

Installing Arch Linux if you were UwUssimo



#arch #linux #archlinux

Connect to the network:

```
ip -c a # to get info about ip
address [warning: lgbt]
iwctl. # if you have to connect over
wifi
device list # to get to know name of
wlan0
station wlan0 get-networks # to get the list of
available routers
station wlan0 connect SSID # to connect to the given
router
```

We will start changing the font and warming up clusters:

```
setfont ter-132n # make the font bigger and better
loadkeys us # set the US english keyboard map
pacman -Syy # refresh the database of package
manager
```

```
# If you love gambling: set country to Uzbekistan, else
Russia
reflector -c Uzbekistan --sort rate --save /etc/
pacmand.d/mirrorlist
```

Get started with partitions:

```
# [Guide]
# Use gdisk for gpt, fdisk mbr. Avoid cfdisk & gfdisk at
any cost!
# GUI & TUI while archlive is true evil!!!
```

Drivers	/dev/sdX1	/dev/sdX2	/dev/sdX3
---------	-----------	-----------	-----------

Size limitations	200/500 MB	2 times RAM if less than 8GB or 8GB	*rest of the disk*
Formatting	mkfs.vfat /dev/sdX1	mkswap /dev/sdX2	mkfs.ext4 /dev/sdX3
Mount Point	/mnt/boot/efi	[swapfile]	/mnt

Installing the essential packages for bootstrap

```
pacstrap /mnt base linux linux-firmware helix {intel-ucode | amd-ucode} git
```

Create the fstab files and chroot

```
genfstab -U /mnt >> /mnt/etc/fstab # generate fstab
arch-chroot /mnt # enter to the machine
```

Setup basic things

Date and locales

```
ln -sf /usr/share/zoneinfo/Asia/Tashkent /etc/localtime
# link the localtime
hwclock --systohc      # synchronise clock to hardware
hx /etc/locale.gen      # uncomment utf-8.us
locale-gen              # generate the locale files
hx /etc/locale.conf     # content: LANG=en_US.UTF-8
hx /etc/vconsole.conf   # content: KEYMAP=us
```

Host files

```
echo "uwu" >> /etc/hostname # give a name to the machine
vim /etc/hosts.              # start editing the
localhost routes
```

[Content => /etc/hosts]

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

[End => /etc/hosts]

Installing all required packages

```
pacman -S grub efibootmgr networkmanager network-
manager-applet wpa_supplicant dialog mtools dosfstools
reflector base-devel linux-headers bluez bluez-utils
cups xdg-utils xdg-user-dirs rsync inetutils dnsutils
nfs-utils gvfs gvfs-smb openssh
```

Choosing graphic driver

```
pacman -S xf86-video-amdgpu      # amd processors
```

```
pacman -S xf86-video-intel      # intel processors
pacman -S nvidia nvidia-utils   # nvidia gahps
pacman -S virtualbox-guest-utils # xf86-video-vmware #
virtualbox
```

Change some parameter on mkinitcpio

```
vim /etc/mkinitcpio.conf # add {nvidia | amdgpu | i915}
to MODULES
mkinitcpio -p linux      # compile for linux kernel
```

Create bootloader

```
grub-install --target=x86_64-efi --efi-directory=/boot/
efi --bootloader-id=OWO # create the bootloader
vim /etc/default/grub    # append
video=1920x1080 to LINUX_DEFAULT # warn about the
display
grub-mkconfig -o /boot/grub/grub.cfg # update the config
```

Enabling all services

```
systemctl enable NetworkManager # turn on network
management
systemctl enable bluetooth      # turn on bluetooth
service
systemctl enable cups           # printer manager gets
on
systemctl enable sshd           # ssh server is getting
on also
```

```
# If there's error, it's totally ok as you don't have
XDISPLAY
# currently running in your session, once reboot, it
gets fixed!
systemctl enable --now reflector.timer # make reflector
refresh by the time
systemctl enable --now fstrim.timer    # make the ssd
get updated by the time
```

Accounts

```
passwd          # enter password for the
root user
useradd -mG wheel uwussimo # create the user
passwd uwussimo  # enter password for the
user
EDITOR=hx visudo # open sudo privileges for
wheel
```

Exit and reboot

```
exit
umount -a
exit
reboot
```

[~] Installing the DE

```
reflector -c [Russia|Uzbekistan] --sort rate --save /
etc/pacman.d/mirrorlist
sudo pacman -S {xorg | wayland} {sddm | gdm} {kde: kde-
applications packagekit-qt5} {gnome: gnome gnome-tweaks}
sudo systemctl enable {gdm | sddm}
```

[~] or installing bspwm

...currently being written (WIP)

Installing the AUR manager and configure

```
sudo pacman -S rustup # Install rust to compile paru
rustup install stable # Get the latest stable version
git clone https://aur.archlinux.org/paru.git .paru
cd .paru && makepkg -si # Compile the dang paru
mkdir -p ~/.config/paru
hx ~/.config/paru/paru.conf
```

[Content => ~/.config/paru/paru.conf]

```
#
# GENERAL OPTIONS
# Turning off review, etc to fuck off bloody warnings
#
[options]
SkipReview
PgpFetch
Devel
Provides
DevelSuffixes = -git -cvs -svn -bzip -darcs -always -hg
-fossil
CleanAfter
[End => ~/.config/paru/paru.conf]
```

Oh!! Yuri on Terminal!!!

```
# Install oh-my-zsh, write Y when asks to set zsh as
default shell...
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/
ohmyzsh/master/tools/install.sh)"

# Let's install all required plugins and things before
```

```
editing .zshrc
sudo pacman -S starship zoxide wget btop exa bat fd
ripgrepprocs gping hyperfine just gitui lolcat tealdeer
xh grex neofetch onefetch
```

[My favourite zsh plugins!!]

