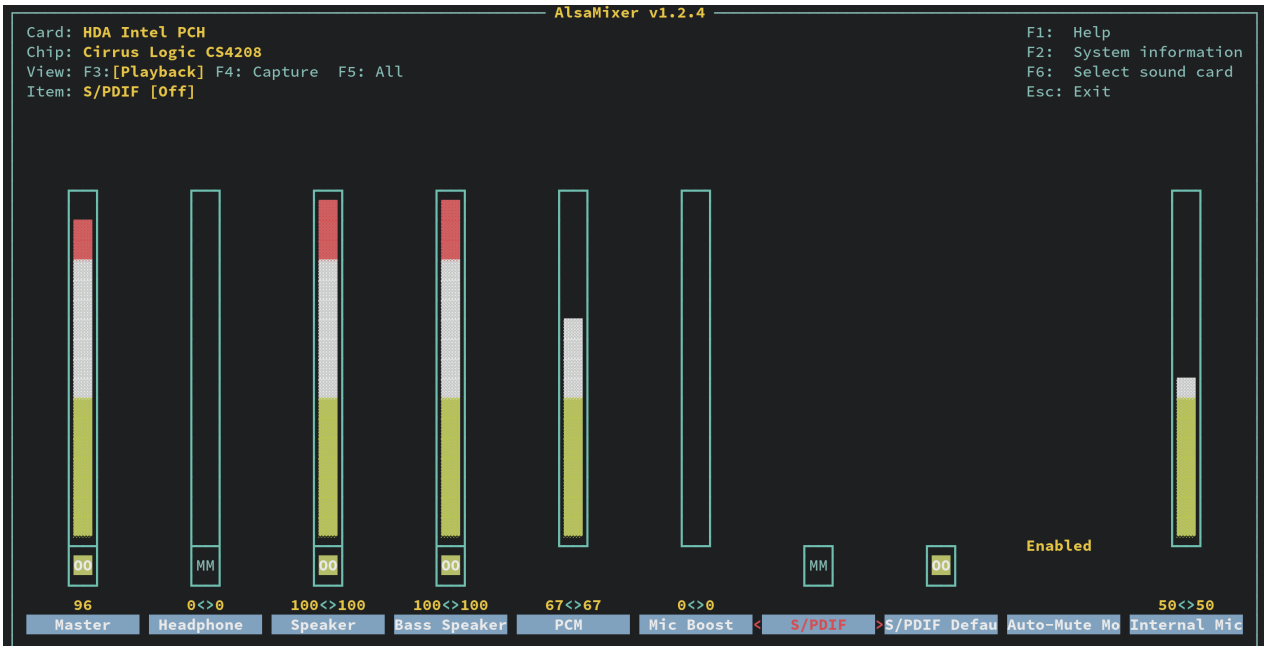
```
sudo pacman -Sy pulseaudio alsa-utils

# Reboot
```

- Run `alsamixer` and tune your sound setting
  - Press `F6` to choose your major audio playback device:



  - After selecting your correct audio playback device, press `F3` to setup the `Playback` volume. Use left/right arrow key to switch item and use

Up/Down arrow key to change the volume.



- You should be able to control with your `volume up/down` key on your keyboard

  If you installed `i3`, by default it has the volume keybindings like below:

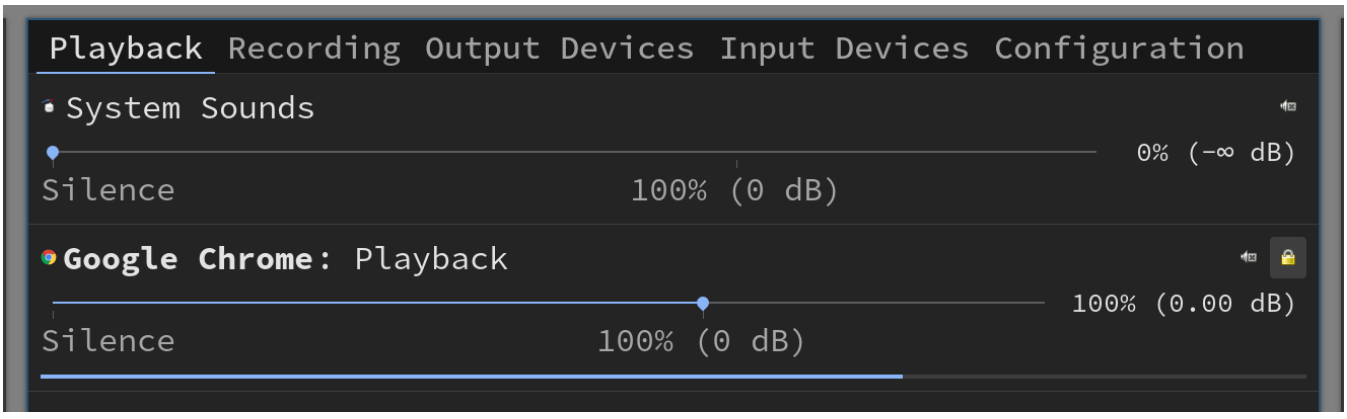
```
#
=============================================================================

# Audio and volume control
#
=============================================================================

# Use pactl to adjust volume in PulseAudio.
set $refresh_i3status killall -SIGUSR1 i3status
bindsym XF86AudioRaiseVolume exec --no-startup-id pactl set-sink-
volume @DEFAULT_SINK@ +10% && $refresh_i3status
bindsym XF86AudioLowerVolume exec --no-startup-id pactl set-sink-
volume @DEFAULT_SINK@ -10% && $refresh_i3status
bindsym XF86AudioMute exec --no-startup-id pactl set-sink-mute
@DEFAULT_SINK@ toggle && $refresh_i3status
bindsym XF86AudioMicMute exec --no-startup-id pactl set-source-mute
@DEFAULT_SOURCE@ toggle && $refresh_i3status
```

It should work out-of-the-box.

- Optionally, you can install `pavucontrol` GUI volume control app

```
sudo pacman -Sy pavucontrol
```

It looks like this:



Fix webcam issues

- Check your `webcam` whether being detected
  - If you got `v4l-utils`, then try to run:

    ```
    v4l2-ctl --list-devices

    # Cannot open device /dev/video0, exiting.
    ```

  - If you got `vlc`, then just run:

    ```
    vlc v4l2:// :input-slave=alsa:// :v4l-vdev="/dev/video0"
    ```

    Or open `vlc` and choose `media --> Open Capture Device` and test it.

  Basically, if you don't have `/dev/video0` device, then you webcam is not supported by the default kernel driver. So, you need to solve that manually.

- List your hardware and find the driver

  First you need to know what webcam hardware you have on your computer by running:

```
sudo lspci -v | grep -e Cam -A18

# 04:00.0 Multimedia controller: Broadcom Inc. and subsidiaries 720p
FaceTime HD Camera
#       Subsystem: Broadcom Inc. and subsidiaries 720p FaceTime HD
Camera
#       Flags: bus master, fast devsel, latency 0, IRQ 39
#       Memory at a0a00000 (64-bit, non-prefetchable) [size=64K]
#       Memory at 80000000 (64-bit, prefetchable) [size=256M]
#       Memory at a0900000 (64-bit, non-prefetchable) [size=1M]
#       Capabilities: [48] Power Management version 3
#       Capabilities: [58] MSI: Enable+ Count=1/1 Maskable- 64bit+
#       Capabilities: [68] Vendor Specific Information: Len=44 <?>
#       Capabilities: [ac] Express Endpoint, MSI 00
#       Capabilities: [100] Advanced Error Reporting
#       Capabilities: [13c] Device Serial Number 00-00-00-ff-ff-00-
00-00
#       Capabilities: [150] Power Budgeting <?>
#       Capabilities: [160] Virtual Channel
#       Capabilities: [1b0] Latency Tolerance Reporting
#       Capabilities: [220] Physical Resizable BAR
#       Kernel driver in use: bdc-pci
#       Kernel modules: bdc_pci
```

For my case, even the kernel driver `bdc_pci` is loaded, but the webcam still not working. That means the default `bdc_pci` driver is not correct or not support the `Apple MacBook Pro FaceTime HD Camera` provided by `Boradcom Inc`.

After that, you can go to here to find your webcam driver if it exists there.

- Install `linux-header` if you don't have it

  before install driver, you better to check whether the `linux-headers` is installed or not by runing:

```
pacman --query | sort | grep linux
```

If you don't have `linuxXXX-headers` (`XXX` is you kernel version which you can get from `uname -sr`), then you need to install the `linux-headers` first.

```
sudo pacman -S linux-headers
```

Otherwise, you might get error like below during installing your webcam driver:

```
Unable to install module bcwc-pcie/r245.82626d4 for kernel *: Missing
kernel headers.
```

- Install your webcam driver

  For my case, I will install the `bcwc-pcie-git` driver via `AUR`.

  ```
  yay -S bcwc-pcie-git
  ```

  After that, the new module `facetimehd` will be installed and it conflicts with the `bdc_pci` module. So, you need to disable it by doing this:

  `sudo vim /etc/modprobe.d/blacklist.conf` with the following settings:

  ```
  backlist bdc_pci
  install bdc_pci /bin/fals
  ```

  Then next reboot, it will load the correct module to serve your webcam.

  If you don't want to wait for the next reboot, then just do like this:

  ```
  # Unload all of them
  modprobe -r facetimehd
  modprobe -r bdc_pci

  # Then load the new module
  modprobe facetimehd
  ```

  Right now, it should work:

```
v4l2-ctl --list-devices

# Apple Facetime HD (PCI:0000:04:00.0):
#     /dev/video0
```

Support chinese

- Install chinese input method

  ```
  sudo pacman -S fcitx fcitx-im fcitx-configtool
  ```

  After finishing, it will create an auto start desktop link in here:

  `/etc/xdg/autostart/fcitx-autostart.desktop`

- Add input method configuration file

  Put the following settings into `~/.xprofile`:

  ```
  export GTK_IM_MODULE=fcitx
  export QT_IM_MODULE=fcitx
  export XMODIFIERS="@im=fcitx"
  ```

- Auto start `fcitx` in your `i3` configuration

  `vim ~/.config/i3/config` and add the following settings:

  ```
  # Load chinese input method
  exec_always fcitx
  ```

  Make sure you reboot to take effect, or you can reboot after installing chinese fonts below.

- Install chinese fonts

```
yay -S wqy-bitmapfont wqy-microhei \
    wqy-microhei-lite \
    wqy-zenhei \
    adobe-source-han-mono-cn-fonts \
    adobe-source-han-sans-cn-fonts \
    adobe-source-han-serif-cn-fonts
```

  Now, relogin to take effect.

