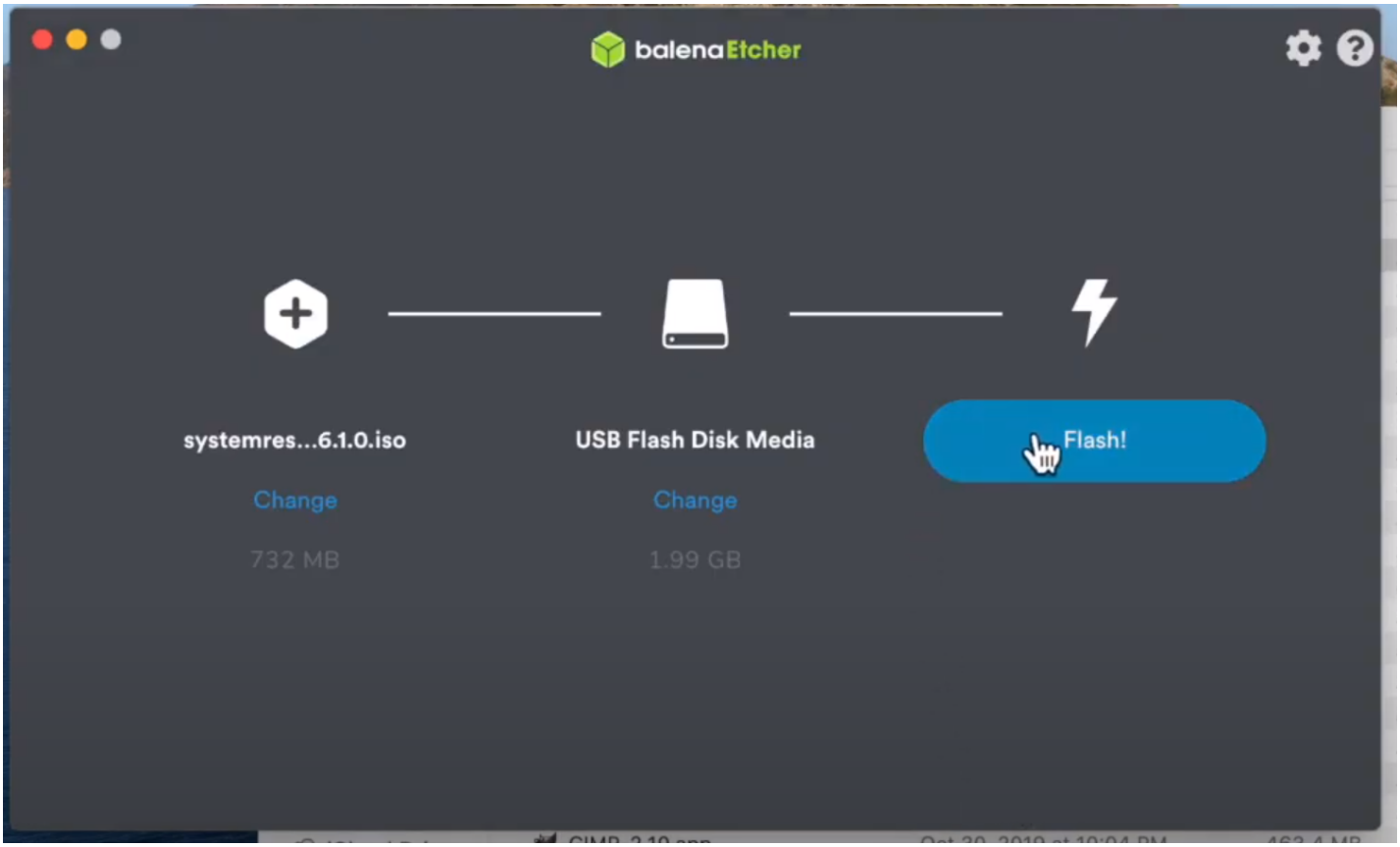
Prepare your live ISO USB

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`.



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
```

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`
    ```

  - 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!!!

# 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
```

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
```

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
```

- Localization

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

  But the current linux doesn't have the `vim` yet, then we need to `exit` current Arch and back to `Live Arch` to edit it.

```
# Exit current Arch root.
exit

# Now, we're in `Live Arch` and edit the file.
# Enable `en_US.UTF-8 UTF-8` line, save and exit.
vim /mnt/etc/locale.gen

# Then, change root back to new Arch Linux to genereate the locale
# settings.
arch-chroot /mnt
locale-gen

# Exit current Arch root.
exit

# Then edit LANG settings.
# Add `LANG=en_US.UTF-8`.
# save and exit.
vim /mnt/etc/locale.conf

# Then edit the keyboard settings
vim /mnt/etc/vconsole.conf

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

- Hostname and host settings
  - `vim /mnt/etc/hostname`, set to your hostname.
  - `vim /mnt/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
```

- Change back into the new `Arch Linux` and set root password

```
arch-chroot /mnt
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`:

```
# Back to `Live Arch`
exit

# Editr the config
vim /mnt/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.
```

  Now go back to new `Arch Linux` and regenerate the initial RAM disk image with the changes made:

```
arch-chroot /mnt

mkinitcpio -p linux
```

- 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:

```
# We're inside installed Arch
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.

```
# Back to `Live Arch`
exit

# Editr the config
vim /mnt/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 /mnt/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:

```
# Go inside the installed Arch
arch-chroot /mnt

# `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
```

- 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/
```

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 dialog 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
  ```

  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
  ```

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 -S gvim
```

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
```

- 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.

Pay attention:

In this tutorial, I still can't connect to **5G**, I think should be the driver issue, as the result below doesn't include **5G** channel

```
iw dev

# phy#0
#   Interface wlan0
#       ifindex 2
#       wdev 0x1
#       addr xx:xx:xx:xx:xx (mac address)
#       type managed
#       channel 1 (2412 MHz), width: 20 MHz, center1: 2312 MHz
```

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 `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.

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.

