MoleArch

Meelis Utt

The purpose of this documentation:

- have simplified installation documentation for ArchLinux, so that I can reinstall Arch simply on any machine;
- have documentation of how and what I set up for my system;
- have documentation about the programs I use (not complete documentation, only functionalities that I use).

1. ArchLinux installation

After installation, the machine has ArchLinux with already some MoleArch flavor. There are no users, no GUI etc. Only root user, a command-line, and the following programs: grub, man, sudo, vim, zsh.

1.1. Setup

I'll add how to diskdump an ISO to a USB from Linux.

```
$ dd if=<path to iso> of=/dev/sdX status="progress"
```

where sdX is the usb device (can be checked with lsblk)

Before starting the installation, check if one has UEFI mode enabled on UEFI board. If yes, then it's recommended to follow the official installation guide at https://wiki.archlinux.org/index.php/Installation_guide. Else follow this guide.

If the following command gives error, then the machine does not have UEFI mode enabled.

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

Set up wifi (if one is using ethernet, then this is not necessary).

```
# wifi-menu
```

Check connection with 'ping <website>'.

(optional) If one wants to change keyboard settings, then check the available keymaps using

```
# ls /usr/share/kbd/keymaps/**/*.map.tar.
```

Then use

```
# loadkeys <keymap>
```

For example, loadkeys et.

Ensure system clock is accurate.

```
# timedatectl set-ntp true
```

1.2. Partition, format and mount disk

In this documentation it is assumed, that sda is partitioned. If one wants to partition other disk, then use that disk instead of sda

To check available disks use the following command.

```
# fdisk -l
```

To enter into the mode, where one can partition the disks, then use the command

```
# fdisk /dev/sda
```

I got it to work normally (without grub error), if I only did 1 partition. The following commands are applicable in fdisk.

- d (repeat until all the partitions are deleted; delete partitions)
- n (then p <Enter> <Enter> <Enter>; make new partiton as primary and partiton takes up all the space)
- w (write the partition)
- exit

Format the partition.

```
# mkfs.ext4 /dev/sda1
```

Mount the file system on root partition.

```
# mount /dev/sda1 /mnt
```

If other partitions where made, then make corresponding directories (eg /boot and /home) and mount the corresponding partitions there. Also, if swap was made, then use mkswap and swapon commands on correct mounting point.

1.3. Install and setup essential and wanted packages; generate necessary files

First three are must.

```
# pacstrap /mnt base linux linux-firmware vi vim-minimal man-db man-pages texinfo sudo grub
```

Generate fstab (check /mnt/etc/fstab for errors).

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

Change into new system.

```
# arch-chroot /mnt
```

Set timezone.

```
# ln -sf /usr/share/zoneinfo/<Region>/<City> /etc/localtime
```

To generate /etc/adjtime, run

```
# hwclock --systohc
```

Set locales. Uncomment the necessary locales in /etc/locale.gen (for eg. en_US.UTF-8; ee_ET.UTF-8).

```
# vim /etc/locale.gen
```

Generate locales.

```
# locale-gen
```

Create locale.conf and set LANG variable (and keyboard layout, if wanted)

```
# vim /etc/locale.conf
    LANG=en_US.UTF-8
(# vim /etc/vconsole.conf)
    (KEYMAP=<keymap_value>; eg us, ee)
```

1.4. Network configuration, grub and root password

Create files /etc/hostname and /etc/hosts.

```
# vim /etc/hostname
    MoleArch
# vim /etc/hosts
        127.0.0.1   localhost
        ::1         localhost
        127.0.0.1   MoleArch.localdomain MoleArch
```

Install network manager and iputils.

```
# pacman -S iputils networkmanager
# systemctl enable NetworkManager
```

Set root password.

```
# passwd
```

Set up grub. We already installed it, but if one did not install it, then install it.

```
# (pacman -S grub)
# grub-install --target=i386-pc /dev/sda
```

Make grub config file

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

Exit the chroot environment (optionally unmount) and reboot.

```
# exit
# (umount -R /mnt)
# reboot
```

2. MoleArch setup

2.1. Wifi setup

Before I move on to adding a new user, I need the Internet access. The reason for that is that I want MoleArch to use zsh right away. One could install it later aswell, but I think it is easier to set up basic zsh for new user this way.