- Install extra input methods

  If you need 仓颉、五笔等输入法

```
sudo pacman S fcitx-table-extra
```

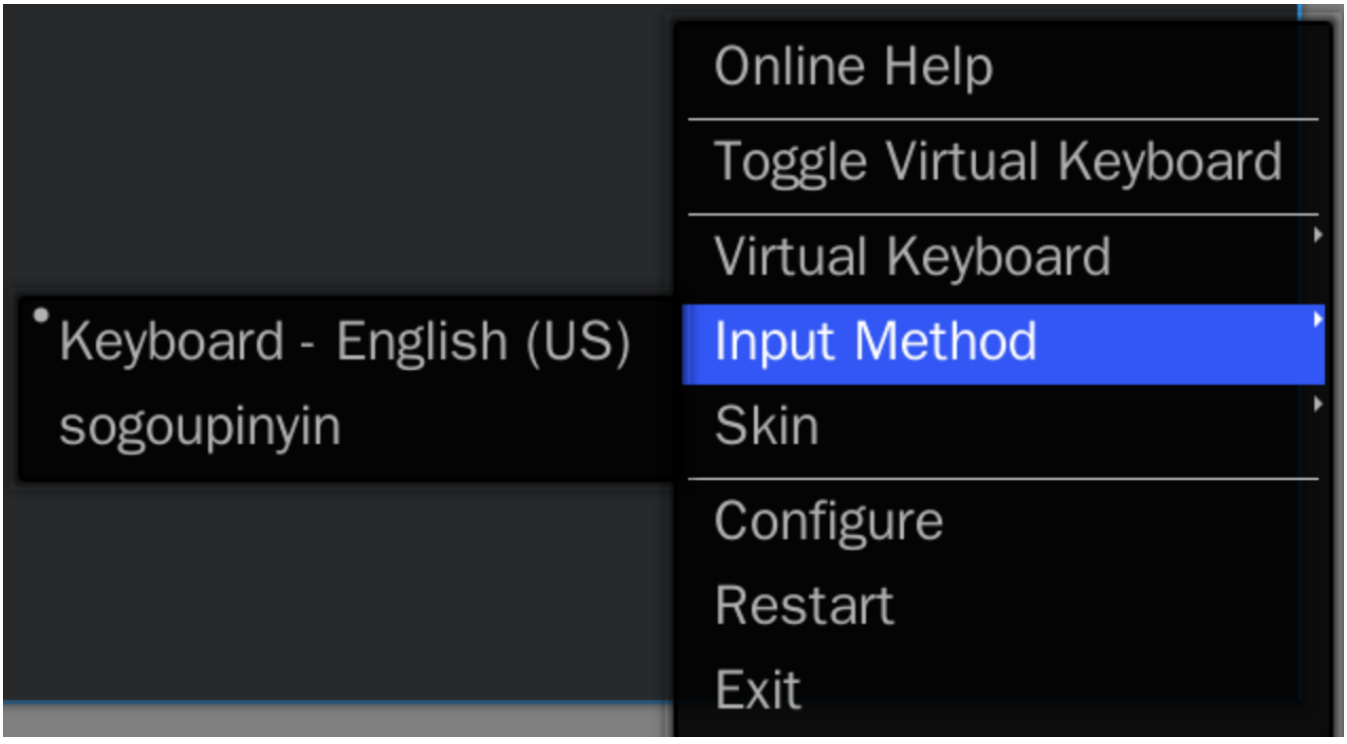
  If you need 搜狗拼音五笔混输

```
# SouGou PinYin (it inclues WuBi)
yay -S fcitx-sogoupinyin
```

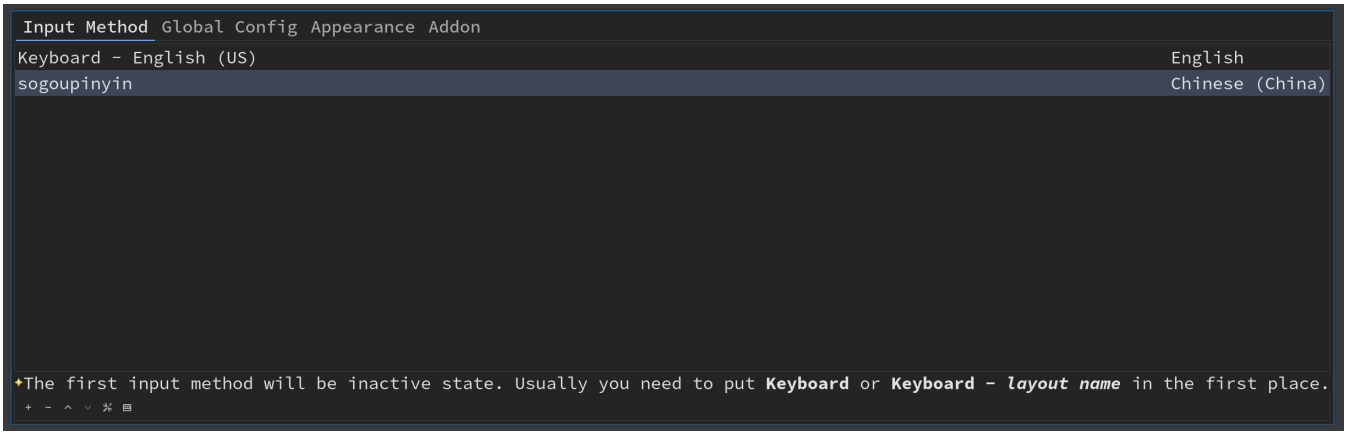
```
translate pathname 'opt/sogoupinyin/files/share/resources/skin/经典酷黑/configtool/picture/single-select-white.png' to UTF-8: Can't translate pathname 'opt
/sogoupinyin/files/share/resources/skin/经典酷黑/configtool/picture/single-select.png' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/reso
urces/skin/经典酷黑/configtool/picture/sogou.png' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/resources/skin/经典酷黑/configtool/pictur
e/sogou130x130.png' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/resources/skin/经典酷黑/configtool/picture/test.png' to UTF-8: Can't tr
anslate pathname 'opt/sogoupinyin/files/share/resources/skin/经典酷黑/configtool/sogou-logo.png' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/
share/resources/skin/经典酷黑/configtool/sogouImeService.qss' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/resources/skin/经典酷黑/经典
酷黑.zip' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/shell/dict/PCPYDict/scd/古诗词名句【官方推荐】.scel' to UTF-8: Can't translate pa
thname 'opt/sogoupinyin/files/share/shell/dict/PCPYDict/scd/成语俗语【官方推荐】.scel' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/shel
l/dict/PCPYDict/scd/政府机关团体机构大全【官方推荐】.scel' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/shell/dict/PCPYDict/scd/行政地区
.scel' to UTF-8: Can't translate pathname 'opt/sogoupinyin/files/share/shell/dict/PCPYDict/scd/计算机词汇大全【官方推荐】.scel' to UTF-8==> Leaving fakeroo
t environment.
==> Finished making: fcitx-sogoupinyin 2.4.0.2732-1 (Wed 23 Dec 2020 03:02:56 PM NZDT)
==> Cleaning up...
[sudo] password for wison:
loading packages...
resolving dependencies...
looking for conflicting packages...

Packages (1) fcitx-sogoupinyin-2.4.0.2732-1

Total Installed Size:  290.62 MiB

:: Proceed with installation? [Y/n]
(1/1) checking keys in keyring                                      [#####################################################] 100%
(1/1) checking package integrity                                    [#####################################################] 100%
(1/1) loading package files                                         [#####################################################] 100%
(1/1) checking for file conflicts                                   [#####################################################] 100%
(1/1) checking available disk space                                 [#####################################################] 100%
:: Processing package changes...
(1/1) installing fcitx-sogoupinyin                                  [#####################################################] 100%
:: Running post-transaction hooks...
(1/2) Arming ConditionNeedsUpdate...
(2/2) Updating icon theme caches...
I | wison | /home/wison ☞ dust -d1 /opt/sogoupinyin/
 4.0K    ┌── info                                                                                                    0%
 140K    ├── entries                                                                                                 0%
 286M    ├── files                                                                                                 100%
 287M    └── sogoupinyin                                                                                           100%
```

- About the configuration

  After re-login, you should be able to see the `fcixt-configtool` icon shows on the bottom-right of your screen. Click on it, it will show a menu like below:



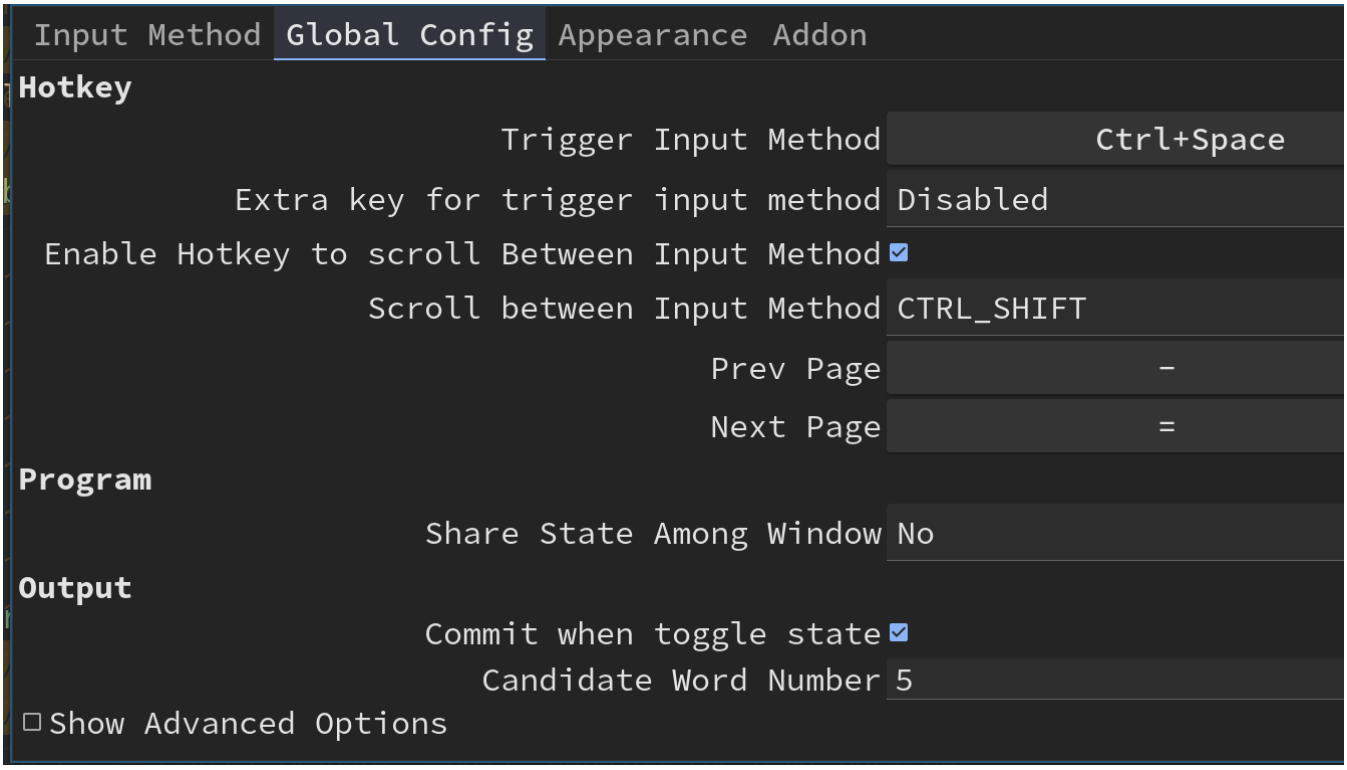
  Click on `configure`, then you will see the configuration UI below:



If you don't see your installed input method, then click the left-bottom `+` button to add it.

The default shortcuts

- `Ctrl+Space`: Trigger input method. If you only install single input method (`fcitx-sogoupinyin`), then it works like toggle chinese input method.
- `Shirt+Ctrl+f`: Switch between `Traditional Chinese` and `Simplified Chinese`.



Press `Ctrl + Space`, you should able to see your input method, then you can configure it if you wanted:

Install docker

- Installation

```
sudo pacman -S docker
```

What components are installed be default?

```
Packages (4) bridge-utils-1.7-1  containerd-1.4.3-1  runc-1.0.0rc92-1  docker-1:19.03.14-3

Total Download Size:    87.84 MiB
Total Installed Size:  375.75 MiB
```

- Handle `docker` as a service
  - Query status

    ```
    sudo systemctl status docker
    ```

  - Start

    ```
    sudo systemctl start docker
    ```

  - Stop

    ```
    sudo systemctl stop docker
    ```

  - Enable auto start

    ```
    sudo systemct enable docker
    ```

  - Disable auto start

    ```
    sudo systemct disable docker
    ```

  Of course you can add the commands above to your shell configuration as an `alias` or `abbreviation`. For example, `fish` abbreviation sample:

  ```
  abbr startdocker "sudo systemctl start docker"
  abbr stopdocker "sudo systemctl stop docker"
  ```

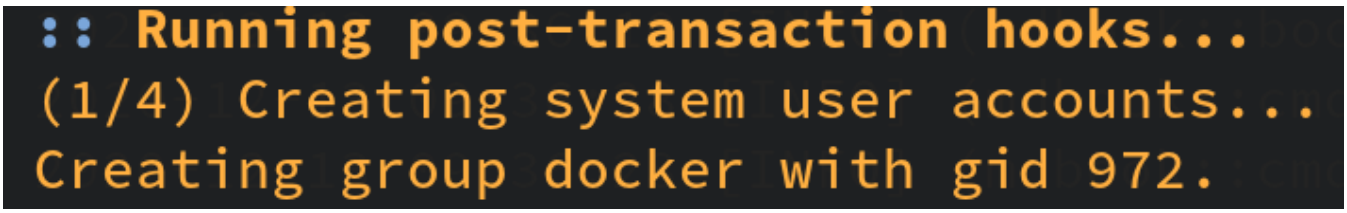
- Run `docker` client without `sudo`

The `Docker` daemon binds to a `Unix socket` instead of a TCP port. By default that `Unix socket` is owned by the user `root` and other users can only access it using `sudo`. The `Docker` daemon always runs as the `root` user.