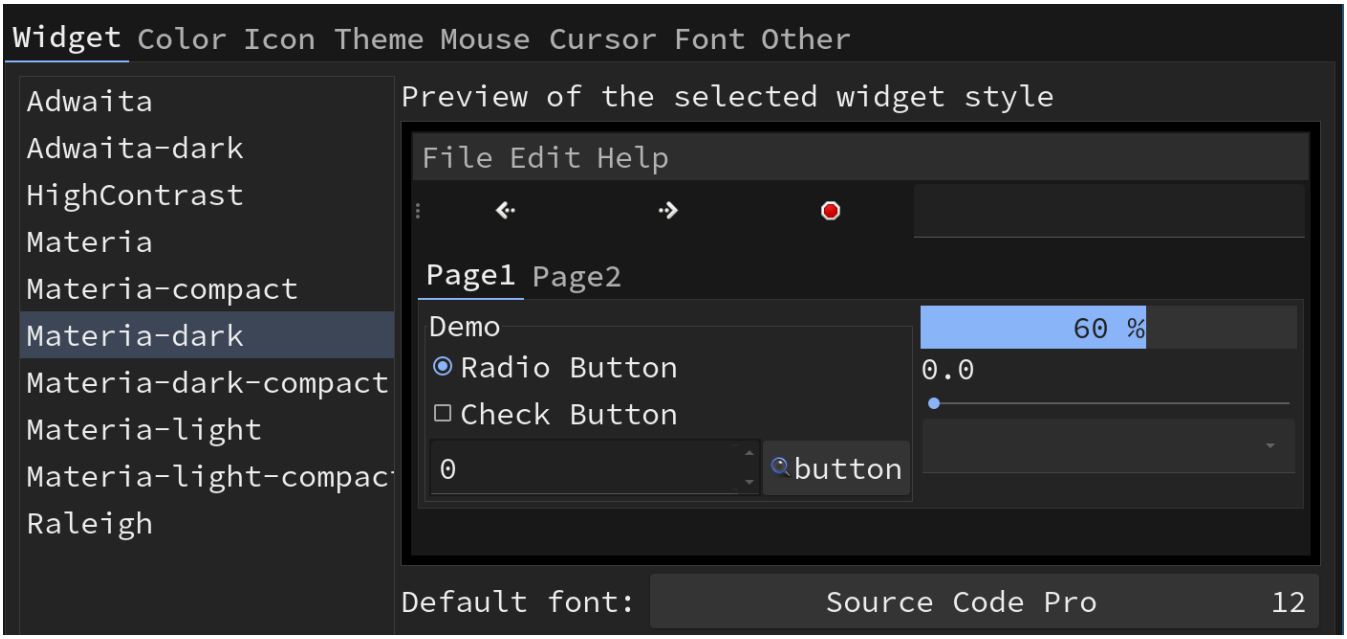
- Install themes and configuration tools

  ```
  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+`

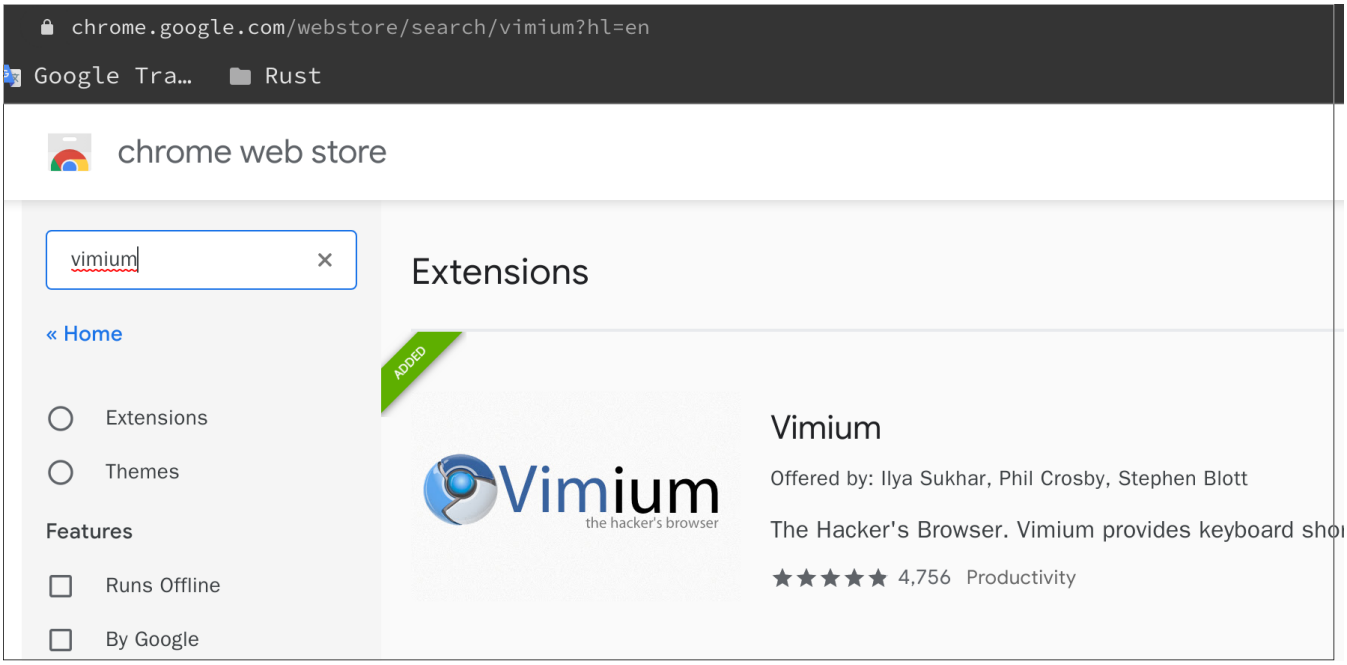
- Install `xdg-utils` and set `Chrome` as the default browser

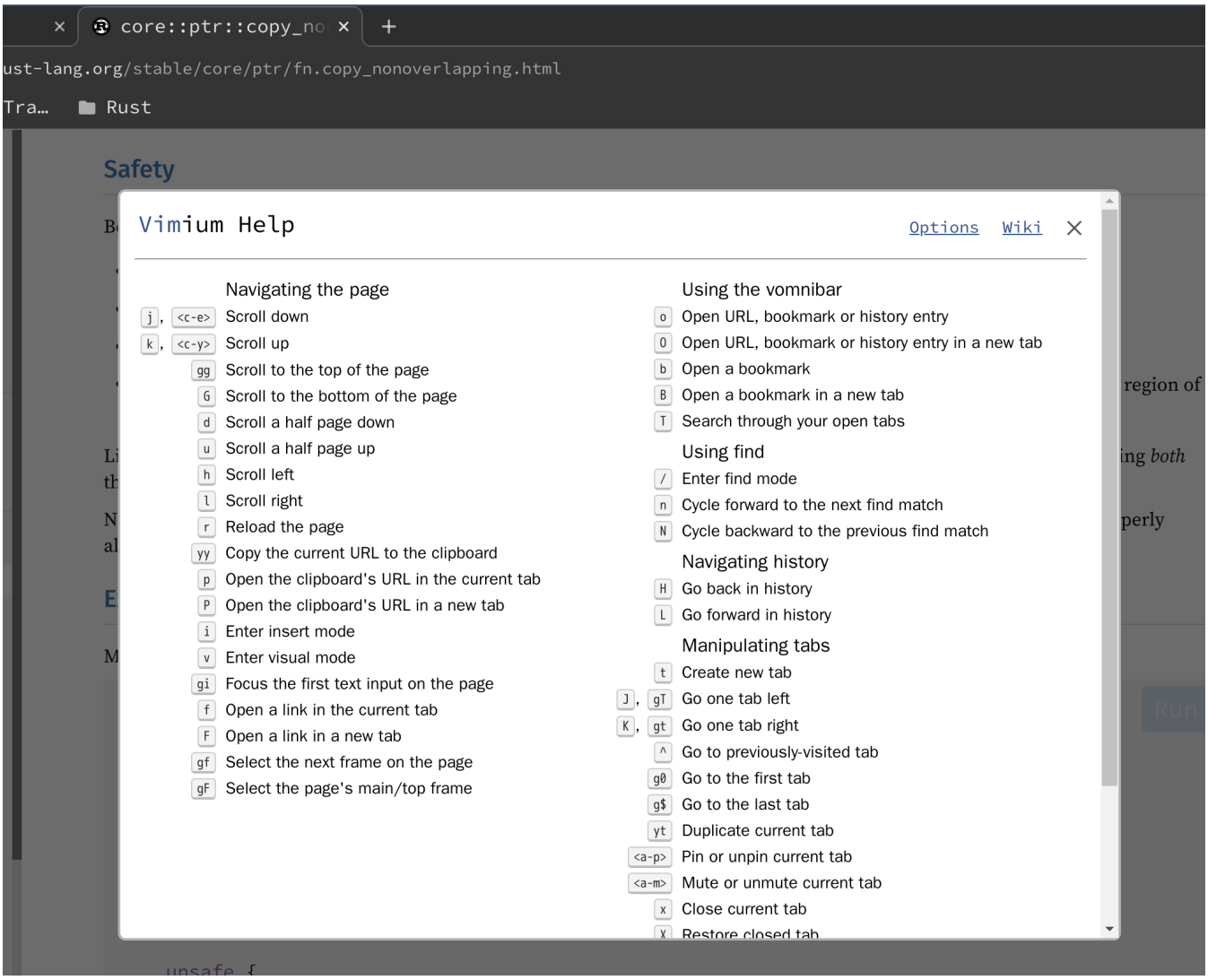
```
sudo pacman -Sy xdg-utils