If install was done over wifi, then one can connect to that wifi (if available). To check which connections exists, one can run the following command (here it's assumed, that NetworkManager is used).

```
# nmcli connection
```

If found the connection, then we can run the following command

```
# nmcli connection up uuid <UUID>
```

where UUID could be obtained from the nmcli connection command. One could type the UUID in manually, however we can use the following command to do that for us.

```
# nmcli connection | grep <SSID> | awk '{print $2}' | xargs -I {} nmcli connection up uuid '{}'
```

Add -a flag, to be asked a password. This is necessary when there is no connection file in the directory /etc/NetworkManager/system-connections (need to be the root user). The mentioned flag can be used in the commands we are about to mention aswell.

If one wants to connect to undefined wifi network, then to check available networks one can use the following command.

```
# nmcli device wifi list
```

To connect, one can use the following command.

```
# nmcli device wifi connect <SSID> password <password>
```

If one does not want to type password, so that everyone can see, then use the -a flag.

```
# nmcli -a device wifi connect <SSID>
```

2.2. Add new user

As mentioned at the beginning, I want the new user to have zsh. Let's install zsh.

```
# pacman -S zsh
```

Add new user to the group wheel (usual practice), with default zsh as the default shell.

```
# useradd -s /bin/zsh -m -g wheel <new user>
```

Add password to the new user.

```
# passwd <new user>
```

Now we need to uncomment the sudo capabilities of the wheel group in the /etc/sudoers file.

```
# vim /etc/sudoers
```

If one does not want to type in password after logging in, then uncomment (MoleArch choice):

```
%wheel ALL=(ALL) NOPASSWD: ALL
```

If one wants to have to type in password every time, then uncomment:

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

Now it is reasonable to log out of root and login in as the new user. If there is further interest in modifing the user in any way, then check https://wiki.archlinux.org/index.php/Users_and_groups password <password>.

2.3. ZSH (and dash)

Since I set zsh as the default shell for the user, then when logging into the new user for the first time, zsh config function is prompted. Follow that. Most settings were set on and command history is saved into ~/.config/zsh/.histfile

After zsh basic configuration, let's log into the new user.

(At first dash was intended to be used instead of bash. However, for the time being, MoleArch will use bash. Still, the way to make system use dash instead of bash (and vise versa) is still left here.

To check what is used for running scripts run the following command.

```
$ ls -l /bin/sh
```

Use dash instead of bash.

```
$ ln -sf dash /bin/sh
```

)

2.4. Setting up and configuring

At the moment, we have installed vim-minimal. This probably does not support copy/paste from/to system clipboard. For this we will install gvim (graphical vim; not going to use the graphical vim, but it is compiled with clipboard functionality).

```
$ sudo pacman -S gvim
```

!!!!!!!! NEED TO SET UP SOME NEOVIM CONFIGS, SOME THINGS BROKE (py self funs eg) and SOME IS ANNOYING (file~ eg). CURRENTLY zsh_aliases the vim = nvim has been turned off and I'm using gvim until I have time to tinker and config nvim.

I’m trying Neovim out. Install neovim.

$ sudo pacman -S neovim

Neovim can be call out with command

$ nvim

and uses init.vim as a config. However, we can use the current vimrc. To check how, in nvim type

:help nvim-from-vim

Let’s install some plugins for zsh. The .zshrc file we end up using is in the git repository. But more on that soon.

$ sudo pacman -S zsh-syntax-highlighting

Install and setup git so we can access the existing MoleArch dotfiles, scripts, documentation etc (git pull to the $HOME folder). Also, git allows to download from AUR, use git etc.

$ sudo pacman -S git

Config the git global user.

$ git config --global user.name "<username>"

$ git config --global user.email "<email>"

Since there are no folders in the home directory, let’s make the basic folders automatically (can be done manually). First install the package.

$ sudo pacman -S xdg-user-dirs

Then run the following command.

$ xdg-user-dirs-update

Additionally, let’s add following hidden folders.

$ mkdir ~/.scripts

$ mkdir ~/.suckless

Also, let’s make some additional directories in Documents.

$ cd ~/Documents

$ mkdir Testing

$ mkdir Projects

$ Rolling_files

Let’s install a display server. We are going to install Xorg.

$ sudo pacman -S xorg-server xorg-xinit

Install fonts.