That means you have run `sudo docker COMMAND` rather than `docker COMMAND`!

If you don't want to preface the `docker` command with `sudo`, add your linux account to `docker` group. When the `Docker` daemon starts, it creates a `Unix socket` accessible by members of the `docker` group.

By default, the `docker` group will be created during the installation:



You can confirm that by running the command below:

```
cat /etc/group | grep docker
```

If it doesn't exists, then create it by yourself:

```
# Optional step!
sudo groupadd docker
```

Now, add your linux account into the `docker` group:

```
sudo usermod -aG docker $USER
```

You need to re-login to take effect.

After that, run `docker info` (after running `docker` service) to test the permission, should be ok already.

- Install `docker-compose`

```
sudo bash -c 'curl -L
"https://github.com/docker/compose/releases/download/1.27.4/docker-
compose-$(uname -s)-$(uname -m)" \
    -o /usr/bin/docker-compose && chmod +x /usr/bin/docker-compose'
```

Screen recording

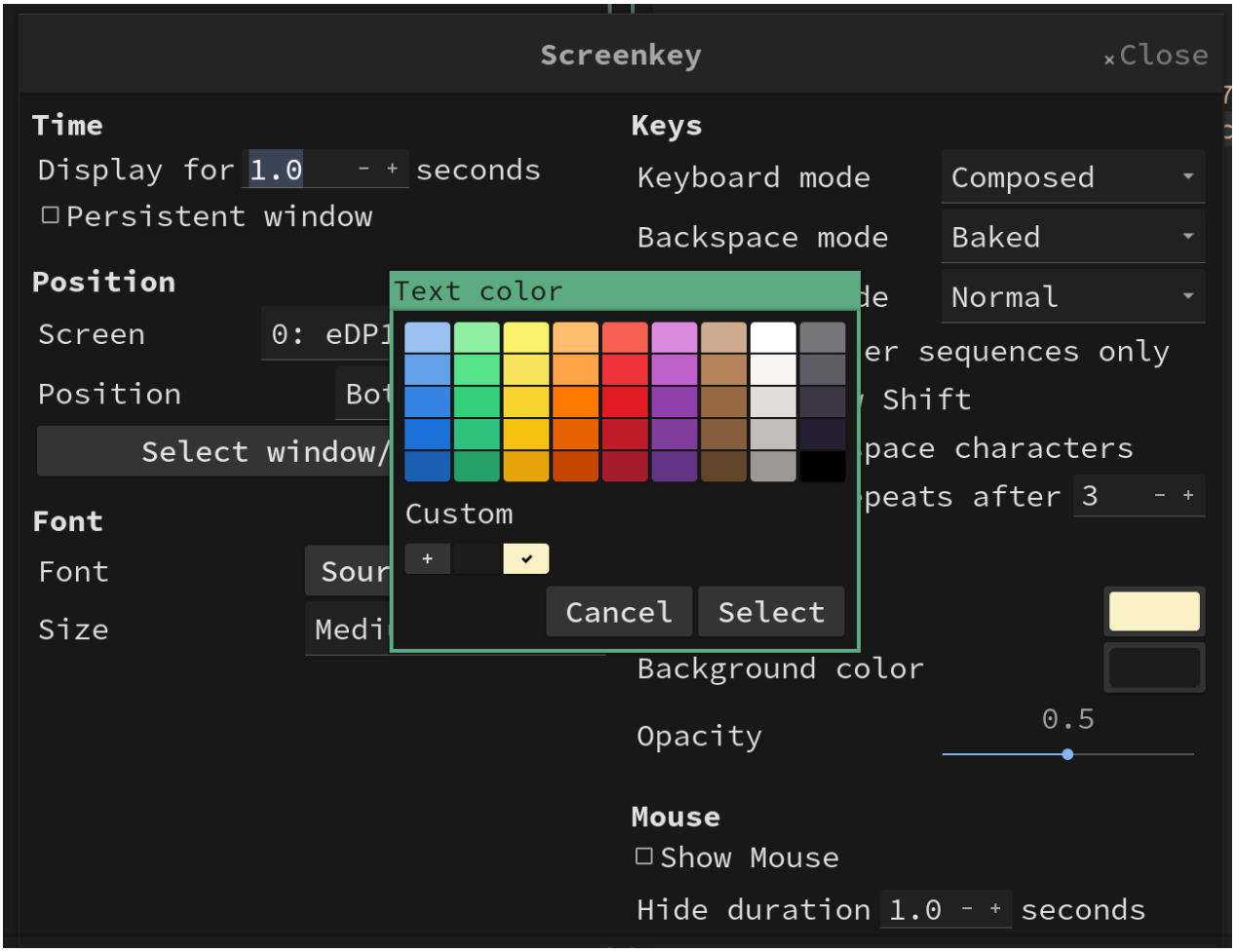
- `screenkey`
  - Install

    ```
    sudo pacman -S screenkey
    ```

  - Configuration

    When you run `screenkey`, the last config will be saved to `~/.config/screenkey.json`.

    But DO NOT edit this file manually, you should use the startup parameters or the `peference` menu in the tray icon to do that.



If you don't pass the `--no-systray`, then the tray icon will be shown.

- About the `color` settings

```
{
    "no_systray": false,
    "timeout": 1.0,
    "recent_thr": 0.1,
    "compr_cnt": 3,
    "ignore": [],
    "position": "bottom",
    "persist": false,
    "font_desc": "Source Code Pro",
    "font_size": "medium",
```


"font_color": "#fbfbf1f1c7c7",
"bg_color": "#1c1c1c1c1c1c",
"opacity": 0.5,
"key_mode": "composed",
"bak_mode": "baked",
"mods_mode": "normal",
"mods_only": false,
"multiline": false,
"vis_shift": false,
"vis_space": false,
"geometry": null,
"screen": 0,
"start_disabled": false,
"mouse": false,
"button_hide_duration": 1
}

As you can see above, the `R,G, B` value will be `double`!!! If you don't double, then `screenkey` won't run!!!

- `i3` keybinding for toggling `screenkey`

  `vim ~/scripts/toggle-screenkey.sh` with the following content:

```
#!/bin/bash

KILL_RESULT=$(killall -v screenkey 2>&1)
PROCESS_NOT_FOUND_STR="no process found"
echo "KILL_RESULT: $KILL_RESULT"

if [[ $KILL_RESULT == *"$PROCESS_NOT_FOUND_STR"* ]]; then
    echo "'screenkey' started."
    screenkey --no-systray --timeout 1 \
            --opacity 0.5 \
            --no-whitespace \
            --font "So urce Code Pro" \
            --font-size medium \
            --font-color "#FBFBF1F1C7C7" \
            --bg-color "#1C1C1C1C1C1C" &
else
    echo "Killed all 'screenkey'"
fi
```

  `vim ~/.config/i3/config` and add the following settings:

```
# Toggle `screenkey`
bindsym $mod+s exec --no-startup-id ~/scripts/toggle-
screenkey.sh
```

- Simple screen recorder

```
sudo pacman -S simplescreenrecorder
```

Running wechat in docker

- Install

```
# Download the docker run script
curl "https://raw.githubusercontent.com/huan/docker-
wechat/master/dochat.sh" --output ~/scripts/dochat.sh

# Make it executable
chmod +x ~/scripts/dochat.sh
```

- Better to make an `alias` or `abbreviation` in your shell

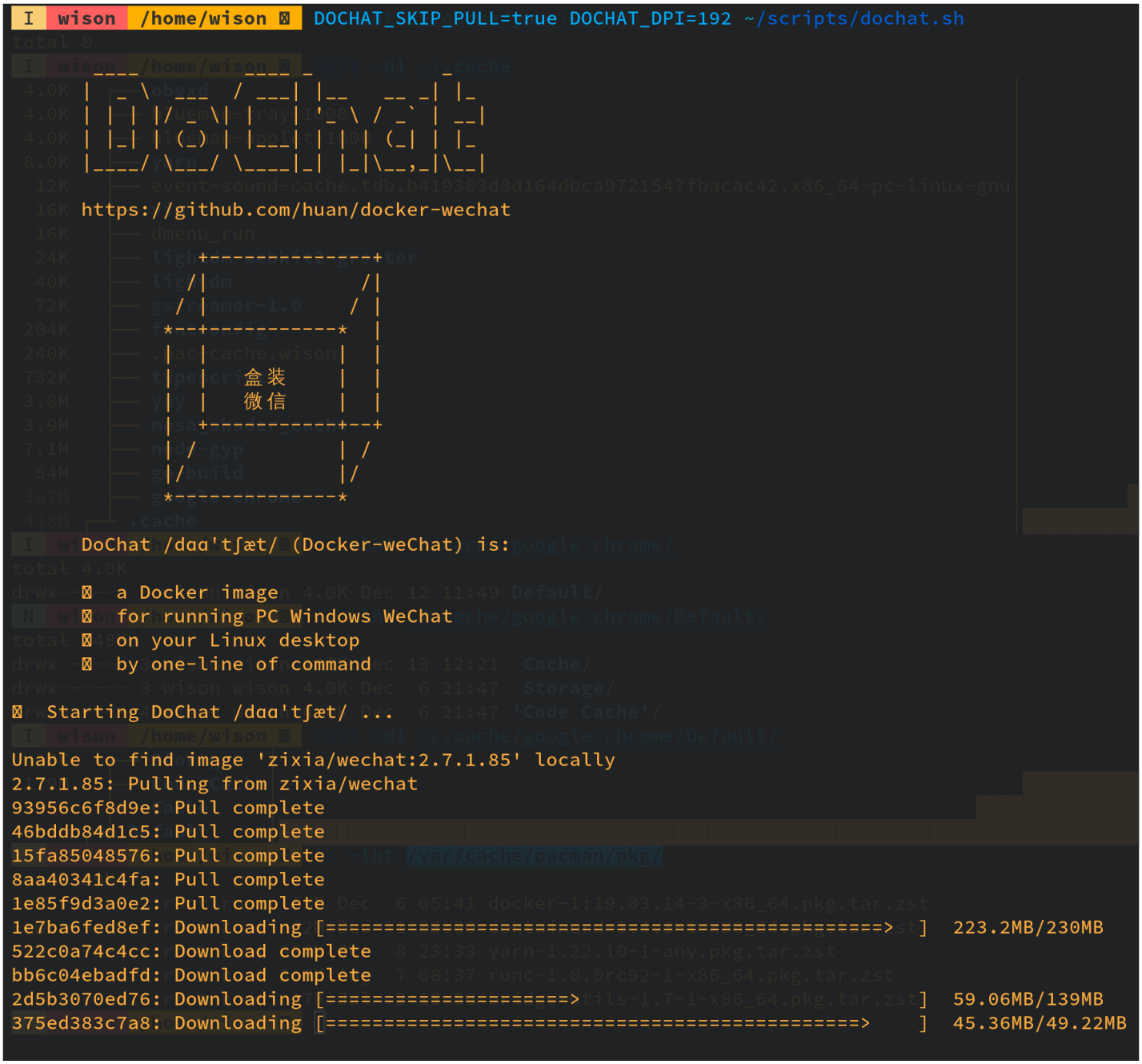
  For example in `fish` shell

```
# "DOCHAT_DEBUG=true": print out the debug info which can help if
start fail.
#
# "DOCHAT_DPI=192": for the display scale. Allows value below:
#       96  100%
#       120 125%
#       144 150%
#       192 200%
#
# "DOCHAT_SKIP_PULL=true": Don't pull latest docker image every time.
#
# For more detail informations, access here:
https://github.com/huan/docker-wechat
#
abbr startwebchat "DOCHAT_DEBUG=true DOCHAT_SKIP_PULL=true
DOCHAT_DPI=192 ~/scripts/dochat.sh"
```