# 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/`.

- Optional choice: `vimium` extension if you're `vim` user:)

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





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
```

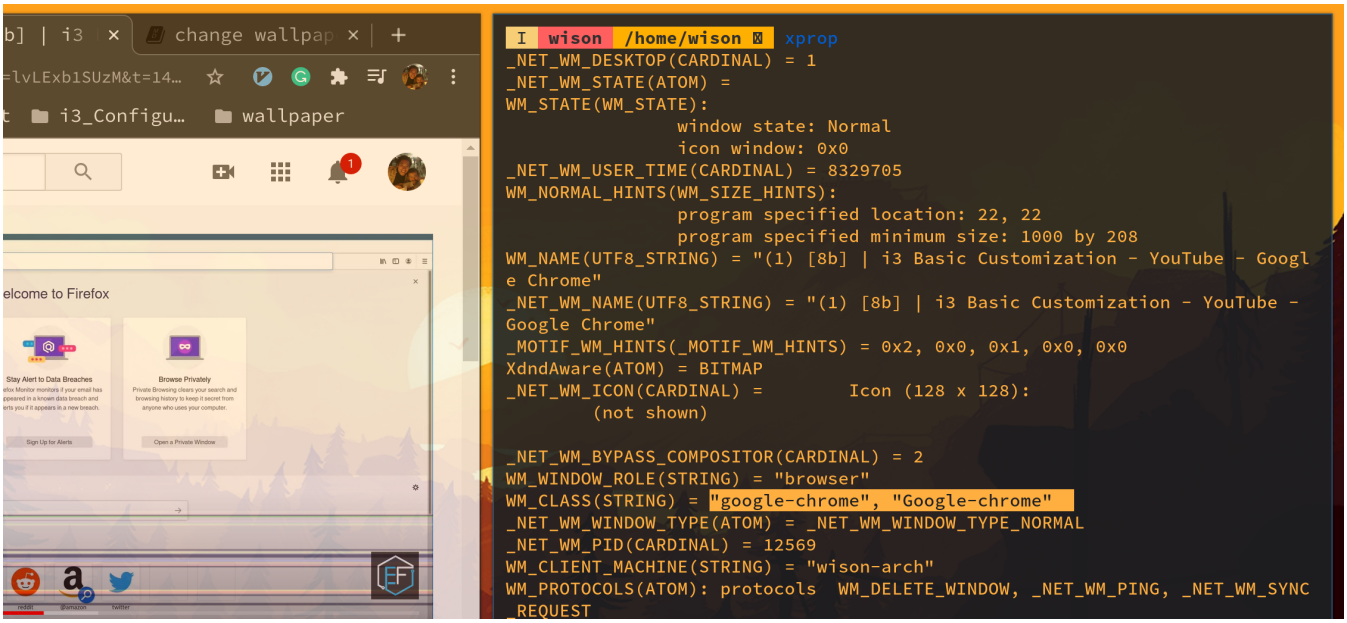
After that, open `https://fontawesome.com/cheatsheet` and copy any icon, then you can paste into any configuration file directly. You need to relogin to take effect.

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!!!:)

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:SourceCodePro 11

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

# 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
exec_always compton

# Load chinese input method
exec_always fcitx

# Load wallpaper
exec_always feh --bg-fill ~/Photos/wallpaper/6.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@ +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


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

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

# split in vertical orientation
# bindsym $mod+v split v

# 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 or $mod+r
        bindsym Return mode "default"
        bindsym Escape mode "default"
        bindsym $mod+r mode "default"
}
bindsym $mod+r mode "resize"


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

# Toggle fullscreen on current focused window
bindsym Control+$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



# ===========================================================================
# 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) &"

# Updates manager
bindsym $mod+u exec "pamac-manager --updates"

# Software manager
bindsym $mod+s exec pamac-manager

# 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'"


# ============================================================================
# Force some launch programs open in floating mode
# ============================================================================

# 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 [window_type="dialog"] floating enbale border pixel 1
# for_window [title="google-chrome"] floating enable
# 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
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


# ================================================================================
# 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
    }
}


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

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
```

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

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

Keybinding	Description
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. 
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

Keybinding	Description
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.
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:

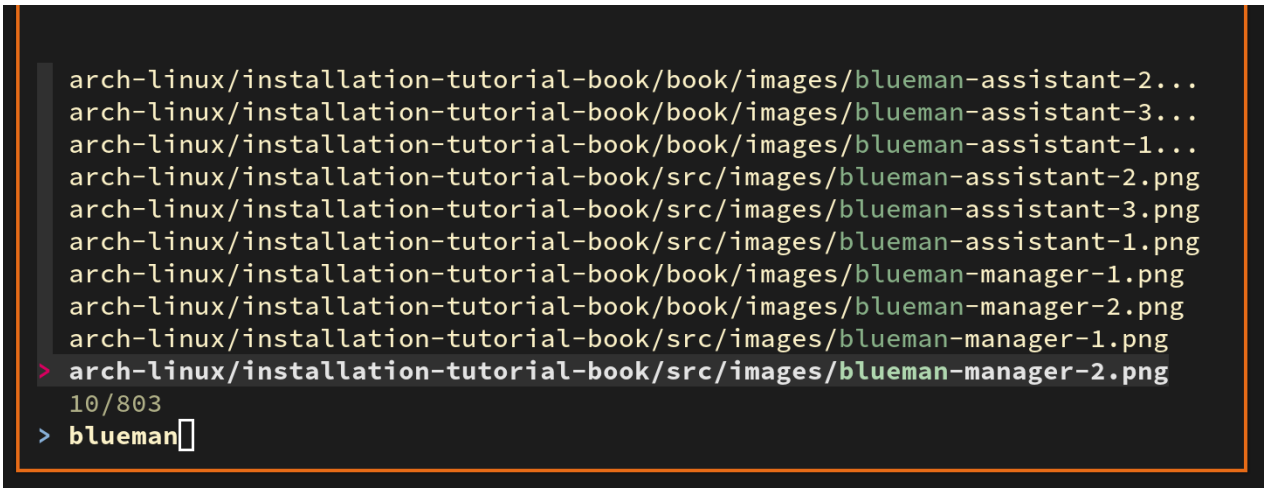
- Add `fuzzy finder` to `ranger`
  - First you need to install `fuzzy finder` by running `sudo pacman -S fzf`
  - Add `~/.config/ranger/commands.py` with the following settings

```
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)
```