$ sudo pacman -S noto-fonts

Let’s install a driver. Check the driver with the following command.

$ lspci | grep -e VGA -e 3D

Install the driver.

$ sudo pacman -S xf86-video-<driver>

Since we did not install base-devel previously, but we want to be able to install from the AUR, then let’s install base-devel (--needed flag only installs the packages that are missing).

$ sudo pacman -S --needed base-devel

For sound, let’s install alsa-utils.

```
$ sudo pacman -S alsa-utils
```

To get the CLI sound mixer, use the following command.

```
$ alsamixer
```

Install acpid and acpi, which helps to handle different events.

```
$ sudo pacman -S acpi acpid
```

Enable systemctl service.

```
$ systemctl enable acpid.service
```

Now let's set up a configuration to handle screen brightness and volume events (if necessary). For this run the command

```
$ acpi_listen
```

and push the button (compinations), to get the acpi button identity. For now we check the following buttons: volume up, volume down, mute, brightness up, brightness down. Let's enable volume control. For that we do the following.

```
$ sudo vim /etc/acpi/events/vol-u
        event=button/volumeup <acpi_identity>
        action=amixer set Master 1+
$ sudo vim /etc/acpi/events/vol-d
        event=button/volumedown <acpi_identity>
        action=amixer set Master 1-
$ sudo vim /etc/acpi/events/vol-m
        event=button/mute <acpi_identity>
        action=amixer set Master toggle
```

New let's enable brightness control. For that we will use a script.

```
$ sudo mkdir /etc/acpi/handler
$ sudo vim /etc/acpi/handler/bl
        #!/bin/sh
        bl_dev=/sys/class/backlight/<correct_folder_to_working_backlight>
        step=5
        case $1 in
            -) echo $(($(< $bl_dev/brightness) - $step)) >$bl_dev/brightness;;
            +) echo $(($(< $bl_dev/brightness) + $step)) >$bl_dev/brightness;;
        esac
$ sudo chmod +x /etc/acpi/handler/bl
$ sudo vim /etc/acpi/events/bl_u
        event=video/brightnessup <acpi_identity>
        action=/etc/acpi/handlers/bl +
$ sudo vim /etc/acpi/events/bl_d
        event=video/brightnessdown <acpi_identity>
        action=/etc/acpi/handlers/bl -
$ systemctl restart acpid.service
```

Let's add an event for when there is a screenlock function key on the computer.

```
$ sudo vim /etc/acpi/events/scr_lck
        event=button/screenlock <acpi_identity>
```

action=slock

Now we need to restart acpid.service. Also, in the future I might add other event handlers aswell.

Install utility to manage/have a simple firewall. Also, enable the service with systemctl (TODO: study the utility more).

$ sudo pacman -S nftables

$ systemctl enable nftables.service

Install utility to be able to see the what are the keyboard and mouse values sent to xorg.

sudo pacman -S xorg-xev

The program can be run with the following command.

$ xev

Before installing suckless utilities, we need to install make, so we could compile the programs. (The package fontconfig was also needed when trying to make st, but I have to test, if it's needed when dmenu is installed first).

$ sudo pacman -S make (fontconfig)

For terminal we are going to install simple terminal (st) from suckless. Before installing st, let's install dmenu, since it seems it solves some problems further down the road when making st.

$ sudo pacman -S dmenu

Since one might not be in a graphical environment yet, then we go with the git version.

$ cd ~/.suckless

$ git clone https://git.suckless.org/st

$ cd st

$ sudo make clean install

Let's install some patches. Under suckless.org->st->patches, select the following patches, download them and move them to directory ~/.suckless/st.

- scrollback- allows to scroll back in st.

$ cd ~/suckless/st

$ patch -p<nr of patch> < <patch>

- Add the first (scrollback) patch. For this one needs to add two lines manually to config.h. For example:

  { XK_ANY_MOD, XK_Page_Up, kscrollup, {.i = -1} },

  { XK_ANY_MOD, XK_Page_Down, kscrolldown, {.i = -1} },

  In this case, we allow scrolling in terminal with just PgUp and PgDn keys. The corresponding keynames might be different in different machines and can/should be changed for personal preference.