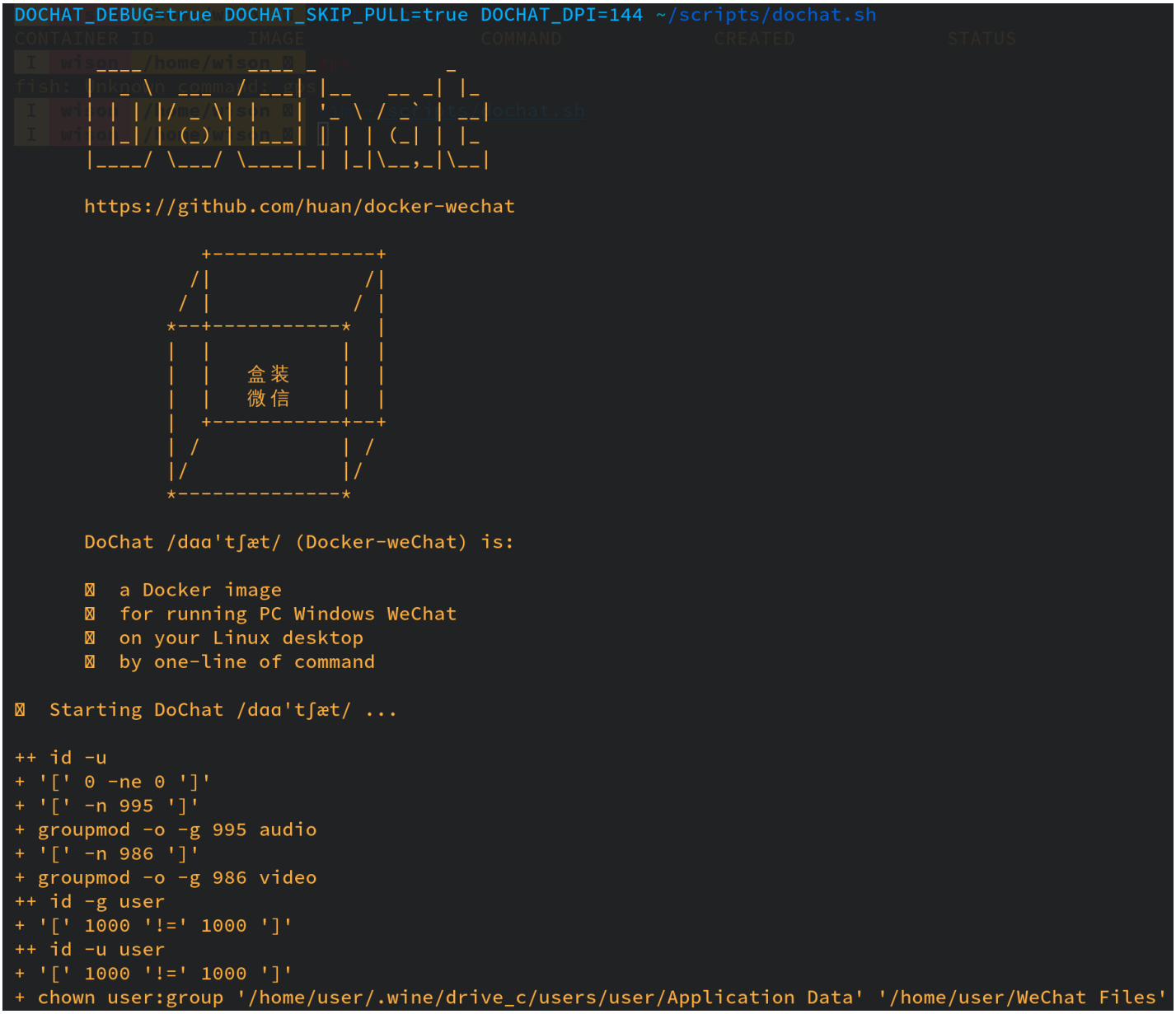
- Run it

```
# Make sure to run this command once to allow all users can access
your X
xhost +

startwebchat
```

When you run it at the first time, it will pull the image then run it.



After that, it should run:



Wine

What is `Wine`?

Wine is not an Emulator. It is a compatibility layer capable of running Windows applications. Wine translates Windows API into `POSIX` call on-the-fly, eliminating the performance and memory penalties of other methods and allowing you to cleanly integrate Windows applicants into your desktop.

- Install `wine` from `pacman`.

`sudo vim /etc/pacman.conf` and uncomment the lines below to enable the `multilib`:

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

Sync the databse:

```
sudo pacman -Sy
```

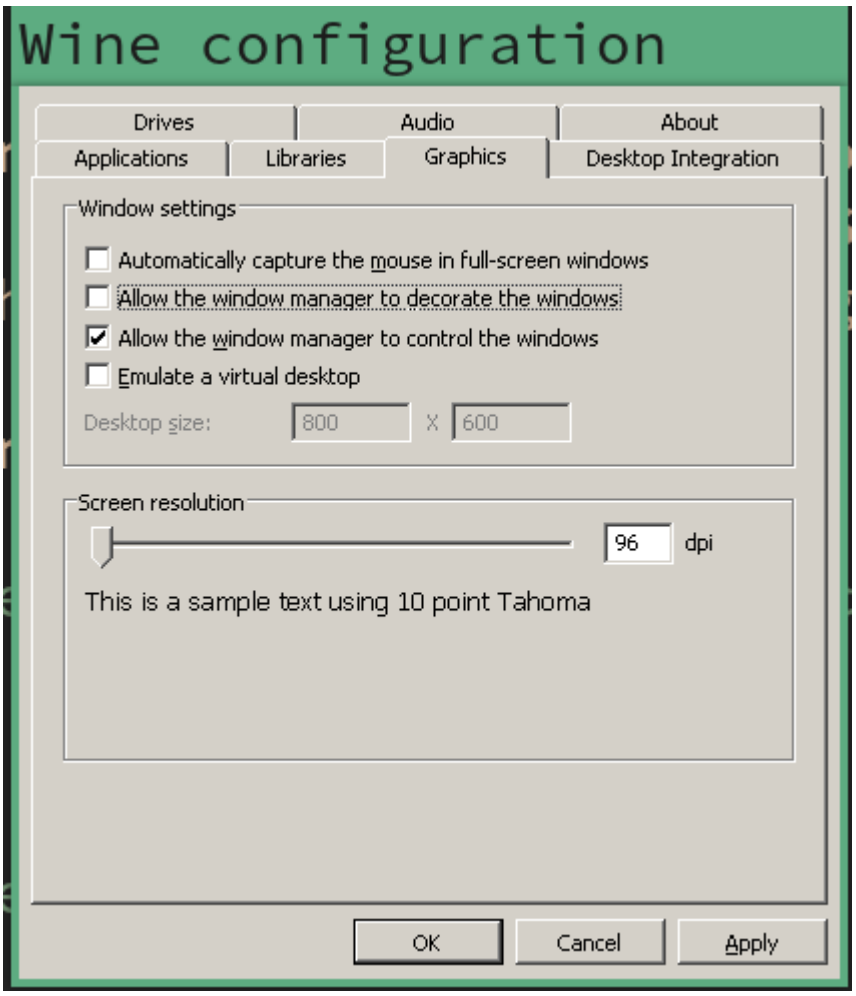
Then install it:

```
sudo pacman -S wine winetricks
```

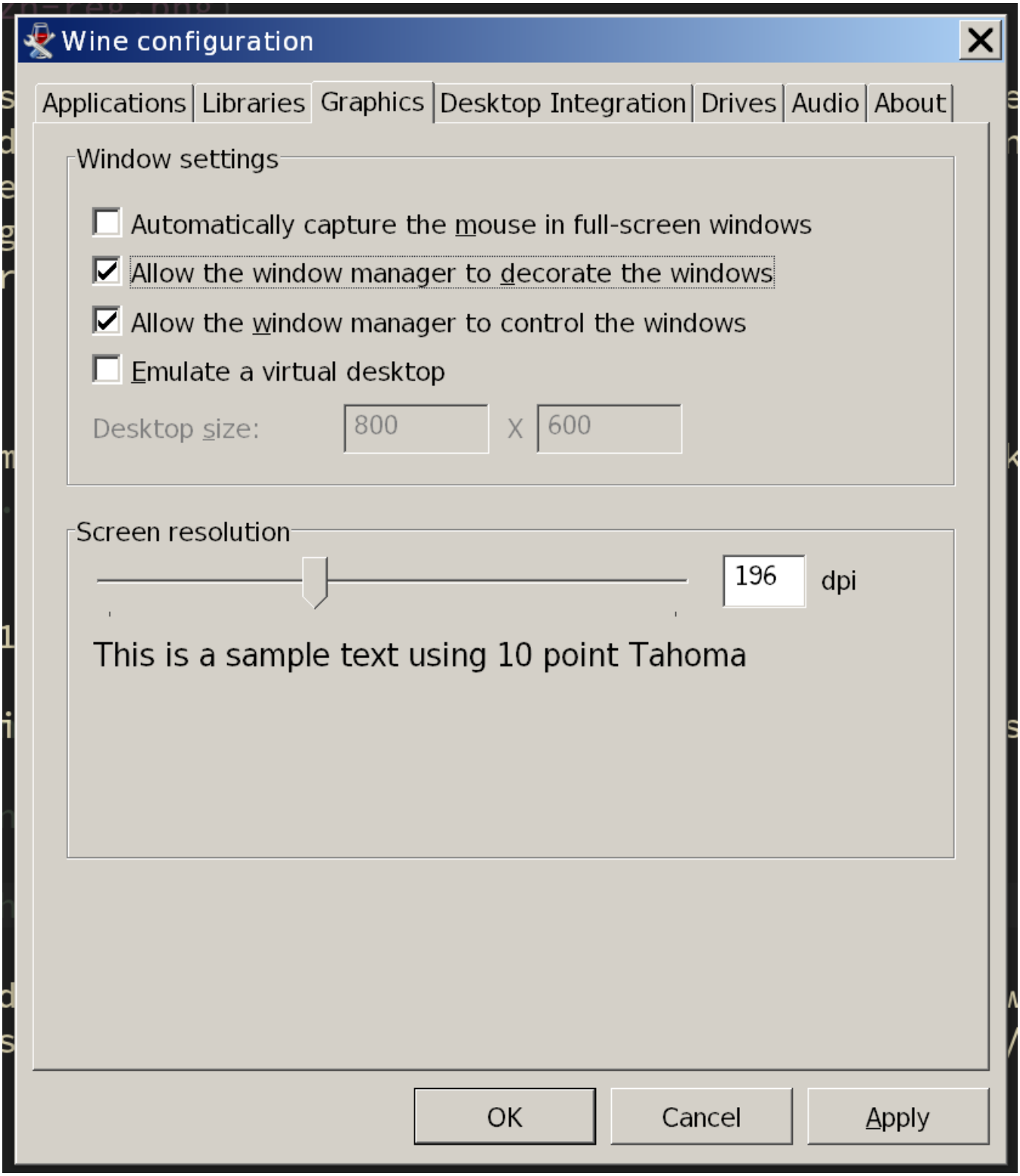
Run `winecfg` to create a default `C:/` in `$HOME/.wine/drive_c`. It will ask you to install some extra stuff, just click `cancel` for all of them, as it doesn't matter to run `wecaht`.

```
winecfg
```

For the first time, you might see the UI looks like below (very tiny font):



After you change the correct DPI, re-run `winecfg` again, it should look better:



- Link Chinese fonts to your local font

  Save the following content to a temporary file, you need to `import` it later, make sure NOT save into a `hidden` folder, as the `winecfg` program won't be able to see your `hidden` folder!!!