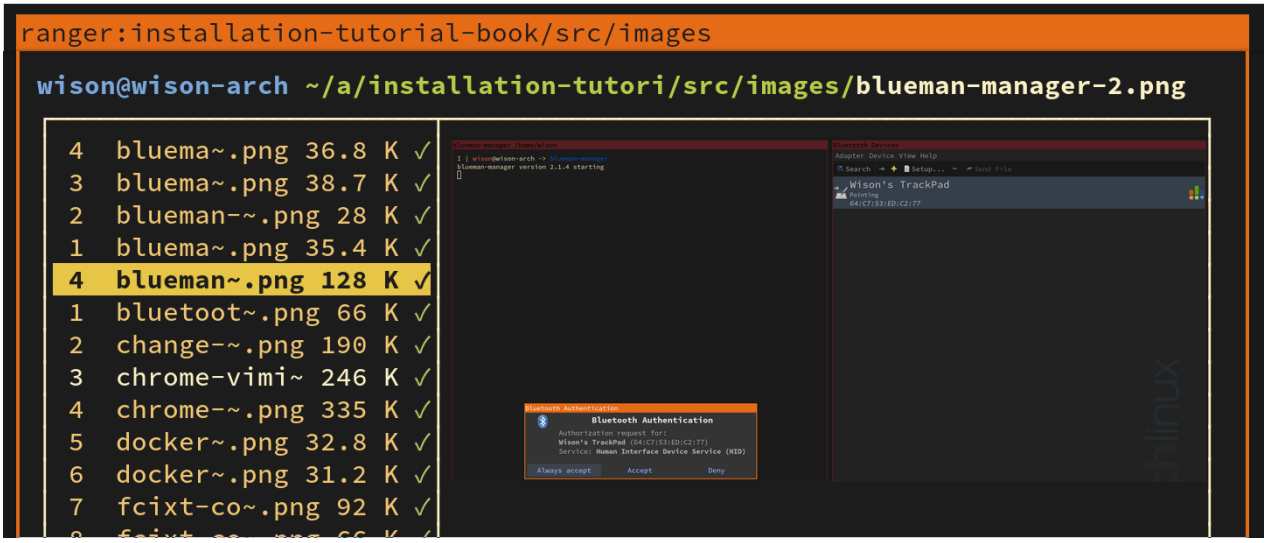
- Add keybinding to `~/.config/ranger/rc.conf`

```
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:

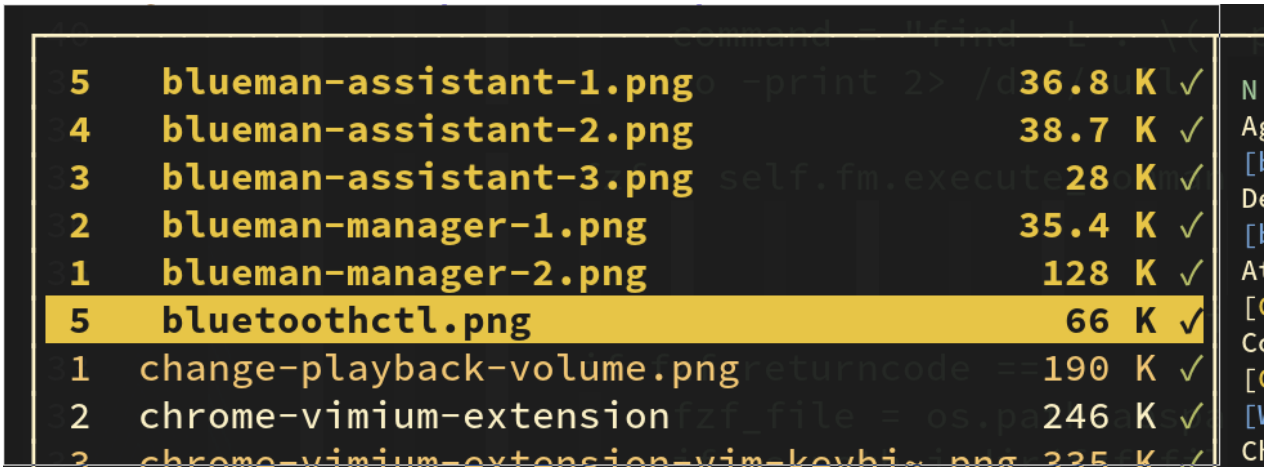


- Fast rename multiple selected files

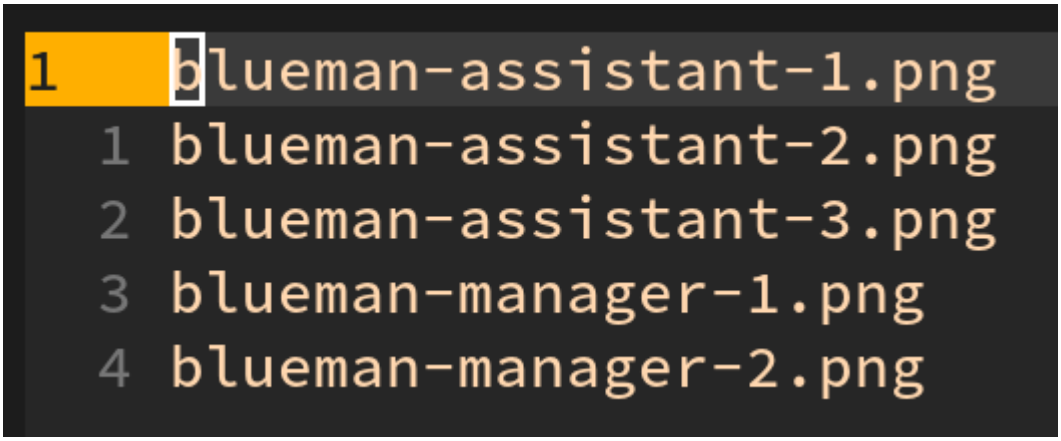
`vim ~/.config/ranger/rc.conf`, and then replace the below keybinding to overwrite the default one:

```
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.