  Another example what to do:

  { ShiftMask, XK_Page_Up, kscrollup, {.i = -1} },

  { ShiftMask, XK_Page_Down, kscrolldown, {.i = -1} },

  { XK_ANY_MOD, XK_Prior, kscrollup, {.i = 1} },

  { XK_ANY_MOD, XK_Next, kscrolldown, {.i = 1} },

  One can add analogically scrolling with a mouse. I haven't figured it out yet how to scroll with a trackpoint.

$ sudo make clean install

The following suckless utilities can be installed later.

Install coping CLI tool sselp or xclip.

```
$ sudo pacman -S xclip
$ cd ~/.suckless
$ wget https://dl.suckless.org/tools/sselp-0.2.tar.gz
$ tar -xvzf <tar.gz name>
$ cd sselp
$ sudo make clean install
```

Install suckless utility for locking screen.

```
$ cd ~/.suckless
$ git clone https://git.suckless.org/slock
```

Now let's change the values for user and group. The group will be 'wheel' (if not chosen otherwise) and the user will be the current user. Then run the commands.

```
$ sudo make clean install
```

If wanted, change the color of init, input and failed. One possiblity is to keep the default colors and git pull already configured slock. Let's create a service that locks screen when the lid is closed.

```
$ sudo vim /etc/systemd/system/slock@.service
  [Unit]
  Description=Lock X session using slock for user %i
  Before=sleep.target

  [Service]
  User=<user>
  Environment=DISPLAY=:0
  ExecStartPre=/usr/bin/xset dpms force suspend
  ExecStart=/usr/bin/slock

  [Install]
  WantedBy=sleep.target
```

If this does not work, then locate the slock machine code and change the ExecStart value accordingly. Example

- ExecStart=/usr/locale/bin/slock

One can use the commend

```
$ whereis slock
```

to locate the correct path.

It is recommended to make file /usr/share/X11/xorg.conf.d/xorg.conf, to block tty access when in an X and prevent a user from killing when it is running.

```
$ sudo vim /usr/share/X11/xorg.conf.d/xorg.conf
    Section "ServerFlags"
        Option "DontVTSwitch" "True"
        Option "DontZap"      "True"
    EndSection
```

Now lets enable the slock service (if enable does not work, do a restart aswell).

$ systemctl enable slock@<user>.service

Install suckless browser and enable it to have tabbed window. On suckless's page, it is recommended to build ones own WebKitGTK. First go to https://webkitgtk.org/releases/ and take the latest stable version. Let's download it to the Downloads directory at the moment.

$ cd ~/Downloads

$ wget <url of the tar>

$ tar -xf webkitgtk-<version>.tar.xz

$ cd webkitgtk-<version>

(resolve all deficiencies, meaning install all necessary packages)

$ cmake -DPORT=GTK -DCMAKE_BUILD_TYPE=RelWithDebInfo -GNinja

$ ninja

$ sudo ninja install

$ cd ~/.suckless

$ git clone https://git.suckless.org/surf

$ cd surf

$ sudo pacman -S gcr (webkit2gtk) (needed for make install; gcr is a crypto tool I think; webkit2gtk is the engine)

$ sudo make clean install

$ git clone https://git.suckless.org/tabbed

$ cd tabbed

$ sudo make clean install

$ sudo pacman -S gstreamer gst-libav gst-plugins-good (for video support in surf; however, if not installed, eg yt runs faster for me, I anyway youtube-dl them and use mpv)

the last line for video support, did not solve the problem with video audio.

To make surf bit more functional, then we will apply some patches. Under suckless.org->surf->patches, select the following patches, download them and move them to directory ~/.suckless/surf.

- modal - allows to use vim keys without Ctrl.
- playexternal plays video in external program (mpv)
- (searchengine - allows to set shortcuts to search engines)
- (web search - gives duckduckgo search bar functionality)

$ cd ~/suckless/surf

$ patch -p<nr of patch> < <patch>

$ sudo make clean install

If there are any errors when patching, then manually include the lines from .rej file to the corresponding file. Now let's add some scripts from suckless.org->surf->files. Let's copy the following script files to ~/.surf/scripts.js.

- link hints
- easy links

TODO: Install other suckless utilities.

The parts inside parenthesis are important, if there is no access to MoleArch git repository (is effective unitl the end of the document).

(Let’s copy the default config files to / directory.

$ cp /etc/X11/xinit/xinitrc ~/.xinitrc

$ cp /etc/X11/xinit/xserverrc ~/.xserverrc

Change the ~/.serverrc file as following.

exec /usr/bin/Xorg -nolisten tcp "$@" vt$XDG_VTNR

To make sure that "startx" is autostarted at login, add the following to ~/.zprofile.

```
if systemctl -q is-active graphical.target && [[ ! $DISPLAY && $XDG_VTNR -eq 1 ]]; then
    exec startx
fi
```

)

Now let’s install chosen tiling window manager, which in this case is bspwm and sxhkd manages the hotkeys.

$ sudo pacman -S bspwm sxhkd

(Now let’s add the following lines to ~/.zprofile.

```
XDG_CONFIG_HOME="$HOME/.config"
export XDG_CONFIG_HOME
```

Make directories.

$ mkdir ~/.config/bspwm

$ mkdir ~/.config/sxhkd

Copy the bspwmrc and sxhkdrc to / directory.

$ cp /usr/share/doc/bspwm/examples/bspwmrc ~/.config/bspwm/bspwmrc

$ cp /usr/share/doc/bspwm/examples/sxhkdrc ~/.config/sxhkd/sxhkdrc

Now add the following line to .xinitrc (take note to add it before other exec’s).

exec bspwm

)

Now let’s install status bar. We are currently installing lemonbar (from AUR).

$ mkdir .lemonbar

$ cd lemonbar

$ git clone https://aur.archlinux.org/lemonbar-git.git

$ cd lemonbar-git

$ makepkg -sirc

(Now let’s add a line to bspwmrc, that runs lemonbar, when bspwm is ran (assumes, that panel is a script, that does all necessary to make a bar; furthermore, it seems my current implementation is different from what I found on the Internet, but it works for now).

$ vim ~/.config/bspwm/bspwmrc

```
panel &
```

)

Now let’s install all the other wanted programs. The config files are going to end up in the MoleArch git repository.

$ sudo pacman -S htop fzf keepassxc youtube-dl zathura zathura-pdf-mupdf mpv redshift sxiv scrot (nitrogen) firefox transmission-cli wget newsboat cronie texlive-most texlive-lang biber pandoc xorg-xrandr

Lastly, let’s pull MoleArch git files to $HOME directory.

$ cd

$ git pull

Some more useful programs.

- ntfs-3g -- ables easily mount ntfs partitions.

  $ sudo mount -t ntfs-3g /dev/sdX <mount point>

Let's pull the MoleArch git repository to the / folder. If MoleArch is not in the condition yet, I would suggest not pulling to root directory and instead to other directory. Then move/use the good bits of code, configs, scripts etc. Also, the following commands show briefly how to do git init.

$ git init

$ git remote add origin https://meelis_utt@bitbucket.org/meelis_utt/molearch.git

$ git pull --set-upstream origin master

3. Some last setups

Let's add a cronjob to update newsboat every <x> minutes.

$ crontab -e

*/<x> * * * * /usr/bin/newsboat -x reload

- If this is the first cronjob and cronie service is not enabled in systemctl yet, then do the following.

$ (sudo) systemctl enable cronie.service

The following part is setting up a VGA1 monitor.

$ xrandr --output LVDS1 --auto --output VGA1 --auto --left-of LVDS1

Let's download Arch wiki docs into the local machine.

$ sudo pacman -S arch-wiki-docs

The documentation are located in /usr/share/doc/arch-wiki/.

4. For programming

Now we install programs necessary for programming tasks.

Install R (package tk is necessary to be able to install other packages in R)

$ sudo pacman -S r tk

In R run the following commands to install some most used libraries.

- install.packages("tidyverse") - dplyr, plyr, etc
- install.packages("rmarkdow") - to be able to run and compile Rmd files.

Let's make a .Rprofile file in the home directory. In that we specify what we want to run when .R file is opened. For example, we source the rolling files and functions. An example:

$ vim ~/.Rprofile

```
# Load rolling functions and other files
source(<path of rolling file>)

library(dplyr)
```

Install compilators for C/C++/C#. The compilator g++ is probabily already installed. Also install doxygen for automated documentation. The smoother for g++ is gdb and for clang it's lldb.

$ sudo pacman -S clang doxygen

Install python, pip and jupyter notebook.

$ sudo pacman -S python python-pip jupyter-notebook

Let's add the directory ~/Documents/Rolling_files/Python_files (if the last directory does not exist, make it) to python path, we can import the modules in that folder easily into python scripts.