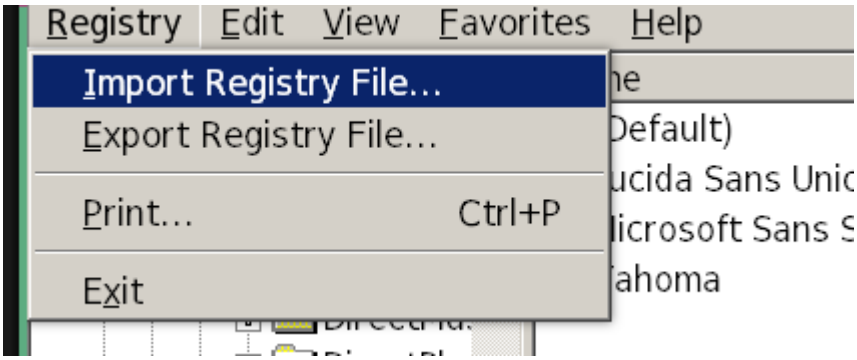
For example, save to `~/zh.reg`:

```
$ vim ~/zh.reg

# You can find the font name in `/usr/share/fonts`, just replace
"wqy-microhei.ttc"
# to the one you like. e.g. "SourceHanSansCN-Medium.otf"

REGEDIT4
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\FontLink\SystemLink]
"Lucida Sans Unicode"="wqy-microhei.ttc"
"Microsoft Sans Serif"="wqy-microhei.ttc"
"Microsoft YaHei"="wqy-microhei.ttc"
"微软雅黑"="wqy-microhei.ttc"
"MS Sans Serif"="wqy-microhei.ttc"
"Tahoma"="wqy-microhei.ttc"
"Tahoma Bold"="wqy-microhei.ttc"
"SimSun"="wqy-microhei.ttc"
"Arial"="wqy-microhei.ttc"
"Arial Black"="wqy-microhei.ttc"
"宋体"="wqy-microhei.ttc"
"新細明體"="wqy-microhei.ttc"
```

Run the command below to import `zh.reg`:

```
wine regedit
```

Then click `Import Registry File` and choose your `zh.reg`:



- Download `DLLs` by running `winetricks`

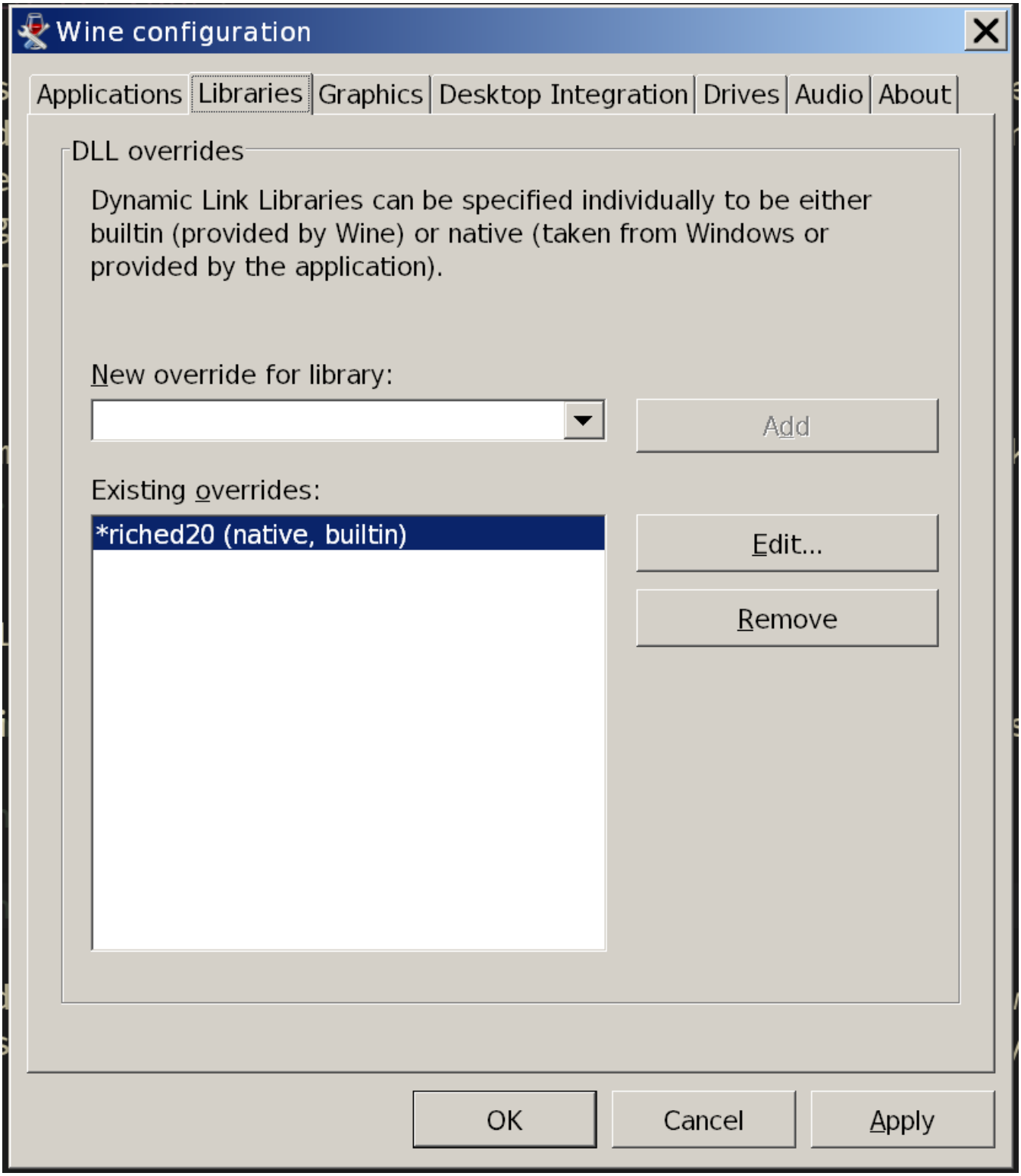
  ```
  winetricks riched20
  ```

  It will download the dependencies into `$HOME/.cache/winetricks` then install it into `$HOME/.wine/drive_c`.

  Pay attention:

  ```
    If you can't download `W2KSP4_EN.EXE`, then you can download it
  from:

    http://www.mywindowspage.com/index.php/download/w2ksp4_en-exe/

    After downloaded, move it into
  `$HOME/.cache/winetricks/wine2ksp4/`,
    and then run `winetricks riches20` again
  ```

  After installed the `DLLs`, if you run `winecfg` again, you should be able to see the `DLL` there:



- Install `Wechat` or `WxWork`

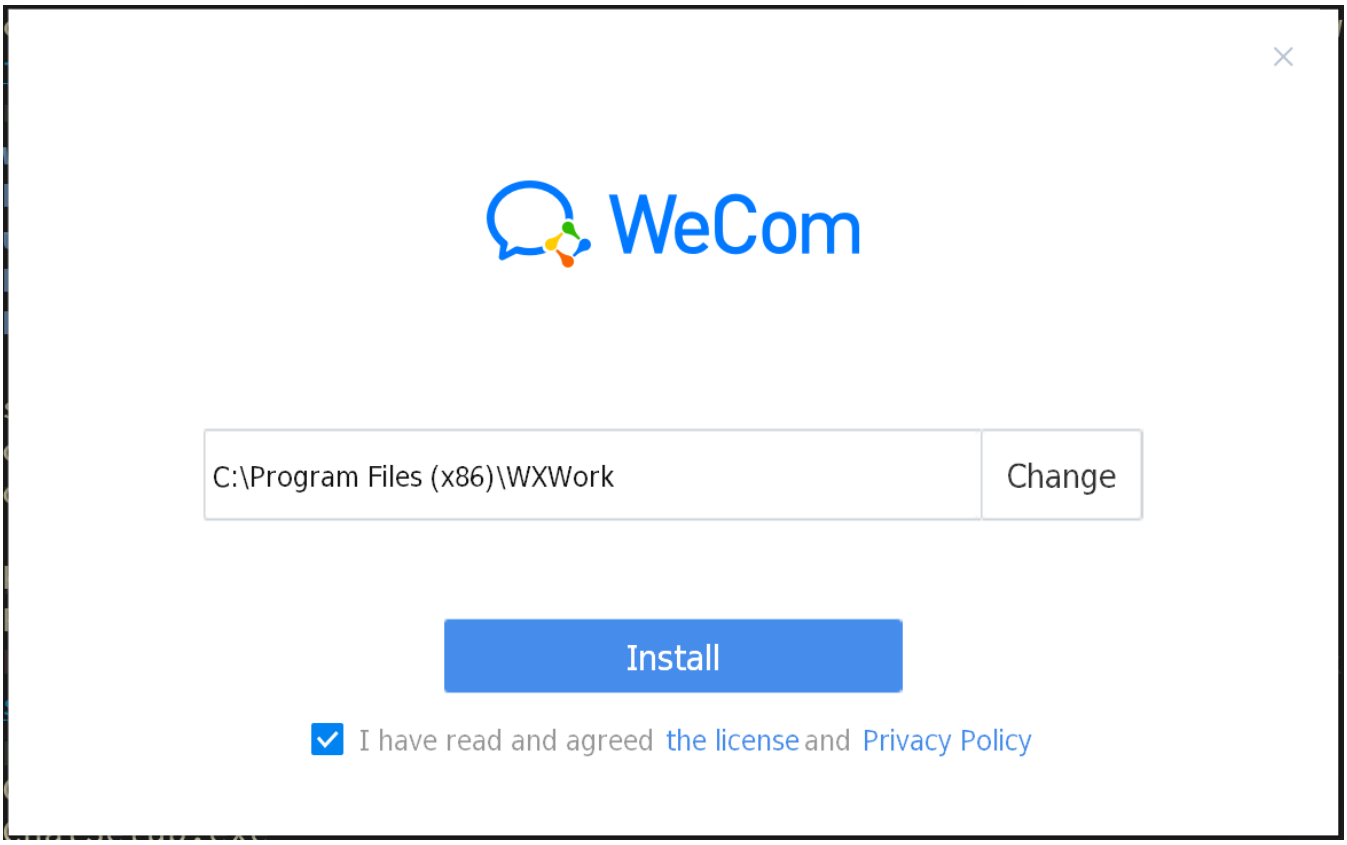
  Download the window version installation EXE from here, just pick the one you needed:

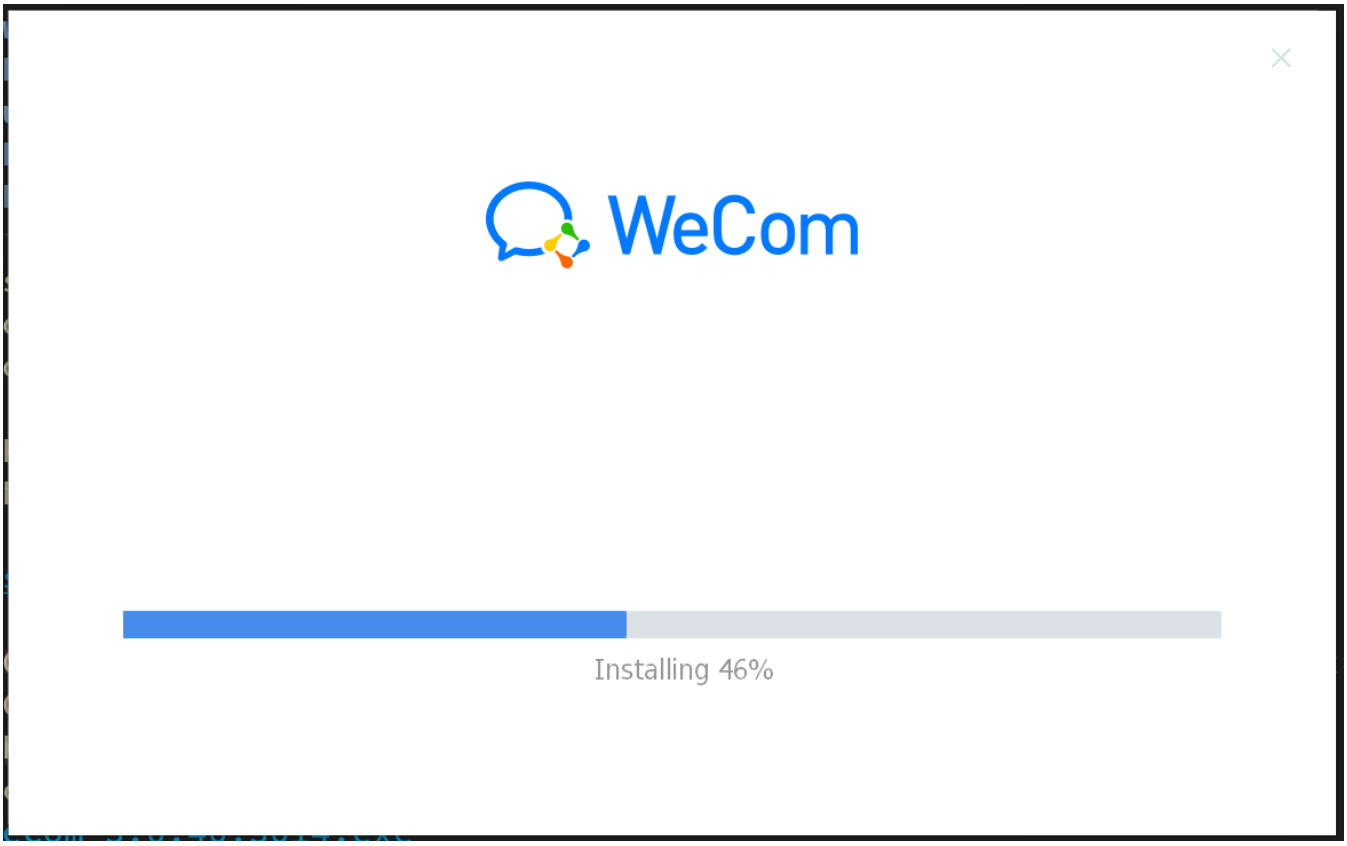
- WxWork
- Wechat

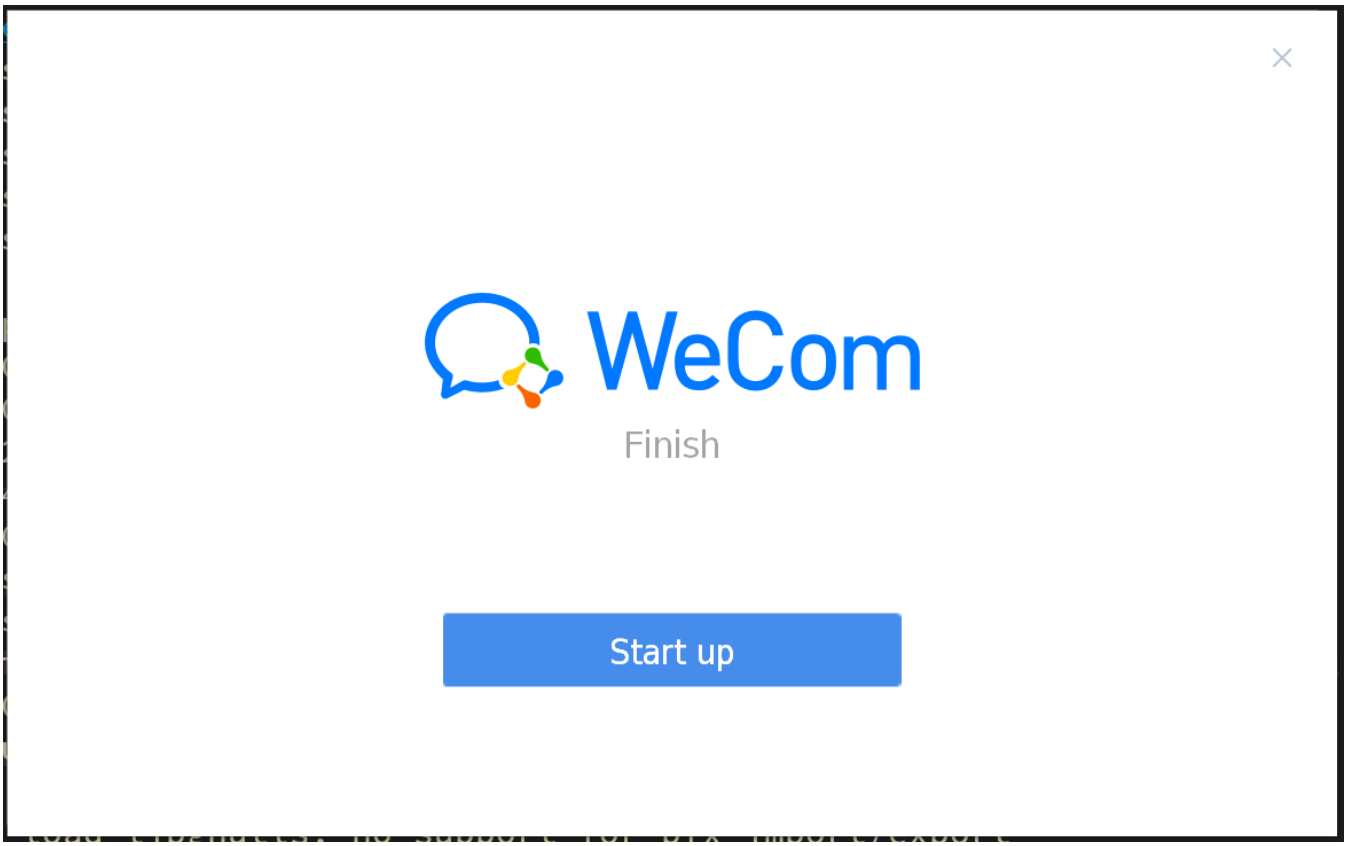
After download, run the installation program you needed. Let's take the `WxWork` as example:

```
# If you need `WxWork`
wine $HOME/Downloads/WE_CHAT_WORK_EXE_FILE_NAME_HERE
```

Then, you should be able to see the install UI like below:




WeCom
Installing 46%


WeCom
Finish
Start up

- How to use `i3` keybinding to lanuch `WxWork`

- Create lanuch script

  `vim ~/scripts/wechat_work.sh` with the following content:

  If you change the default installation folder, then you should change to yours!!!

  ```
  wine ~/.wine/drive_c/Program\ Files\ \(x86\)/WXWork/WXWork.exe &
  ```

  Make it executable:

  ```
  chmod +x ~/scripts/wechat_work.sh
  ```

- `i3` keybinding

  `vim ~/.config/i3/config` and add the following keybinding to lanuch `WxWork`:

  ```
  # Launch WxWork
  bindsym $mod+w exec --no-startup-id ~/scripts/wechat_work.sh

  # Enable the floating mode for `WxWork` window
  for_window [class="wxwork.exe"] floating enable border pixel 1
  ```

  A tips:

  As `WxWork` runs in floating mode and its UI is bigger, for getting the better user experience, you might switch to a empty workspace to run the `WxWork`.

- How to remove `wine` and `winetricks` completely?

  If you don't like it, walk through the steps below to remove them completely:

```
# Remove the package and all related dependencies
sudo pacman -Rsn wine winetricks

# Remove the entire `~/.wine`, as that's huge!!!
rm -rf ~/.wine

# Remove the cache as well
rm -rf ~/.cache/winetricks/
```

Install zoom

- Install

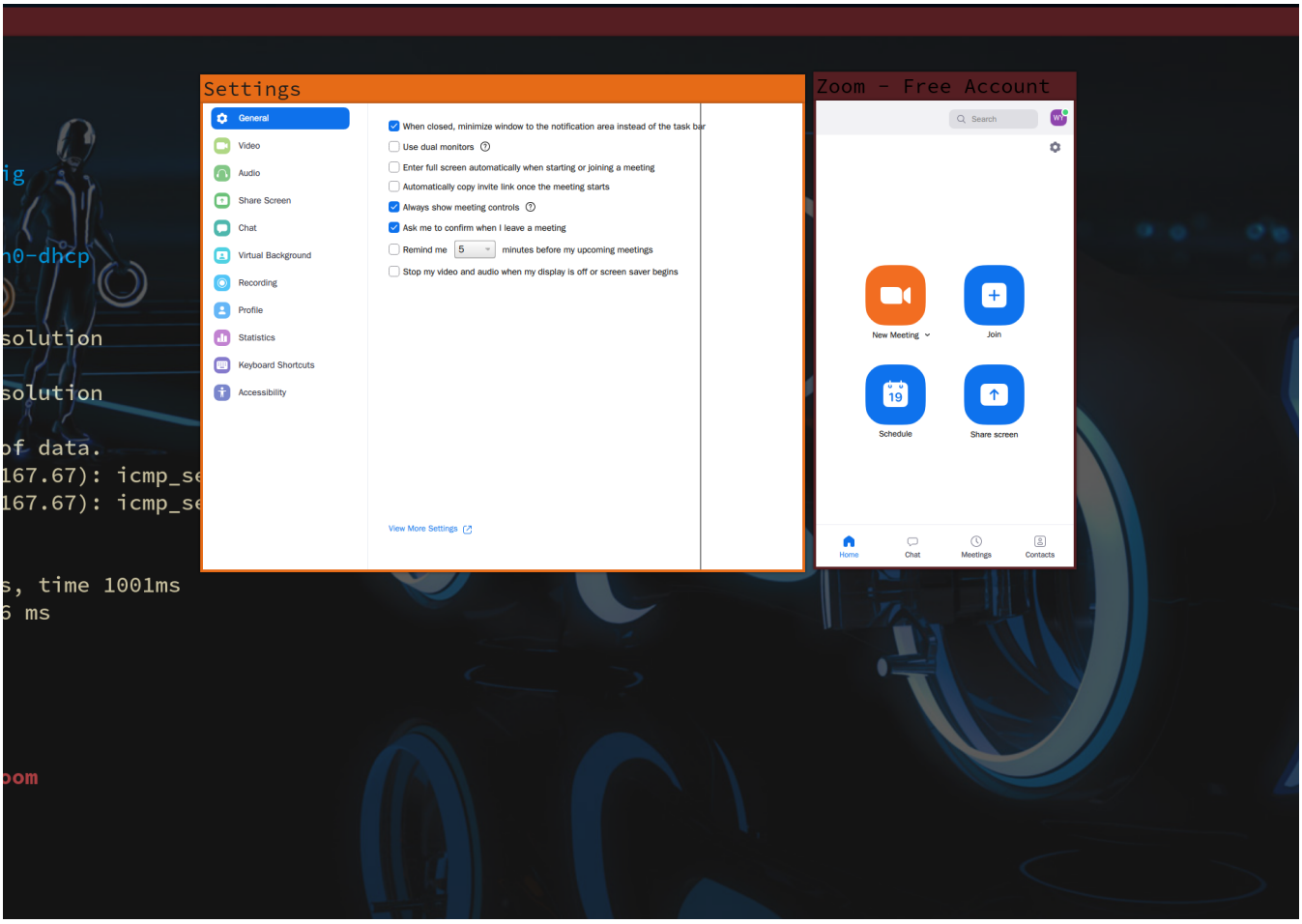
  Download `zoom_x86_64.pkg.tar.xz` from here

  After that, install it by running:

  ```
  sudo pacman -U ./zoom_x86_64.pkg.tar.xz
  ```

- Fixed the UI font size

  Mayb you will see the UI has very tiny font size like this:



If that happens, then add the `QT_SCALE_FACTOR` environment var to fix that. You can try `1` or `2` even `3` to have a look.

- Add hotkey to `i3`

  `vim ~/scripts/startzoom.sh` with the following settings:

```
#!/bin/sh
QT_SCALE_FACTOR=2 zoom &
```

  Change the `QT_SCALE_FACTOR` value to yours.

  Then add the following settings to `~/.config/i3/config`

```
# Zoom
bindsym $mod+z exec --no-startup-id ~/scripts/startzoom.sh
```

About cleaning cache

After running `Arch Linux` for a while, you will notice that your disk free space keeps reducing and never stop, that should be a signal that you should clean your cache. Usually, it will save over `GB` disk space:)

```
# Show how much your cache hold the disk space
dust -d1 ~/.cache

# Clean yay cache and unneeded dependencies
yay -Sc
yay -Yc

# Clean pacman cache (it locates `/var/cache/pacman/pkg/`)
sudo pacman -Scc

# Clean yarn (if you installed)
yarn cache clean

# Clean google-chrome cache (sometimes, this folder is huge!)
rm -rf ~/.cache/google-chrome/Default

# After that, calculate again, it should get big improved.
dust -d1 ~/.cache
```