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:

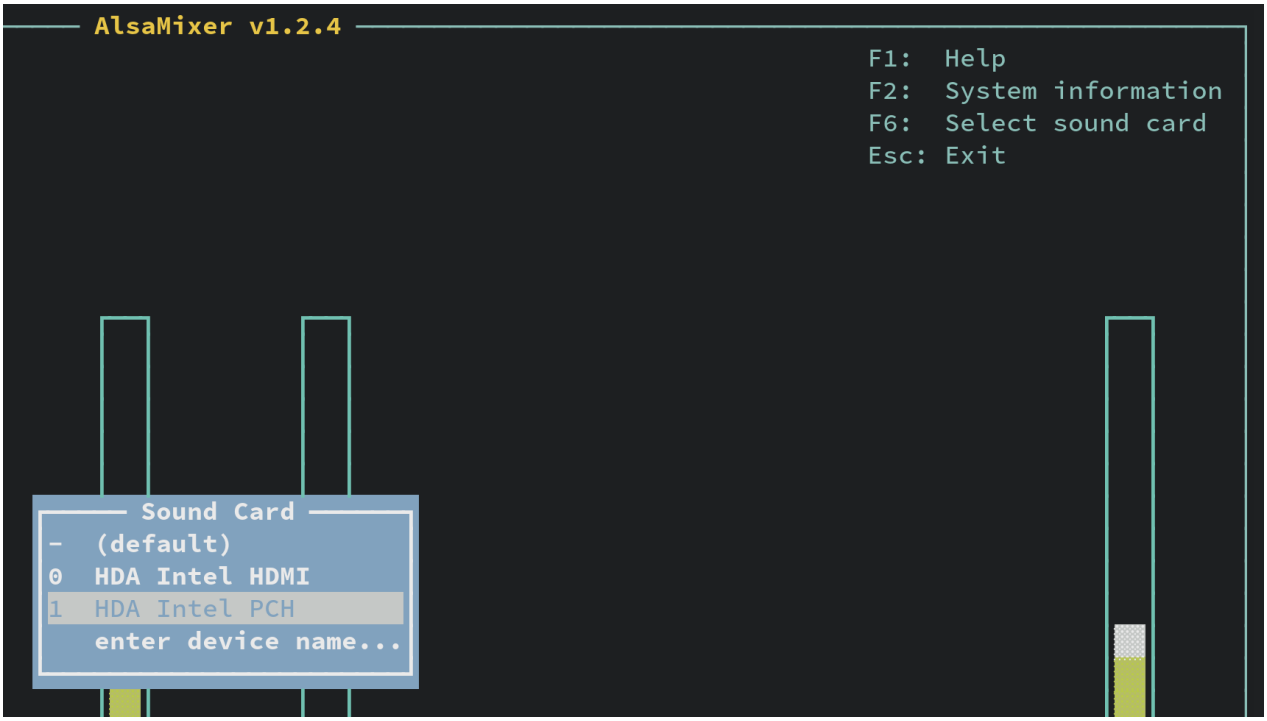
```
# Check the OpenGL version
glxinfo | grep "OpenGL version"""

# Test the OpenGL render framerate
glxgears
```

Fix sound issue

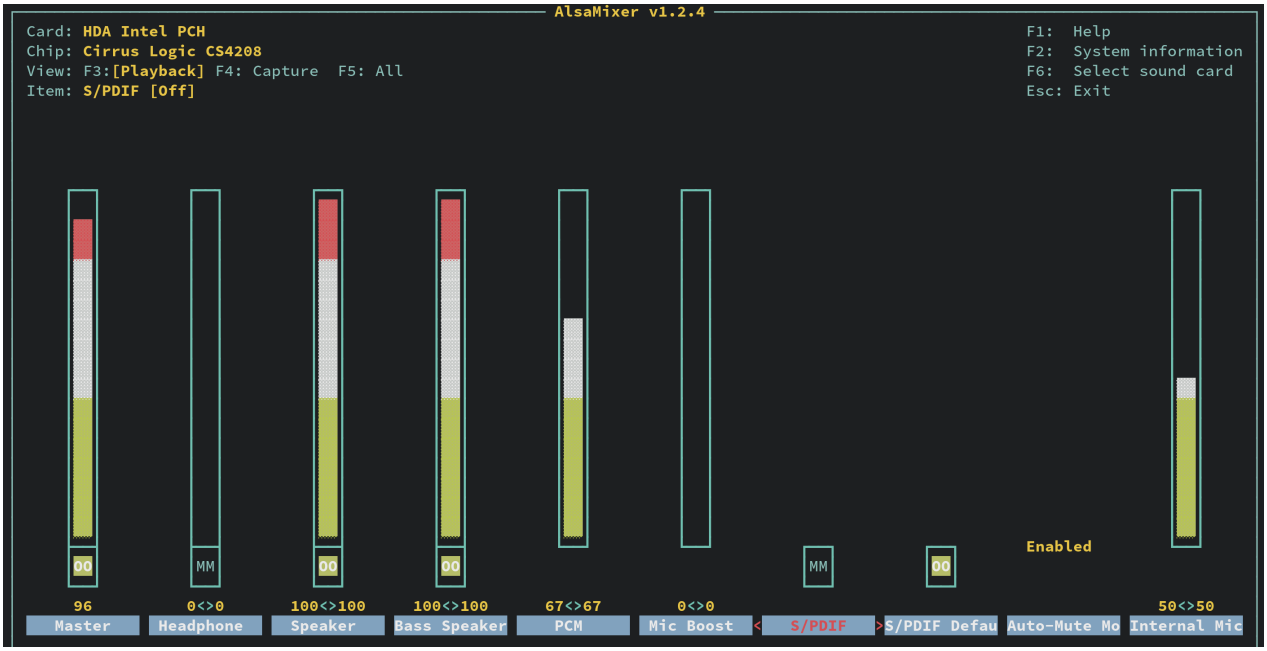
- Install the packages below:

```
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:

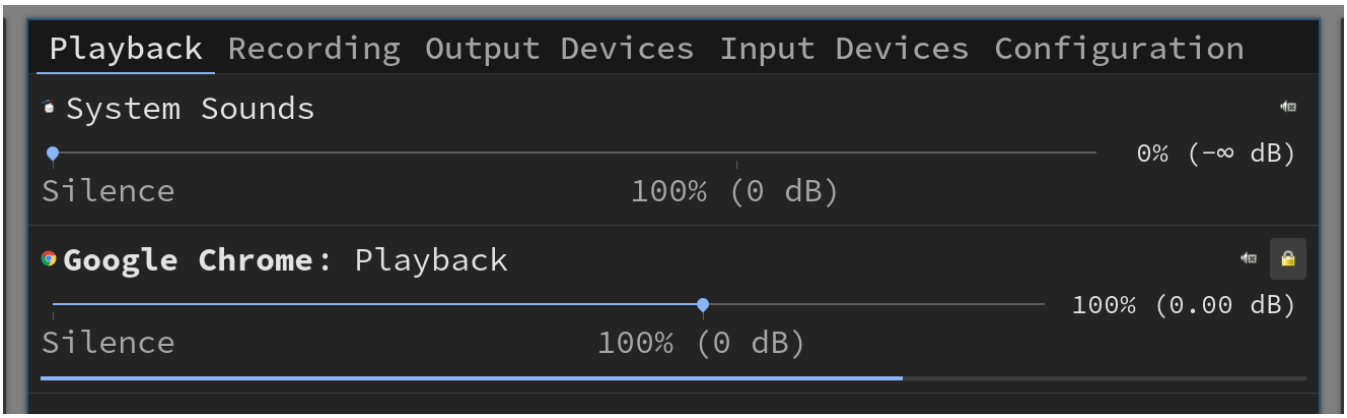
```
# ================================================================================
# Audio and volume control
# ================================================================================
# Use pactl to adjust volume in PulseAudio.
set $refresh_i3status killall -SIGUSR1 i3status
bindsym XF86AudioRaiseVolume 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:



Support chinese

- Install chinese input method

```
# `fcitx-table-extra` includes `WuBi` and `WuBiPinYin`
sudo pacman -S fcitx fcitx-im fcitx-configtool fcitx-table-extra

# SouGou PinYin (it inclues WuBi)
yay -S fcitx-sogoupinyin
```

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

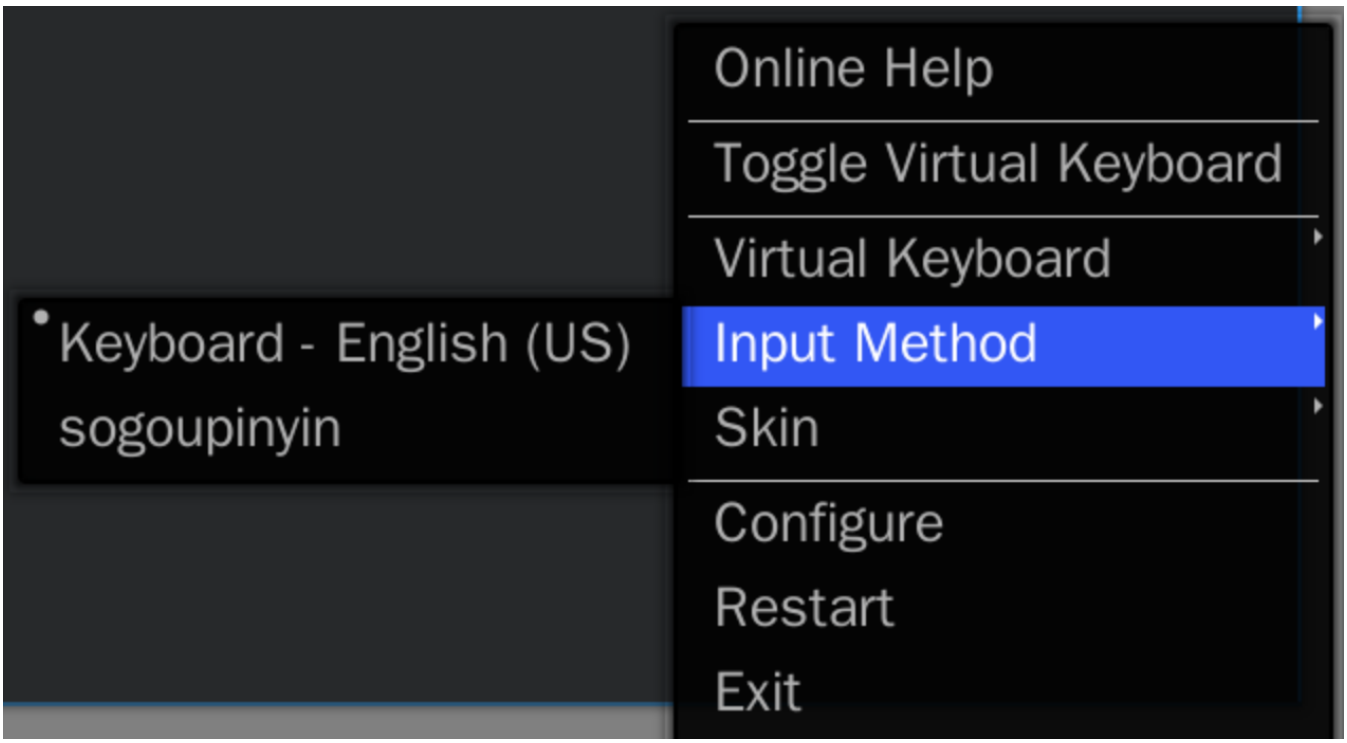
```
yay -S wqy-bitmapfont wqy-microhei \
    wqy-microhei-lite \
    wqy-zenhei \
    adobe-source-han-mono-cn-fonts \
    adobe-source-han-sans-cn-fonts \
    adobe-source-han-serif-cn-fonts
```

  Your browser needs to restart to take effect.

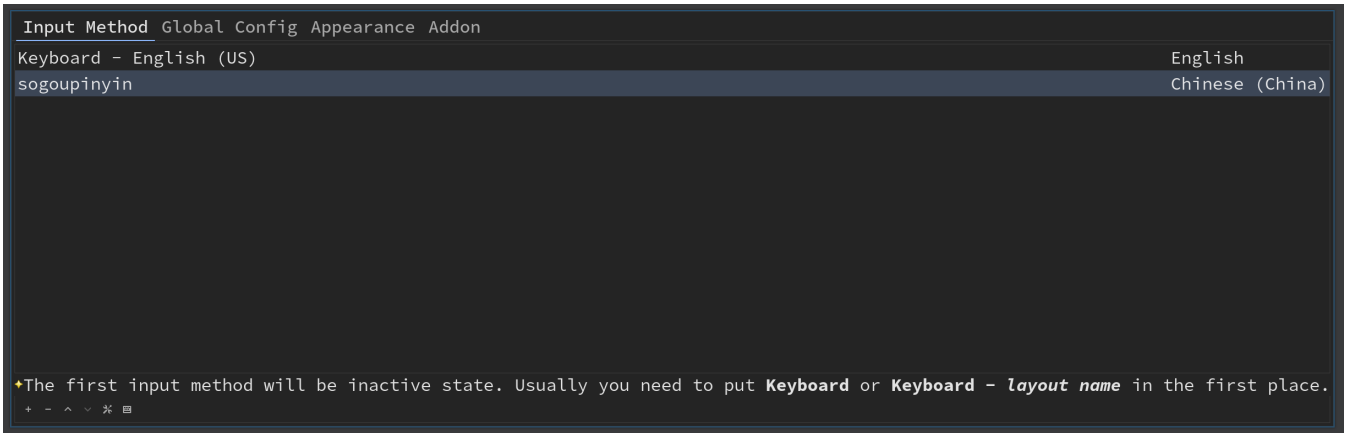
- The default shortcuts
  - `Ctrl+Space`: Trigger input method. If you only install single input method (`fcitx-sogoupinyin`), then it works like toggle chinese input method.
  - `Shirt+Ctrl+f`: Switch between `Traditional Chinese` and `Simplified Chinese`.