Fast Syntax Highlighting

```
git clone https://github.com/zdharma-continuum/fast-
syntax-highlighting.git \
    ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/plugins/fast-
syntax-highlighting
```

Zsh Completions

```
git clone https://github.com/zsh-users/zsh-completions \
    ${ZSH_CUSTOM:-$ZSH:~/.oh-my-zsh}/custom}/plugins/
zsh-completions
```

Zsh Autosuggestions

```
git clone https://github.com/zsh-users/zsh-
autosuggestions \
    ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-
autosuggestions
```

My config files

```
CURRENT_PATH="$(pwd)"
mkdir -p ~/.dots/
hx ~/.dots/alias.config
```

[Content => ~/.dots/alias.config]

User Aliases

```
alias archbitch="neofetch"
alias down="cd ~/Downloads"
alias ..="cd .."
alias ....="cd ../.."
alias kawaii="sudo pacman -Syyu | lolcat"
alias celar="clear"
```

Made with Rust

```
alias top="btop"
alias cat="bat"
alias caaat="bat -p"
alias ls="exa"
alias sl="exa"
alias fak="tealdeer"
alias fok="onefetch"
```

```
# Refresh
alias refresh="source ~/.zshrc"

# Find
alias look="fd"
alias find="fd"

# Grep
alias grep="rg"
alias search="rg"

# PS
alias ps="procs"

# Ping
alias ping="gping"

# Time
alias time="hyperfine"

# Make
alias j="just"
alias make="just"

# Vim
alias vim="hx"
alias vi="hx"

# GitUI for cli
alias rit="gitui"

alias ports="lsof -PiTCP -sTCP:LISTEN"
alias gc="git -c http.sslVerify=false clone"
alias git="git -c http.sslVerify=false"
alias gch="git checkout"
alias xclip="xclip -selection c"
alias speedtest="curl -o /dev/null
cachefly.cachefly.net/100mb.test"
[End => ~/.dots/alias.config]

hx ~/.dots/func.config

[Content => ~/.dots/func.config]
# Functions
google() {
    if [ -z "$1" ]; then
```

```

        echo "No argument supplied"
        return
    fi
    echo "Searching for $@"
    search_string="$@"
    open "https://www.google.com/search?
q=$search_string"
}

lazygit() {
    USAGE="
lazygit [OPTION]... <msg>
    GIT but lazy
    Options:
        --fixup <commit>           runs 'git commit --fixup
<commit> [...]'
        --amend                     runs 'git commit --amend
--no-edit [...]'
        -f, --force                 runs 'git push --force-
with-lease [...]'
        -h, --help                 show this help text
"
    while [ $# -gt 0 ]
    do
        key="$1"

        case $key in
            --fixup)
                COMMIT="$2"
                shift # past argument
                shift # past value
                ;;
            --amend)
                AMEND=true
                shift # past argument
                ;;
            -f|--force)
                FORCE=true
                shift # past argument
                ;;
            -h|--help)
                echo "$USAGE"
                EXIT=true
                break
                ;;
            *)
                MESSAGE="$1"

```

```

        shift # past argument
        ;;
    esac
done
unset key
if [ -z "$EXIT" ]
then
    git status .
    git add .
    if [ -n "$AMEND" ]
    then
        git commit --amend --no-edit
    elif [ -n "$COMMIT" ]
    then
        git commit --fixup "$COMMIT"
        GIT_SEQUENCE_EDITOR=: git rebase -i --
autosquash "$COMMIT"^
    else
        git commit -m "$MESSAGE"
    fi
    git push origin HEAD $([ -n "$FORCE" ] && echo
'--force-with-lease')
    fi
    unset USAGE COMMIT MESSAGE AMEND FORCE
}

```

[End => ~/.dots/func.config]

```

# Let's modify the zshrc
rm -rf ~/.zshrc
hx ~/.zshrc

```

[Content => ~/.zshrc]

```

# Exports
export ZSH="$HOME/.oh-my-zsh"
export LC_ALL=en_US.UTF-8
export LSCOLORS=""

```

```

# Oh-My-Zsh Variables
ZSH_THEME=""
ENABLE_CORRECTION="false"
HOMEBREW_NO_ENV_HINTS="true"
CACHED_PATH="$HOME"

```

```

# Oh-My-Zsh Configs
plugins=(zsh-autosuggestions zsh-completions fast-
syntax-highlighting git docker gh rust cp github)
autoload -U compinit && compinit

```



```
# Evals
eval "$(starship init zsh)"
eval "$(zoxide init zsh)"

# Completion config
fpath+=${ZSH_CUSTOM:-${ZSH:-~/.oh-my-zsh}/custom}/
plugins/zsh-completions/src

# Starting Oh-My-Zsh
source $ZSH/oh-my-zsh.sh

# Alias to existing functions
source $HOME/.dots/alias.config

# Functions
source $HOME/.dots/func.config
[End => ~/.zshrc]
```