Backup and restore

- Backup

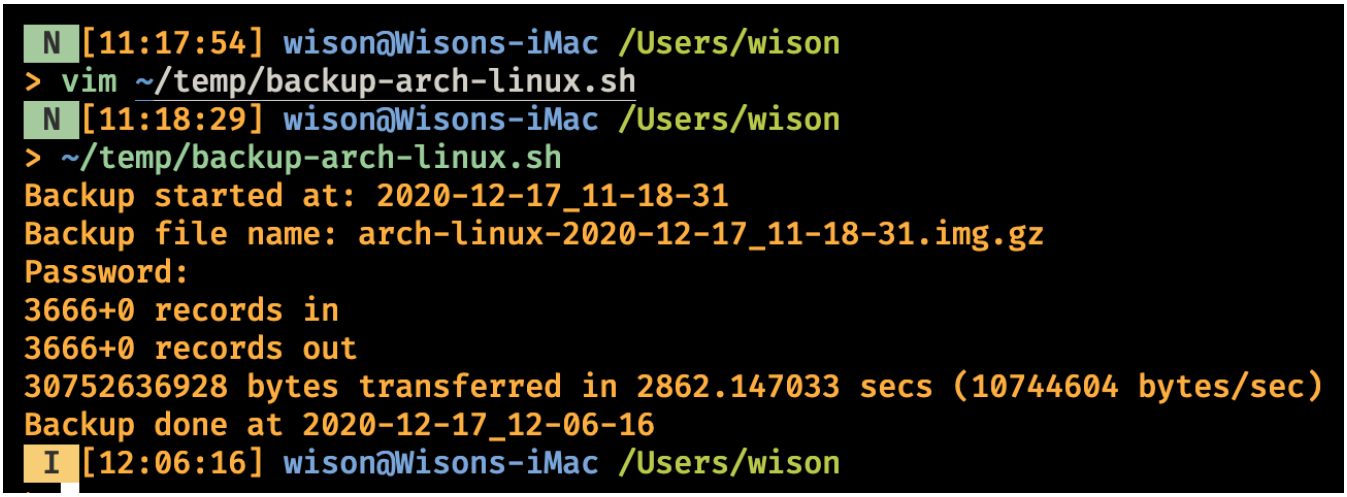
  Create the backup script with following content:

  ```
  #!/bin/bash
  echo "Backup started at: $(date +'%Y-%m-%d_%H-%M-%S')"
  BACKUP_FILE_NAME="arch-linux-$(date +'%Y-%m-%d_%H-%M-%S').img.gz"
  echo "Backup file name: $BACKUP_FILE_NAME"
  sudo dd if=/dev/disk2 bs=8m | gzip -c > $BACKUP_FILE_NAME
  echo "Backup done at $(date +'%Y-%m-%d_%H-%M-%S')"
  ```

  Of course, you need to change something above which match your real case:

```
- `BACKUP_FILE_NAME`, change to your prefer name (with path)

- `/dev/disk2`, change to your real dev name
```

In my case, I backup the `32GB` USB to `.img.gz`, it takes around `47 mins` and the compress size is `14GB`.




```
14G 17 Dec 12:06 arch-linux-2020-12-17_11-18-31.img.gz
```

- Restore

  Prepare your USB, and make sure the size should be equal or great than your original backup USB size.

  Better to remove all the partitions before you restore, I think it can make sure the bootloader works fine. (not 100% sure)

  Then create the restore script with the folling content:

```
#!/bin/bash
echo "Restore started at: $(date +'%Y-%m-%d_%H-%M-%S')"
RESTORE_FILE_NAME="arch-linux-$(date +'%Y-%m-%d_%H-%M-%S').img.gz"
echo "Restore file name: $RESTORE_FILE_NAME"
sudo bash -c 'gunzip -c ${RESTORE_FILE_NAME} | sudo dd of=/dev/disk2
bs=8m'
echo "Restore done at $(date +'%Y-%m-%d_%H-%M-%S')"
```

Firewall

- Install

```
sudo pacman -Sy firewalld ipset
```

- About the `zone` and `service`
  - What is a `zone`

    Long story short, a zone just like a `folder` contains predefined firewall rules.

  - Predefined `services`

    A service is a combination of port and/or protocol rules.

- Available `zones`

  Here only list the usual zone which user will use, for more details plz access the man page by running: `man firewalld.zones`.

  - `drop`

    The high secure option, no service open by default. Any incoming network packets are dropped, there is no reply. Only outgoing network connections are possible.

```
sudo firewall-cmd --info-zone=drop
#drop
#  target: DROP
#  icmp-block-inversion: no
#  interfaces:
#  sources:
#  services:
#  ports:
#  protocols:
#  forward: no
#  masquerade: no
#  forward-ports:
#  source-ports:
#  icmp-blocks:
#  rich rules:
```

- `public` (The default zone)

  For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.

```
sudo firewall-cmd --info-zone=home
#home
#  target: default
#  icmp-block-inversion: no
#  interfaces:
#  sources:
#  services: dhcpv6-client mdns samba-client ssh
#  ports:
#  protocols:
#  forward: no
#  masquerade: no
#  forward-ports:
#  source-ports:
#  icmp-blocks:
#  rich rules:
```

- Configuration

  All configurations in `firewalld` separate in `runtime` and `permanent`, `runtime` means only take effect from now on, but will not exists after reboot. If you run the command without `--permanent`, that means only affect to the `runtime` configuration.

```
# List the default zone
firewall-cmd --get-default-zone

# List all available zones
firewall-cmd --get-zones

# List all enabled services under current zone (`public` by default)
sudo firewall-cmd --list-service
sudo firewall-cmd --list-service --permanent

# List all available services which you can add or remove
firewall-cmd --get-services

# Add the specified (enable) sevice to the current zone
sudo firewall-cmd --add-service SERVICE_NAME_HERE
sudo firewall-cmd --add-service SERVICE_NAME_HERE --permanent

# Remove the specified sevice to the current zone
sudo firewall-cmd --remove-service SERVICE_NAME_HERE
sudo firewall-cmd --remove-service SERVICE_NAME_HERE --permanent

# Usually, all setting commands are instancely.
# But you can reload all the rules at any time.
sudo firewall-cmd --reload
```

  But you can config all the rules in a `GUI` tool:

```
sudo firewall-config
```

Change `DPI` by script

If you install `Arch Linux` to a **USB**, then you definitely need to change screen `DPI` when you switch between `iMac` and `MacBookPro`.

- **`iMac (5K)`**
  - Create `~/scripts/Xresources-imac` with the following settings:

    ```
    Xft.dpi: 144
    ```

  - Create `~/scripts/change-dpi-for-imac.sh` with the following settings:

    ```
    #!/bin/bash
    cp -rvf ~/scripts/Xresources-imac ~/.Xresources
    i3exit logout
    ```

  - Make it executable

    ```
    chmod + x~/scripts/change-dpi-for-imac.sh
    ```

- **`MacBookPro 2015`**
  - Create `~/scripts/Xresources-mbp-2015` with the following settings:

    ```
    Xft.dpi: 192
    ```

  - Create `~/scripts/change-dpi-for-mbp-2015.sh` with the following settings:

```
#!/bin/bash
cp -rvf ~/scripts/Xresources-mbp-2015 ~/.Xresources
i3exit logout
```

    - Make it executable

```
chmod + x~/scripts/change-dpi-for-mbp-2015.sh
```

- `MacBookPro 2012`
    - Create `~/scripts/Xresources-mbp-2012` with the following settings:

```
Xft.dpi: 96
```

    - Create `~/scripts/change-dpi-for-mbp-2012.sh` with the following settings:

```
#!/bin/bash
cp -rvf ~/scripts/Xresources-mbp-2012 ~/.Xresources
i3exit logout
```

    - Make it executable

```
chmod + x~/scripts/change-dpi-for-mbp-2015.sh
```

After that, you can change to any screen DPI at anytime you want in real-time.

Pay attention: After you run the script, it will log you out to take effect!!!

```
# For iMac 5K Screen
~/scripts/change-dpi-for-imac.sh

# For MacBookPro 2015 screen
~/scripts/change-dpi-for-mbp-2015.sh

# For MacBookPro 2012 screen
~/scripts/change-dpi-for-mbp-2012.sh
```

Screen brightness control

The basic approach to control screen brightness

```
# Print out the current screen brightness
cat /sys/class/backlight/intel_backlight/brightness

# Print out the maximum screen brightness
cat /sys/class/backlight/intel_backlight/max_brightness

# Set the current screen brightness
sudo bash -c "echo 512 > /sys/class/backlight/intel_backlight/brightness"
```

For the `iMac`

Sometimes `acpi` doesn't work very well on `iMac`, for the case I met, the `iMar 5K` screen is very bright by default and can't control via the `/sys/class/backlight` file system path.

Even you can't control the brightness, but it's reduce the default brightness (at least not 100% brightness). For that purpose, you can disable the `acpi` brightness control for the `iMac` via the steps below:

- `sudo vim /etc/default/grub` and add the `acpi_backlight=none` settings like below:

  ```
  # Assume `loglevel=3 quiet` is your original settings
  GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet acpi_backlight=none"
  ```

- Then re-generate the `GRUB` based on your new settings

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Build the `mac-light-controller` (it only work for `MacBookPro`)

- Pull from repo, build it

```
# Go into any folder you want
git clone https://github.com/wisonye/mac-light-controller.git
cd mac-light-controller
cargo clean && \
cargo build --release && \
strip ./target/release/mac-light-controller

# Move to any folder you want `i3` to launch from there (Optional)
mv ./target/release/mac-light-controller ~/scripts/
```

- Add keybinds to `i3` configuration file

  `vim ~/.config/i3/config` and add the following settings:

```
#
=============================================================================

# Screen brightness control
#
=============================================================================

bindsym XF86MonBrightnessUp exec --no-startup-id ~/scripts/mac-light-
controller Screen +
bindsym XF86MonBrightnessDown exec --no-startup-id ~/scripts/mac-
light-controller Screen -
bindsym XF86KbdBrightnessUp exec --no-startup-id ~/scripts/mac-light-
controller Keyboard +
bindsym XF86KbdBrightnessDown exec --no-startup-id ~/scripts/mac-
light-controller Keyboard -
```

- Allow any linux account in `wheel` group can modify the backlight system file with `root` permission
  - `sudo vim /etc/udev/rules.d/90-backlight.rules` and add the following settings:

    ```
    SUBSYSTEM=="backlight", ACTION=="add", \
        RUN+="/bin/chgrp wheel /sys/class/backlight/%k/brightness", \
        RUN+="/bin/chmod g+w /sys/class/backlight/%k/brightness"
    ```

  - `sudo vim /etc/udev/rules.d/91-leds.rules` and add the following settings:

    ```
    SUBSYSTEM=="leds", ACTION=="add", \
        RUN+="/bin/chgrp wheel /sys/class/leds/%k/brightness", \
        RUN+="/bin/chmod g+w /sys/class/leds/%k/brightness"
    ```

  Of course, make sure your current linux account is in `wheel` group!!!

Reload `i3`, then press the screen brightness control key, it should work. You can run the script manually like below to test it if it doesn't work.

```
DEBUG=true ./target/release/mac-screen-brightness-controller -
is_add_brightness: false
total_steps: 10
max_brightness: Some(1953)
current_brightness: Some(1953)
temp_step_value: 195
new_brightness_value: 1758
Set new brightness value '1758' successfully.
```

Trackpad support

- List your `Apple Trackpad` device

```
sudo libinput list-devices | grep Trackpad -A 17
```

```
# By default, this folder should empty if you didn't add any
configuration
# manually
ls -lht /etc/X11/xorg.conf.d/

# If you installed `synaptics` (`xf86-input-synaptics`) and `libinput`
(`xf86-input-libinput`),
# then you should see result like this:
ls -lht /usr/share/X11/xorg.conf.d/

-rw-r--r-- 1 root root 1.4K Dec  2 08:51 10-quirks.conf
-rw-r--r-- 1 root root   92 May 19  2020 10-amdgpu.conf
-rw-r--r-- 1 root root 1.4K May 19  2020 40-libinput.conf
-rw-r--r-- 1 root root   92 May 17  2020 10-radeon.conf
-rw-r--r-- 1 root root 1.8K May 17  2020 70-synaptics.conf

# If match the above case, then create to soft link like this:
sudo ln -s /usr/share/X11/xorg.conf.d/40-libinput.conf
/etc/X11/xorg.conf.d/40-libinput.conf

# After restart the X, run the command below and you should some ouput
# like below:
grep -e "Using input driver 'libinput'" /var/log/Xorg.0.log

[  5734.514] (II) Using input driver 'libinput' for 'Power Button'
[  5734.579] (II) Using input driver 'libinput' for 'Video Bus'
[  5734.633] (II) Using input driver 'libinput' for 'Power Button'
[  5734.687] (II) Using input driver 'libinput' for 'Sleep Button'
[  5734.729] (II) Using input driver 'libinput' for 'Apple Inc. Apple
Internal Keyboard / Trackpad'
[  5734.887] (II) Using input driver 'libinput' for 'Broadcom Corp.
Bluetooth USB Host Controller'
[  5734.927] (II) Using input driver 'libinput' for 'Broadcom Corp.
Bluetooth USB Host Controller'""
```

That proves your trackpad is using the `libinput` driver. So you can add the following setting to improve your trackpad experiences:

- Same scrolling experience within **`MacOS`**

  `sudo vim /etc/X11/xorg.conf.d/40-libinput.conf`, then add the following settings to `libinput touchpad catchall` section.