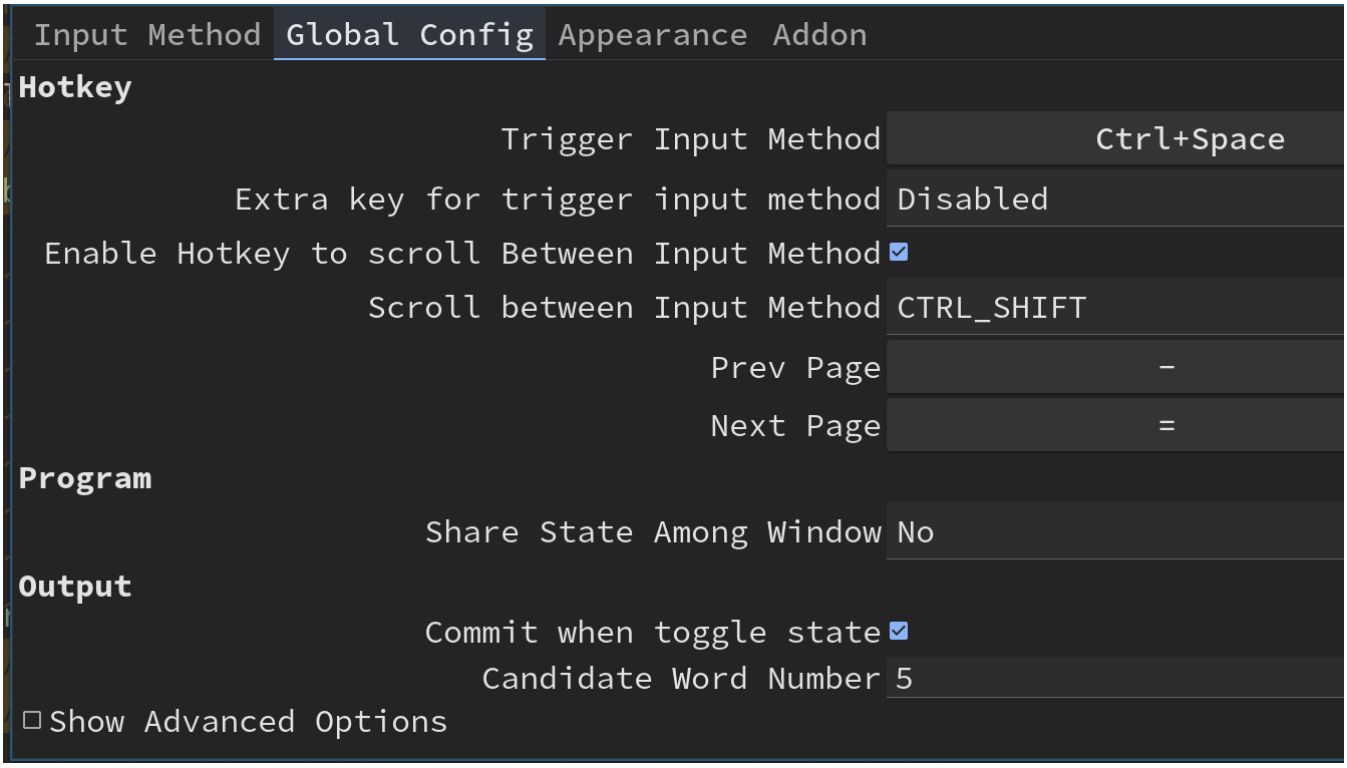
- About the configuration

  After rebooting, 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:





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:

```
:: Running post-transaction hooks...
(1/4) Creating system user accounts...
Creating group docker with gid 972.
```

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'
```

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

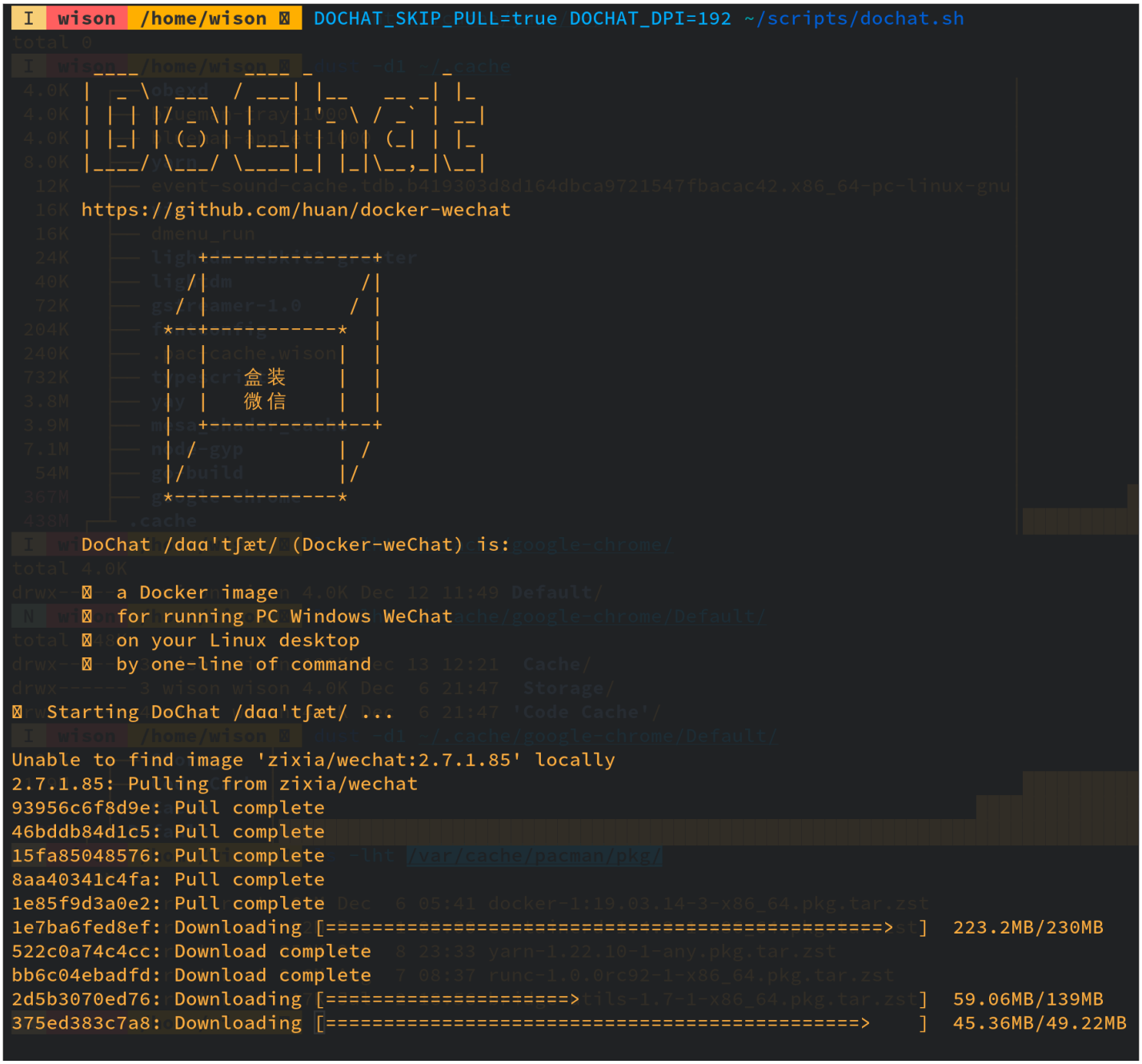
For example in `fish` shell