$ cd /usr/lib/python3.8/site-packages

$ sudo vim self_paths.pth

/home/<user>/Documents/Rolling_files/Python_files/

If the directory to site-packages is different (python version 2.x eg), then find correct path to that directory. Now let's install some extra modules.

$ sudo pip install numpy scipy pandas dfply

Short explanation of the packages:

- numpy - vector and math operators.
- scipy -
- pandas - data frames in python
- dfply - simulates dplyr in python (similar/same function name; piping). Only works with data.frames. See example in ~/Documents/Testing/piping_dfply_example.py
- pipeop - can use pipes with other data types aswell. See example in ~/Documents/Testing/piping_dfply_example.py

5. A little help from a friend

5.1. Programs

- gim/vim - Text editor
- git - git
- alsa-utils - Sound
- acpi, acpid - Different event handler
- dmenu - program launcher essentially, but it's very extensible
- st (simple terminal) - terminal emulator
- sselp (&| xclip) - commandline clipboard tool (eg $(sselp) runs the primary selection in terminal)
- slock - simple screenlock
- tabbed - enables tabbed surf browser or terminal etc
- surf - simple web browser
- firefox - web browser
- bspwm - binart space partition windom manager (tiling window manager)
- sxhkd - simple x hotkey daemon (hotkey daemon)
- lemonbar - status bar
- zathura. zathura-pdf-mupdf - extensible document viers, pdf viewer
- mpv - media player (music, movies)
- redshift - night light essentially
- sxiv - simple x image viewer
- youtube-dl - downloads content from youtube
- fzf - fuzzy finder (searching program)
- transmission-cli - commandline torrent client
- keepassxc - password manager

- pass - terminal password manager (not set up yet)
- wget - get's content from the web
- htop - shows all active processes
- nnn - terminal file manager
- neomutt - terminal email client (not set up yet)
- newsboat - RSS reader
- w3m - terminal browser
- texlive-most biber pandoc - LaTex packages and TeX for Rmd
- xrandr - monitor setup

5.2. Commands

- pacman -<flag(s)> <program>
  S -- install a program
  Q -- check if installed

5.3. TODO

nm-applet (wifi settings)/stalonetray
neomutt
config zathurarc
firewall? (nftables)
config bspwmrc
manage dotfiles, clean .zshrc/.zprofile etc
Backup (setting up). Do a backup script
connect android, usb, smart card reader - scripts
useful commands doc (eg pacman -Q, pacman -Syu, grep, sed etc)
power management (hibernate, sleep)
Camera, mic management
block ad & other sites
git password asking.
Virtual machine
st config/patches
password manager pass
screenshots (scrot(/imagemagick?))
esc to caps lock and caps lock to menu/alt_gr orsth
Config R for vim (.Renviron, showing plots, installing packages in script (.Renviron point again))
config C++ for vim
more cool VIM PLUGINS, configs etc
lemon bar config even more optimal (bspc subscribe (Brodie vid)) + FIFO system instead of while
reinstall and path surf. Config surf. tabbed+surf
CLEAN DOCUMENTATION and add explanations to commands and programs.

CONFIG and add urls to newsboat

Explore w3m - terminal web browser

cron vs systemd .timer

Clean vim config (also, it seems that compiling and opening to pdf might be able to compress into single command with multiple file types)

config scrot, document scrot

map lockscreen to LCKSCRN button. document process.

rsync to replace cp and mv or as an alternative ??

automatic mount/umount scripts

SETUP BACKUP AND DO BACKUPS.

xrandr document!!!!! (monitor)

setup shortcuts so they can be usable with monitor and keyboard

autorun monitor scripts (through acpi event??)

able to sleep computer, when monitor attached, with a keybinding

scroll terminal with mouse.

battery (and other) from acpi to /sys/ based (lukes video Using and scripting battery information) like monitors script.

(See last point) explore /sys/ directory.

REPEATING MYSELF, but re-organize different program conf and rc files. look into xorg.conf.

- Xorg :0 -configure (xorg.conf.new in /root/)

for skeleton to copy to

Monitors script run when logged in properly (smth broke)

panel not hardcoded monitors

If no monitors attached, desktops in status bar needs fix.

config nnn and ls

zsh_env -> zshenv, clean other files.

vimrc