  - `Option "NaturalScrolling" "true"` - Reverse the scrolling direction
  - `Option "Tapping" "on"` - Tap to click

  Finally, it looks like this:

```
Section "InputClass"
        Identifier "libinput touchpad catchall"
        MatchIsTouchpad "on"
        MatchDevicePath "/dev/input/event*"
        Driver "libinput"
        Option "NaturalScrolling" "true"
        Option "Tapping" "on"
EndSection
```

Bluetooth support

- Install

```
sudo pacman bluez bluez-utils blueman
```

- Enable/start service

```
# If you want auto bluetooth service
sudo systemctl enable bluetooth.service

# You can restart it manually anytime you need input
sudo systemctl restart bluetooth.service
sudo systemctl status bluetooth.service
```

- Make sure `rfkill` not block your bluetooth adapter

  `rfkill` a tool for enabling and disabling wireless device.

```
sudo rfkill list

# 0: hci0: bluetooth
#       Soft blcoked: no
#       Hard blcoked: no
```

The case above means no block at all, that's fine. If you see it's blocked, then run the command below to unblock:

```
sudo rfkill unblock bluetooth
```

- Scan and pair, then connect

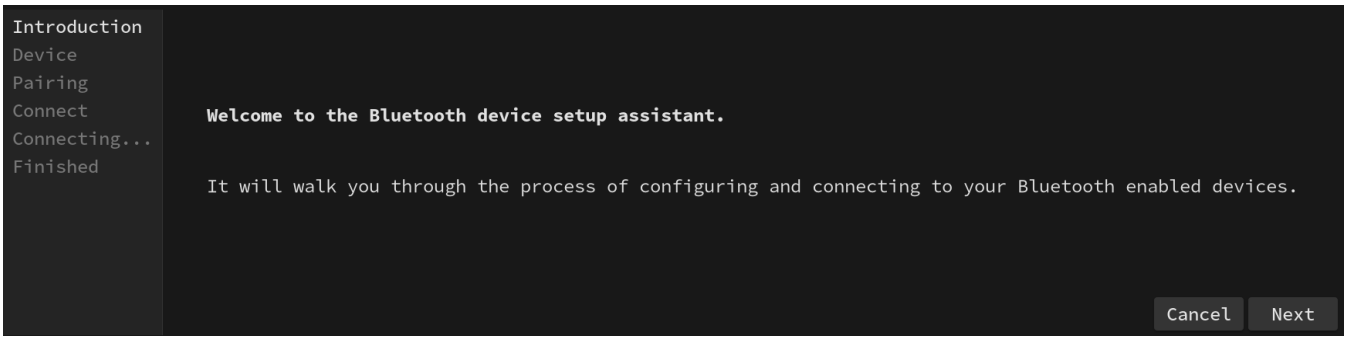
  First, turn off your bluetooth device which want to connect to. Run `bluetoothctl`, then follow the steps below to connect:

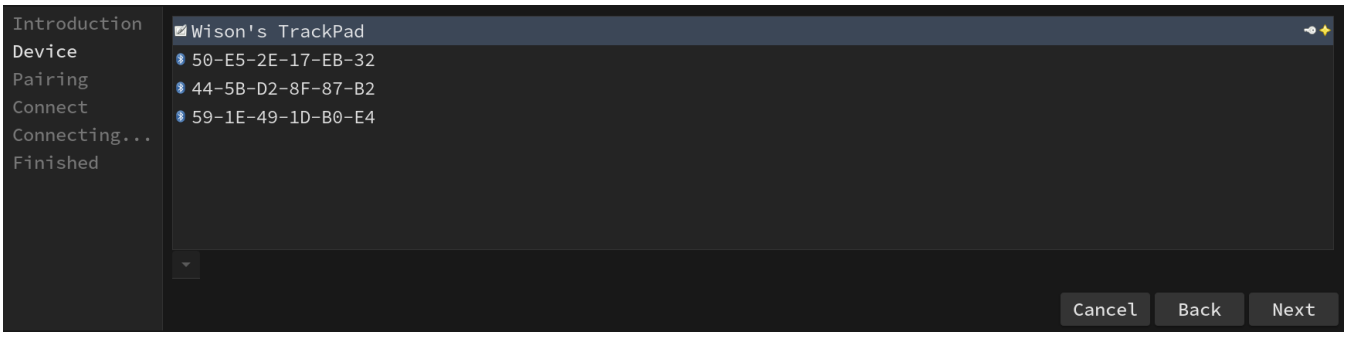
```
# Make sure turn on the bluetooth
power on

# Enable scan, after that, bluetooth devices show up there one by one
scan on

# Right now, turn on your bluetooth device, then wait for it to show
up.
# Hopefully, it shows its name directly which you can confirm that's
your device.
# If it doesn't, only show the MAC ID, then copy that Id and run the
# command to confirm.
info XX:XX:XX:XX:XX:XX

# Once you confirm that your device, then do:
pair XX:XX:XX:XX:XX:XX

# After pairing, you can connect to it
connect XX:XX:XX:XX:XX:XX

# Optionally, you can trust it and it will auto connect next time
trust XX:XX:XX:XX:XX:XX

# Quit
quit
```

Below is the real example:

```
N | wison@wison-arch ~> bluetoothctl
Agent registered
[bluetooth]# devices
Device 64:C7:53:ED:C2:77 Wison's TrackPad
[bluetooth]# connect 64:C7:53:ED:C2:77
Attempting to connect to 64:C7:53:ED:C2:77
[CHG] Device 64:C7:53:ED:C2:77 Connected: yes
Connection successful
[CHG] Device 64:C7:53:ED:C2:77 ServicesResolved: yes
[Wison's TrackPad]# trust 64:C7:53:ED:C2:77
Changing 64:C7:53:ED:C2:77 trust succeeded
[Wison's TrackPad]#
```

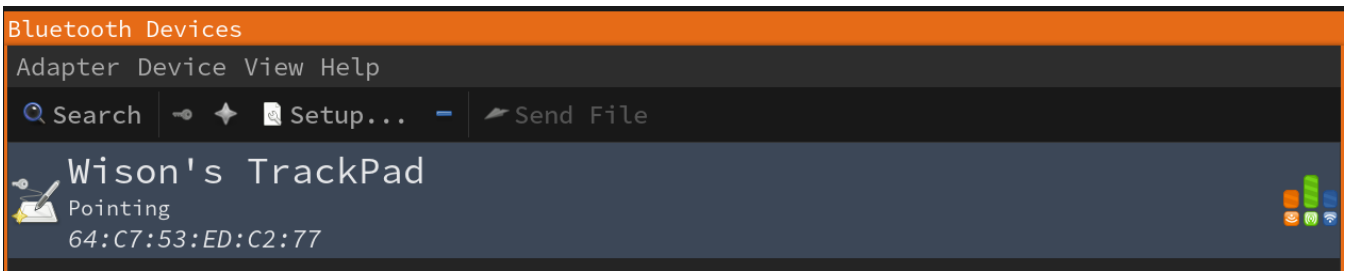
- Alternatively, you can use the `blueman-assistant` to do that in GUI mode:

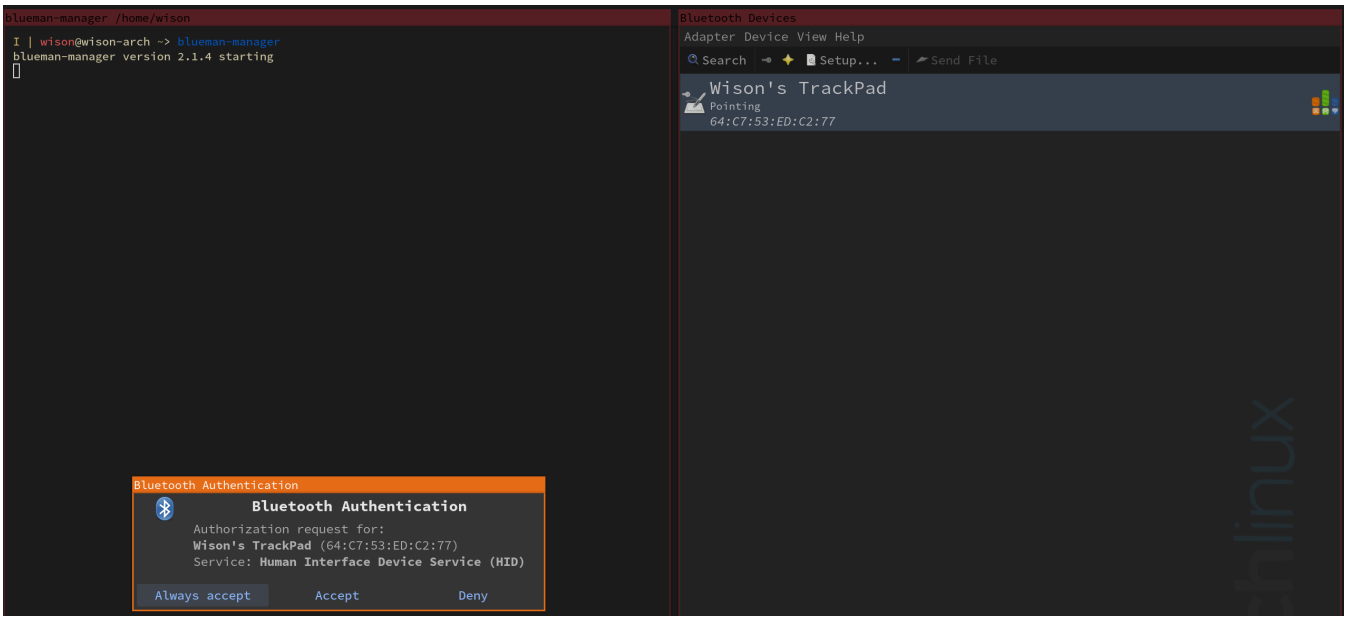




// Ignore the pair and connect steps……

- Another option, you can use `blueman-manger` directly which is another GUI tool:





Access Apple File System

In 2017, Apple changed the default filesystem on their `macOS` (High Sierra and above) to `APFS`, the `Apple File System`. It replaced `HFS+`.

For safety reason, you can use `apfs-fuse` to mount your MacOS drive within the read-only mode.

- Install

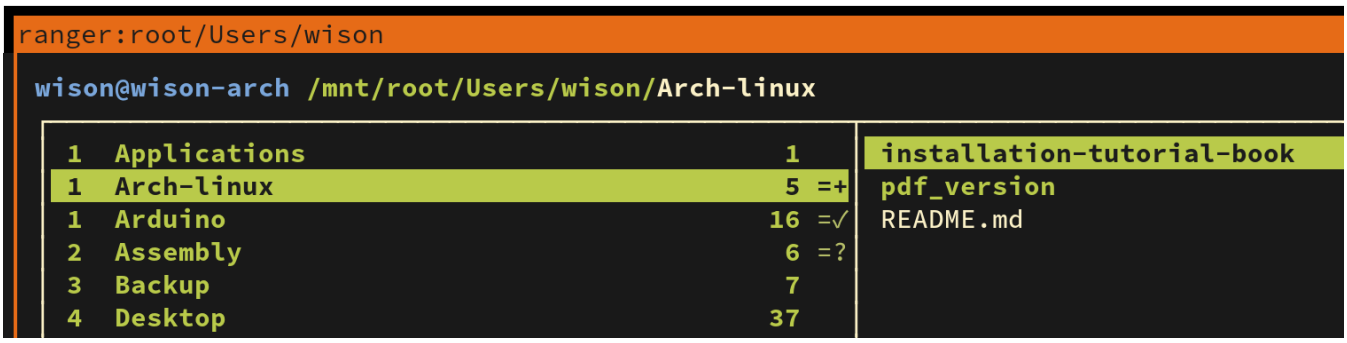
```
sudo pacman -Sy fuse3 zlib bzip2 cmake
```

- Compile

```
git clone https://github.com/sgan81/apfs-fuse.git
cd apfs-fuse/
git submodule init
git submodule update

mkdir build && cd build
cmake ..
make
sudo make install
```

- Mount and umount

```
# Mount with `allow_other` (can access) option
# Plz replace the `sdXX` to your real device
sudo apfs-fuse -o allow_other /dev/sdXX /mnt
```



```
# Unmount
sudo umount /mnt
```