```
# "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"
```

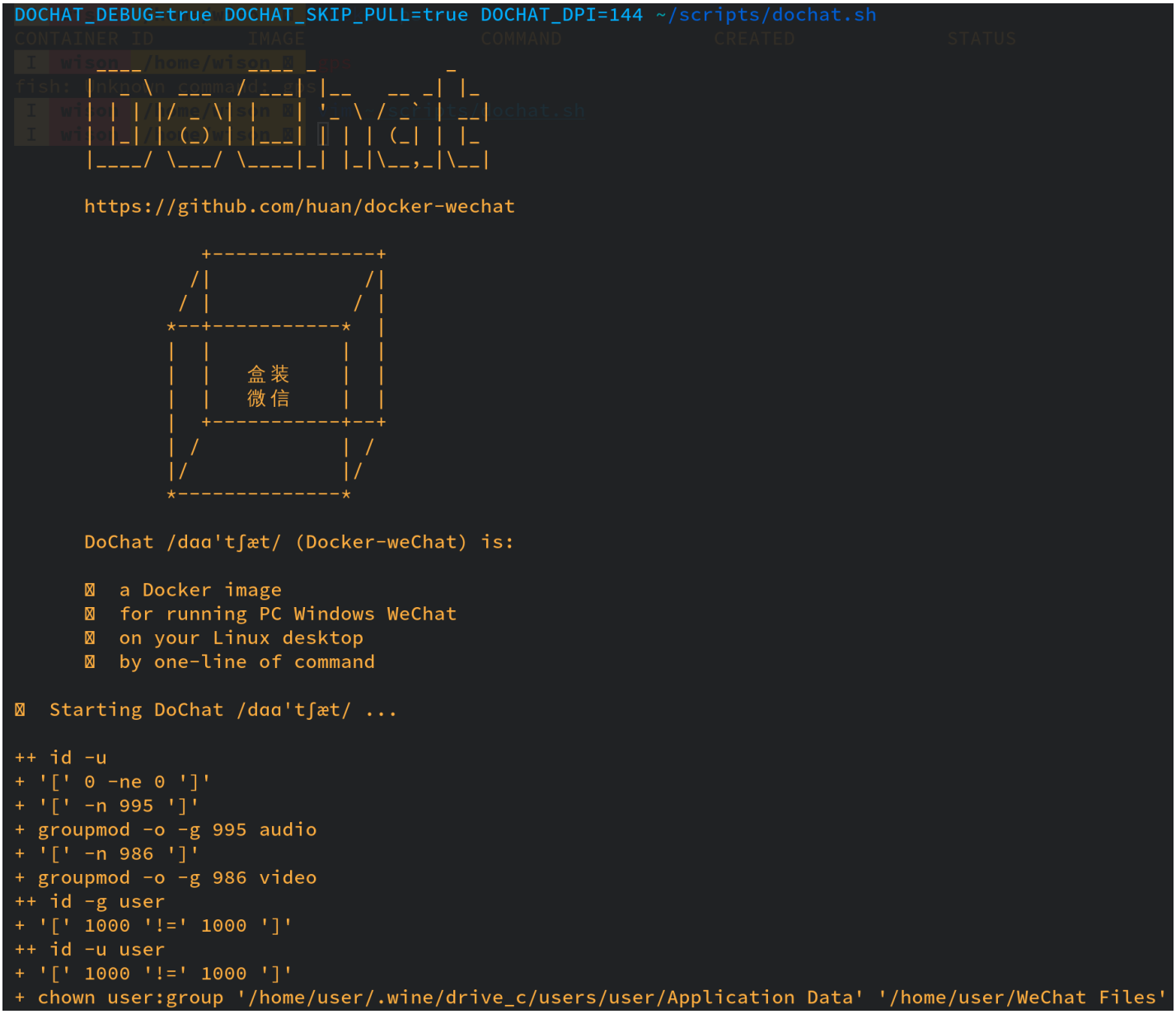
- Run it

```
# 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:



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`.

```
 N [11:17:54] wison@Wisons-iMac /Users/wison
> vim ~/temp/backup-arch-linux.sh
 N [11:18:29] wison@Wisons-iMac /Users/wison
> ~/temp/backup-arch-linux.sh
Backup started at: 2020-12-17_11-18-31
Backup file name: arch-linux-2020-12-17_11-18-31.img.gz
Password:
3666+0 records in
3666+0 records out
30752636928 bytes transferred in 2862.147033 secs (10744604 bytes/sec)
Backup done at 2020-12-17_12-06-16
 I [12:06:16] wison@Wisons-iMac /Users/wison
```

```
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')"
```

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

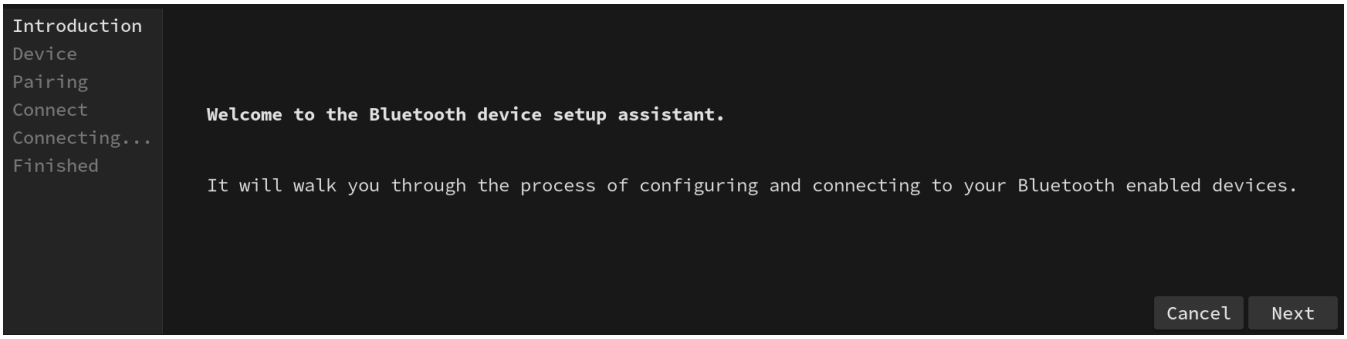
  First, turn off your bluetooth device which want to connect to. Run `bluetoothctl`, then follow the steps below to connect:

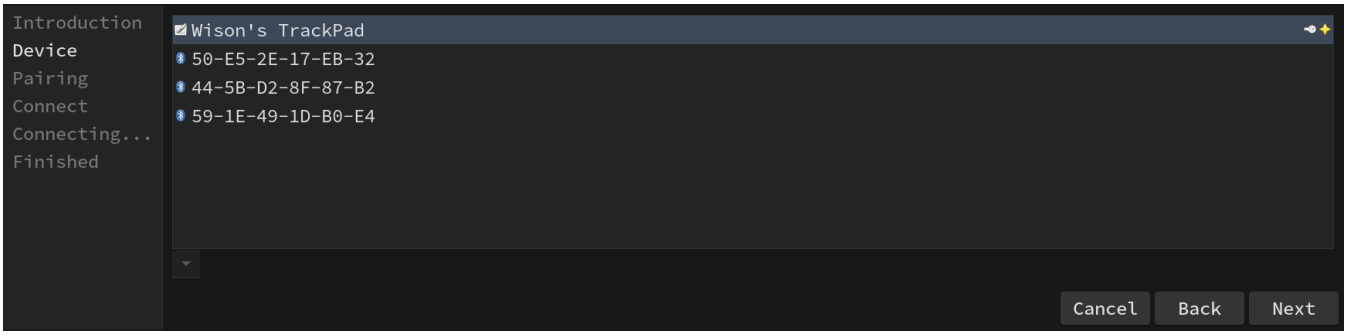
```
# Make sure turn on the bluetooth
power on

# 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: