Arch Linux

AISK

January, 2022

Contents

1. Flash USB

1.1 Download Arch ISO

1. **Download Arch ISO from:**
   https://archlinux.org/download/
2. **Verify Download:**

```
user$ sha1sum <archlinux-YYYY.MM.DD-x86_64.iso>
```

1.2 USB Preparation

1. **Unmount FS!**
2. **Create Partition Table**

```
root# parted -s </dev/sdX> mktable gpt
```

3. **Print change:**

```
root# parted </dev/sdX> (p)rint [free]
```

1.3 Flash ISO to USB

1. **Unmount FS!**
2. **Flash to USB (/dev/sdX):**

```
root# dd if=<./archlinux-YYYY.MM.DD-x86_64.iso> of=</dev/sdX>
[bs=4M | status=progress]
```

1.4 Boot Live Installer

1.4.1 Secure Boot

Make sure, that Secure Boot is Disabled!

1. **During POST press Key to access BIOS/UEFI:**
   BIOS/UEFI Menu Keys For All Vendors

2. **Disable Secure Boot**
3. **Poweroff/Restart**

1.4.2 Boot

1. **Plug in Flashed USB**
2. **During POST press Key to access Boot Menu:**
   Boot Menu Keys For All Vendors
3. **Select USB entry.**

2. Pre-Installation

2.1 Check Disk for bad sectors

2.1.1 Theory

- **Sector:** smallest unit size on disk. Usually 512 bytes, but some hard disks have 4096.
- **Block:** Allocation size the FS uses. Cannot be smaller than size of the sector. Can be group of sectors (4096b: 8 x 512b sectors).
  - **512b** = good for lot of small files. More blocks = more metadata.
  - **4096b** = good for larger files, less metadata. Waste if there are mostly small files.

2.1.2 Disk Info gathering

- **Find disks (block devices):**

```
user$ lsblk [-ap | -apf]
root# fdisk -l [/dev/sdX]
root# gdisk -l </dev/sdX>
root# blkid
```

- **Get raw disk info:**
  - **Disk size in bytes:**

```
root# blockdev [-v] --getsize64 </dev/sdX[Y]>
```

  - **Disk block size in bytes:**

```
root# blockdev [-v] --getbsz </dev/sdX[Y]>
```

  - **Check if disk is readonly (1 = ro, 0 = rw):**

```
root# blockdev [-v] --getro </dev/sdX[Y]>
```

- **See partitions:**

```
root# parted </dev/sdX> (p)rint [free]
```

2.1.3 Check Disk for bad sectors

1. **Unmount FS!**
2. **Check disk for bad blocks:**

```
root# badblocks [-b 4096] [-w [-t 0xaa]] [-v] [-s]
</dev/sdX[Y]> | tee -a <OUTPUT_FILE>
```

3. Installation

3.1 ISO specific

3.1.1 Remove pcspkr

- **Remove pcspkr module:**

```
root# modprobe -r pcspkr
```

3.1.2 Connect to WiFi

1. **Enable WiFi:**

```
root# rfkill unblock wlan
```

2. **Start services:**

```
root# systemctl start wpa_supplicant.service dhcpcd.service
```

3. **Configure WiFi:**
File (/etc/wpa_supplicant/wpa_supplicant.conf):

```
ctrl_interface=/run/wpa_supplicant
update_config=1
country=<2-LETTER-ISO-CODE>

# WPA-PSK protected:
network={
    ssid="<ESSID>"
    scan_ssid=1 # Find hidden network
    key_mgmt=WPA-PSK
    psk="<PLAINTEXT-PASSWD>"
    #psk=<32byte-HEX-NUMBER>
    priority=1 # To which WiFi connect first
}

# WPA-EAP protected::
network={
```

```
        ssid="<ESSID>"
        scan_ssid=1 # Find hidden network
        key_mgmt=WPA-EAP
        #eap=PEAP
        identity="<USERNAME>@<DOMAIN>"
        password="<PLAINTEXT-PASSWD>"
        #psk=<32byte-HEX-NUMBER>
        #ca_cert="/etc/cert/ca.pem"
        #phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
        priority=2 # To which WiFi connect first
}

# Unprotected:
network={
    ssid="<ESSID>"
    scan_ssid=1 # Find hidden network
    key_mgmt=NONE
    priority=3 # To which WiFi connect first
}
```

4. **Connect to WiFi:**

```
root# wpa_supplicant -B -D wext -i <wlan0>
-c </etc/wpa_supplicant/wpa_supplicant.conf>
```

3.2 Disk Partitioning

3.2.1 GPT + BIOS + UEFI + Encrypted LVM

- **Disk Layout:**

```
+-----------+------------+---------------+------------+-----------+
| PARTITION |    SIZE    |  FILESYSTEM   | MOUNTPOINT |   FLAGS   |
+-----------+------------+---------------+------------+-----------+
| /dev/sdX1 |      1 MiB |         FAT32 |       NONE | bios_grub |
+-----------+------------+---------------+------------+-----------+
| /dev/sdX2 |    512 MiB |         FAT32 |    /efi    | boot, esp |
+-----------+------------+---------------+------------+-----------+
| /dev/sdX3 |  100% FREE | crypto_LUKS 1 |
+-----------+------------+---------------+-----------+
```

1. **Get info about disks:**
   See section **2.1.2**.

2. **Create GPT Partition Table:**

```
root# parted -s </dev/sdX> mktable gpt
```

3. **Create partitions:**

   (a) **Enter cfdisk:**

   ```
   root# cfdisk </dev/sdX>
   ```

   (b) **BIOS partition - store 2nd stage of BIOS bootloader (1 MiB):**

   ```
   cfdisk> n
   cfdisk> 1MiB
   cfdisk> t
   cfdisk> BIOS boot
   ```

   (c) **EFI partition (max 512 MiB):**

   ```
   cfdisk> n
   cfdisk> 512MiB
   cfdisk> t
   cfdisk> EFI System
   ```

   (d) **Encrypted partition:**

   ```
   cfdisk> n
   cfdisk>   (Enter)
   ```

   (e) **Write Changes:**

   ```
   cfdisk> W
   cfdisk> yes
   ```

   (f) **Quit cfdisk:**

   ```
   cfdisk> Q
   ```

   (g) **Name partitions:**

   ```
   root# parted -s </dev/sdX> name 1 BIOS
   root# parted -s </dev/sdX> name 2 UEFI
   root# parted -s </dev/sdX> name 3 LUKS
   ```

4. **Format Partitions:**

   (a) **BIOS partition:**

   ```
   root# mkfs.fat -F 12 </dev/sdX1>
   root# fatlabel </dev/sdX1> <BIOS>
   ```

   (b) **EFI partition:**

   ```
   root# mkfs.fat -F 32 </dev/sdX2>
   root# fatlabel </dev/sdX2> <UEFI>
   ```

   (c) **Encrypted partition:**

```
root# cryptsetup luksFormat --type luks1 [--label "LUKS"]
</dev/sdX3>
sudo cryptsetup config /dev/sdb1 --label YOURLABEL
> YES
> <PASSWORD>
> <PASSWORD (VERIFY)>
```

5. **OPTIONAL LUKS stuff:**
   - **Open LUKS:**
     ```
     root# cryptsetup open --type luks </dev/sdX3> <luks>
     > <PASSWORD>
     ```
   - **Close LUKS:**
     ```
     root# cryptsetup close <luks>
     ```
   - **LUKS header:**
     (a) **See LUKS header:**
     ```
     root# cryptsetup luksDump </dev/sdX2>
     ```
     (b) **Make LUKS header backup:**
     ```
     root# cryptsetup luksHeaderBackup </dev/sdX2>
     --header-backup-file <FILE>
     ```
     (c) **Destroy LUKS header :**
     ```
     root# cryptsetup luksErase </dev/sdX2>
     ```
     (d) **restore LUKS header:**
     ```
     root# cryptsetup luksHeaderRestore </dev/sdX2>
     --header-backup-file <FILE>
     ```

6. **Create LVM partition:**
   (a) **Open Encrypted partition:**
   ```
   root# cryptsetup open --type luks </dev/sdX3> <luks_lvm>
   > <PASSWORD>
   ```
   (b) **Create Physical Volume to be used by LVM Volume Group:**
   ```
   root# lvm pvcreate </dev/mapper/luks_lvm>
   ```
   (c) **Create LVM Volume Group *vg0*:**
   ```
   root# lvm vgcreate <vg0> </dev/mapper/luks_lvm>
   ```
   (d) **Create Logical Volumes:**
   i. **Boot partition (512 MiB):**
   ```
   root# lvm lvcreate -L 512MiB -n boot <vg0>
   ```
   ii. **Root partition (rest of the disk):**
   ```
   root# lvm lvcreate -l 100%FREE -n root <vg0>
   ```
   (e) **Show LVM stuff:**

- **Physical Volumes:**
  ```
  root# lvm pvdisplay
  ```
- **Volume Groups:**
  ```
  root# lvm vgdisplay [vg0]
  ```
- **Logical Volumes:**
  ```
  root# lvm lvdisplay
  ```

7. **Format LVM volumes:**

   (a) **Boot Partition:**
   ```
   root# mkfs.ext4 </dev/vg0/boot>
   root# e2label </dev/vg0/boot> <LVM_BOOT>
   ```

   (b) **Root Partition:**
   ```
   root# mkfs.ext4 </dev/vg0/root>
   root# e2label </dev/vg0/root> <LVM_ROOT>
   ```

3.3 Mount FS

1. **Mount Root filesystem:**
   ```
   root# mount </dev/vg0/root> </mnt/>
   ```
2. **Create boot dir:**
   ```
   root# mkdir </mnt/boot/>
   ```
3. **Mount boot partition:**
   ```
   root# mount </dev/vg0/boot> </mnt/boot/>
   ```
4. **Create efi dir:**
   ```
   root# mkdir </mnt/efi/>
   ```
5. **Mount efi partition:**
   ```
   root# mount </dev/sdX2> </mnt/efi/>
   ```

3.4 Install Arch

1. **Check Mirrors:**
   ```
   root# cat /etc/pacman.d/mirrorlist
   ```
2. **Download Arch:**
   This installs BASE packages, LINUX kernel and common LINUX-FIRMWARE for common hardware:
   ```
   root# pacstrap </mnt/> base linux [linux-firmware]
   ```

3. **Generate fstab:**

```
root# genfstab -U </mnt/> >> /mnt/etc/fstab
```

4. **Chroot into arch:**

   (a) **Mount filesystems:**

```
root# mount -t proc /proc/ </mnt/proc/>
root# mount --rbind /sys/ </mnt/sys/>
root# mount --make-rslave </mnt/sys/>
root# mount --rbind /dev/ </mnt/dev/>
root# mount --make-rslave </mnt/dev/>
```

   (b) **Chroot to root filesystem:**

```
root# chroot </mnt/> /bin/bash
```

5. **Set up DNS for chrooted environment:**

```
[root#] echo "nameserver 1.1.1.1" > /etc/resolv.conf
```

6. **Install packages:**

   - **Install LVM support:**

```
[root#] [yes |] pacman -S lvm2
```

   - **Install vim:**

```
[root#] [yes |] pacman -S vim
```

7. **Add encrypted support to mkinitcpio:**
   File (**/etc/mkinitcpio.conf**):

```
...
HOOKS=(base udev autodetect modconf block keyboard keymap
consolefont encrypt lvm2 filesystems fsck)
...
```

8. **Recreate initramfs with encrypted support:**

```
[root#] mkinitcpio -P
```

3.5 Customize settings

3.5.1 Time

1. **Select timezone:**

```
[root#] ln -sf </usr/share/zoneinfo/Europe/Copenhagen>
/etc/localtime
```

2. **Update HW clock (generate: /etc/adjtime):**

```
[root#] hwclock --systohc
```

3.5.2 Locales

1. **Select locales:**
   File (/etc/locale.gen):

   ```
   ...
   en_US.UTF-8 UTF-8
   en_US ISO-8859-1
   ...
   ```

2. **Generate locales:**

   ```
   [root#] locale-gen
   ```

3. **Set language:**
   File (/etc/locale.conf):

   ```
   LANG=en_US.UTF-8
   ```

4. **Set keyboard:**
   File (/etc/vconsole.conf):

   ```
   KEYMAP=us
   ```

3.5.3 Network

1. **Set hostname:**
   File (/etc/hostname):

   ```
   <HOSTNAME>
   ```

2. **Install network packages:**

   - **Install WiFi control:**
     ```
     [root#] [yes |] pacman -S wpa_supplicant
     ```
   - **Install DHCP client:**
     ```
     [root#] [yes |] pacman -S dhcpcd
     ```

3.6 Install bootloader

1. **Download packages:**

   ```
   [root#] [yes |] pacman -S efibootmgr grub
   ```

2. **Find UUID of encrypted luks fs:**

   ```
   [root#] blkid | grep "crypto_LUKS"
   ```

3. **Edit GRUB config for encryption:**
   File (/etc/default/grub):

```
...
GRUB_CMDLINE_LINUX="cryptdevice=UUID=<UUID>:<luks_lvm> \
root=</dev/vg0/root>"
...
GRUB_ENABLE_CRYPTODISK=y
...
```

4. **Make sure EFI partition is mounted!**
   See section **3.3**.

5. **Install GRUB for UEFI:**

```
[root#] grub-install --target=x86_64-efi --efi-directory=</efi/>
[--bootloader-id=<Arch_UEFI>] --recheck [--removable]
```

6. **Make sure BOOT partition is mounted!**
   See section **3.3**.

7. **Install GRUB for BIOS:**

```
[root#] grub-install --target=i386-pc [--boot-directory=</boot/>]
[--bootloader-id=<Arch_BIOS>] --recheck </dev/sdX>
```

8. **Make/Update GRUB config file:**

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

9. **Avoid entering password twice:**

   (a) **Create LUKS key:**

```
[root#] mkdir </root/.luks/>
[root#] dd bs=512 count=4 if=</dev/random> of=</root/.luks/key>
iflag=fullblock
[root#] chmod 0000 </root/.luks/key>
[root#] cryptsetup -v luksAddKey </dev/sdX3> </root/.luks/key>
```

   (b) **Add LUKS key to initramfs image:**
   File (**/etc/mkinitcpio.conf**):

```
...
FILES=(/root/.luks/key)
...
```

   (c) **Recreate initramfs image:**

```
[root#] mkinitcpio -P
```

   (d) **Set initramfs files privileges:**

```
[root#] chmod 0600 /boot/initramfs-linux*
```

   (e) **Add GRUB parameter to unlock LUKS using encrypt hook:**
   File (**/etc/default/grub**):

```
...
GRUB_CMDLINE_LINUX="cryptdevice=UUID=<UUID>:<luks_lvm> \
root=</dev/vg0/root> cryptkey=rootfs:/root/.luks/key"
...
GRUB_ENABLE_CRYPTODISK=y
...
```

(f) **Make/Update GRUB config file:**

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

3.7 Finish installation

3.7.1 Root Password

1. **Create root password:**

```
[root#] passwd root
> <PASSWORD>
> <PASSWORD-VERIFY>
```

3.7.2 Finish installation

1. **Exit chroot:**

```
[root#] exit
```

2. **Umount disk partitions:**

```
root# umount -R </mnt/>
```

3. **Reboot:**

```
root# poweroff
```

4. Post-Installation

4.1 Disable pcspkr

1. **Blacklist pcspkr module:**
   File (/etc/modprobe.d/blacklist.conf):

   ```
   blacklist pcspkr
   ```

4.2 Create SWAP

https://chrisdown.name/2018/01/02/in-defence-of-swap.html

1. **Allocate size for swap file:**

   ```
   root# fallocate -l <2GB> /swap
   ```

2. **Set permissions for swap file:**

   ```
   root# chmod 0600 /swap
   ```

3. **Make swap file:**

   ```
   root# mkswap /swap
   ```

4. **Activate swap file:**

   ```
   root# swapon /swap
   ```

5. **Add swap file to fstab:**
   File (/etc/fstab:)

   ```
   ...
   ## Swap:
   /swap   none    swap    sw    0 0
   ```

4.3 Improve EXT4 performance

1. **Add parameters to ext4:**
   File (/etc/fstab):

```
...
UUID=<UUID> <MOUNT_POINT> <ext4> <rw>,noatime,commit=60 <0> <1>
...
```

4.4 Create user

1. **Set up privilege escalation:**
   (a) **Install:**
   ```
   root# [yes |] pacman -S doas
   ```
   (b) **Create group for privilege escalation:**
   ```
   root# groupadd <doas>
   ```
   (c) **Create privilege rules for *doas* group:**
   File (/etc/doas.conf):
   ```
   ## <permit|deny> [nopass|persist] [USER]:[GROUP] [as <USER2>]
   [cmd <COMMAND> [args <ARGUMENTS>]
   permit [nopass] :<doas>
   ```
   (d) **Create symlink for apps that needs sudo:**
   ```
   root# ln -sf /usr/bin/doas /usr/bin/sudo
   ```
2. **Create user with home directory:**
   (a) **Create user with home directory:**
   ```
   root# useradd -m [-G <doas>] [-s </bin/bash>] <USER>
   ```
   (b) **Create password for this user:**
   ```
   root# passwd <USER>
   ```

5. Package Manager

https://wiki.archlinux.org/title/Pacman/Rosetta

5.1 Settings

5.1.1 Select mirrors

- **Select Mirrors:**
  File (**/etc/pacman.d/mirrorlist**).

5.1.2 Configure pacman

- **Select Mirrors:**
  File (**/etc/pacman.conf**):

```
...
### Ignore during update:
## Ignore package from being updated:
#IgnorePkg=<PKG> [PKG2]
## Do not touch file when upgrading:
#NoUpgrade=</PATH/TO/FILE> [/PATH/TO/FILE2]

###  Misc:
## Allow colors:
Color
## Check for available disk space:
CheckSpace
## Verbose info for download and update:
VerbosePkgLists
## Easter egg:
ILoveCandy
...
## Allow multilib:
[multilib]
```

```
Include = /etc/pacman.d/mirrorlist
...
```

5.2 Pacman

- **Packages:**
  https://archlinux.org/packages/

5.2.1 Update PKG

- **Update system:**

```
root# pacman -Syu
```

- **Update system and do not use cache if used few minutes ago:**

```
root# pacman -Syyu
```

5.2.2 List PKGs

- **List all installed PKGs:**

```
root# pacman -Q[q]
```

- **List specifically installed PKGs:**

```
root# pacman -Qe[q]
```

- **List unneeded dependencie PKGs:**

```
root# pacman -Qdt[q]
```

5.2.3 Search PKG

- **Search packages that contains word in title/description:**

```
root# pacman -Ss[q] <WORD>
```

- **Search installed packages that contains word in title/description:**

```
root# pacman -Qs[q] <WORD>
```

5.2.4 Install PKG

- **Install specific package:**

```
root# [yes |] pacman [--need] -S <PKG> [PKG2]
```

- **Install specific package from different repository:**

```
root# [yes |] pacman [--need] -S <REPO>/<PKG>
```

- **Install packages matching regex:**

```
root# [yes |] pacman [--need] -S $(pacman -Ssq "<REGEX>")
```

- **Install packages with similar pattern:**

```
root# [yes |] pacman [--need] -S <plasma-{desktop,nm}>
```

5.2.5 Remove PKG

- **Remove specific package:**

```
root# [yes |] pacman -R <PKG>
```

- **Remove specific package with it's dependencies:**

```
root# [yes |] pacman -Rs <PKG>
```

- **Remove specific package with it's dependencies and system config files (not dotfiles):**

```
root# [yes |] pacman -Rns <PKG>
```

- **Remove old versions of installed packages:**

```
root# [yes |] pacman -Sc[c]
```

5.3 PKG Licenses

1. **Install package:**

```
root# [yes |] pacman -S expac
```

2. **Query package to see its license:**

```
root# expac [-Ss|-Qs] '% - %n' <REGEX>
```

5.4 AUR

https://aur.archlinux.org/

5.4.1 Manual Installation:

Setting up

1. **Install:**

```
root# [yes |] pacman -S base-devel git
```

2. **Change compilation variables:**
File (/etc/makepkg.conf):

```
...
MAKEFLAGS="-j$(nproc)"
...
```

3. **Create directory for AUR packages:**

```
root# mkdir </etc/AUR/>
```

4. **Transfer directory ownership:**

```
root# chown -R :doas </etc/AUR/>
```

5. **Make directory writable for *doas* group:**

```
root# chmod -R 0775 </etc/AUR/>
```

Install package

1. **Find package in AUR repository:**
   https://aur.archlinux.org/
2. **Navigate to *AUR* directory:**

```
user$ cd </etc/AUR/>
```

3. **Clone AUR package:**

```
user$ git clone <https://aur.archlinux.org/bvi.git>
```

4. **Go to cloned directory:**

```
user$ cd <./bvi/>
```

5. **Check AUR package content:**

```
user$ less ./PKGBUILD
```

6. **Compile package:**

```
user$ makepkg -si
```

7. **If there was GPG key fault:**
   (a) **Import key:**

```
user$ gpg --recv-keys <KEY>
```

   (b) **Compile again:**

```
user$ makepkg -si
```

6. GRUB

6.1 Configuration

1. **Basic GRUB settings:**
   File (/etc/default/grub):

```
## Do not rename network interfaces and preserve
## default ethX and wlanX names:
GRUB_CMDLINE_LINUX="net.ifnames=0"
## GRUB menu default option:
GRUB_DEFAULT=0
## GRUB menu timeout:
GRUB_TIMEOUT=1
## GRUB distributor:
GRUB_DISTRIBUTOR="Arch"
## Disable recovery mode entry in GRUB menu:
GRUB_DISABLE_RECOVERY=true

GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet"

## Encrypted GRUB on /boot/:
GRUB_CMDLINE_LINUX="cryptdevice=UUID=<UUID>:<luks_lvm> \
root=</dev/vg0/root> cryptkey=rootfs:/root/.luks/key"
GRUB_ENABLE_CRYPTODISK=y

## Preload both GPT and MBR modules so they are not missed:
GRUB_PRELOAD_MODULES="part_gpt part_msdos"

## GRUB BG image (*.jpg or *.png) - gfxterm only:
#GRUB_BACKGROUND="/boot/grub/<image.png>"

## Theme - gfxterm only:
#GRUB_THEME="/boot/grub/themes/<THEME>/theme.txt"
```

6.2 Menu Colors

Color BG	Color BG + FG
black	X
blue	light-blue
green	light-green
cyan	light-cyan
red	light-red
magenta	light-magenta
brown	yellow
light-gray	dark-gray

1. **Edit customization file:**
   File (/boot/grub/custom.cfg):

```
## <foreground>/<background>
set color_normal=white/black
set color_highlight=black/white
set menu_color_normal=white/black
set menu_color_highlight=black/white
```

6.3 Update GRUB

1. **Update GRUB:**

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

7. Local Settings

7.1 Hostname and DNSDomainname

1. **Display hostname and dnsdomainname:**

```
user$ hostname
user$ dnsdomainname
```

2. **Change hostname:**
   File (/etc/hostname):

```
<HOSTNAME>
```

   File (/etc/hosts):

```
...
## IPv4 localhost:
127.0.0.1   localhost
127.0.1.1   <HOSTNAME>
#127.0.0.1  <HOSTNAME>.<DOMAIN-NAME> <HOSTNAME>
## IPv6 localhost:
::1   localhost6
::1   <HOSTNAME>
#::1  <HOSTNAME>.<DOMAIN-NAME> <HOSTNAME>
...
```

7.2 Time and Date

1. **Show current timezone:**

```
user$ timedatectl -a
```

2. **List available timezones:**
   Dir: (/usr/share/zoneinfo/).

```
user$ timedatectl list-timezones
```

3. **Change timezone:**

   - **Timedatectl way:**

```
root# timedatectl set-timezone <UTC|Europe/Copenhagen>
```

- **Arch way:**

```
root# ln -sf </usr/share/zoneinfo/Europe/Copenhagen>
/etc/localtime
```

7.3 Locales and Keyboard

7.3.1 Locales

1. **View locales:**
   - **View locales:**

     ```
     user$ localectl
     ```
   - **View generated locales:**

     ```
     user$  localectl list-locales
     ```
2. **Generate locales to be used (uncomment them):**
   File (/etc/locale.gen):

   ```
   ...
   en_US.UTF-8 UTF-8
   en_US ISO-8859-1
   ...
   ```
3. **Generate locales:**

   ```
   root# locale-gen
   ```
4. **Set locale:**
   File (/etc/locale.conf):

   ```
   LANG=en_US.UTF-8
   ## First day in a week MON, not SUN:
   #LC_TIME="en_GB.UTF-8"
   ## Default paper size:
   #LC_PAPER="en_GB.UTF-8"
   #LC_MEASUREMENT="en_GB.UTF-8"
   ```
5. **OPTIONAL: export environment variable:**

   ```
   root# export LANG=en_US.UTF-8
   ```

7.3.2 CLI Keyboard

- **Language:**
  1. **See available keyboards:**
     Dir (/usr/share/kbd/keymaps/[i386/]).

2. **Show current keyboard:**
   `user$ localectl`
3. **Set keyboard:**
   File (/etc/vconsole.conf):
   `KEYMAP=us`

- **Font:**

1. **Show look of the font:**
   `user$ showconsolefont`
2. **Set font:**
   - **Temporary:**
     `user$ setfont <lat2-16>`
   - **Permanently:**
     File (/etc/vconsole.conf):
     `FONT=lat2-16`

8. Network

8.1 Rename Interface

1. **Find interface name in the system:**

```
user$ udevadm info /sys/class/net/<INTERFACE>
```

- **Manually rename interfaces:**
  File: (/etc/udev/rules.d/70-persistent-net.rules):

```
## ethernet (rename enp1s0 to eth0):
#SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", \
#ENV{ID_NET_NAME_PATH}="<enp1s0>", \
#ATTR{type}=="1", KERNEL=="eth*", NAME="<eth0>"
## wireless (rename wlp0s20f3 to wlan0):
#SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", \
#ENV{ID_NET_NAME_PATH}=="<wlp0s20f3>", \
#ATTR{type}=="1", KERNEL=="wlan*", NAME="<wlan0>"
```

8.2 Rfkill

1. **Block every RF device:**

```
user$ rfkill block all
```

2. **Unblock WiFi:**

```
user$ rfkill unblock wlan
```

8.3 Interfaces

8.3.1 Ethernet

1. **Install:**

```
root# pacman -S wpa_supplicant iw wireless_tools
```

2. **Check carrier speed:**

```
user$ ethtool <eth0>
```

8.3.2 Wireless

1. **Install:**

```
root# [yes |] pacman -S wpa_supplicant iw wireless_tools
```

2. **Do not run wpa_supplciant service at start:**

```
root# systemctl disable wpa_supplicant.service
```

3. **Scan for SSIDs:**

```
user$ iwlist <wlan0> scan [| grep -i ssid]
```

4. **WiFi config:**
File (/etc/wpa_supplicant/wpa_supplicant.conf):

```
## Basic settings and language for zones:
ctrl_interface=/run/wpa_supplicant
update_config=1
country=<2-LETTER-ISO-CODE>

## WPA-PSK protected:
network={
    ssid="<ESSID>"
    scan_ssid=1 # Find hidden network
    key_mgmt=WPA-PSK
    psk="<PLAINTEXT-PASSWD>"
    #psk=<32byte-HEX-NUMBER>
    priority=1 # To which WiFi connect first
}

## WPA-EAP protected::
network={
        ssid="<ESSID>"
        scan_ssid=1 # Find hidden network
        key_mgmt=WPA-EAP
        #eap=PEAP
        identity="<USERNAME>@<DOMAIN>"
        password="<PLAINTEXT-PASSWD>"
        #psk=<32byte-HEX-NUMBER>
        #ca_cert="/etc/cert/ca.pem"
```

```
        #phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
        priority=2 # To which WiFi connect first
}

## Unprotected:
network={
    ssid="<ESSID>"
    scan_ssid=1 # Find hidden network
    key_mgmt=NONE
    priority=3 # To which WiFi connect first
}
```

5. **Connect to WiFi:**

   (a) **Bring everything down for restart:**

   ```
   root# dhcpcd --release <wlan0>
   root# ip a flush <wlan0>
   root# ip l set <wlan0> down
   ```

   (b) **Start WiFi:**

   ```
   root# rfkill unblock wlan
   [root# macchanger -A <wlan0>]
   root# rm -rf /var/lib/dhcpcd/*
   root# rm -f /run/wpa_supplicant/<wlan0>
   #root# killall -9 wpa_supplicant
   root# ps -ef | grep "wpa_supplicant" | grep "<wlan0>" |
   tr -s ' ' | cut -d ' ' -f 2 | xargs kill -9
   root# ip l set <wlan0> up
   root# systemctl start wpa_supplicant.service
   root# systemctl start dhcpcd.service
   root# wpa_supplicant -B -D wext -i <wlan0>
   -c </etc/wpa_supplicant/wpa_supplicant.conf>
   root# dhcpcd <wlan0>
   ```

6. **Check WiFi stats:**

   - **Check state (and if connected WiFi interface:**

     ```
     user$ iw dev
     ```

   - **Check carrier speed:**

     ```
     user$ iwlist <wlan0> bitrate
     ```

8.4 DHCP client

1. **Install:**

```
root# [yes |] pacman -S dhcpcd
```

2. **Do not run dhcpcd service at start:**

```
root# systemctl disable dhcpcd.service
```

3. **Configure DHCP client:** File (/etc/dhcpcd.conf):

```
##################
### Anonymity ###
##################
## Send FQDN hostname to the DHCP server so it can be registered in DNS:
#hostname
## Send only hostname to the DHCP server:
#hostname_short
## Do not send DHCP option 60 (Vendor class id) in default format:
## dhcpcd-<version>:<os>:<machine>:<platform>
## (e.g. dhcpcd-5.5.6:NetBSD-6.99.5:i386:i386)
## Send empty vendorclassid (or not at all):
vendorclassid ""

##################
### IPv4/IPv6 ###
##################
## Don't attempt to obtain IPv4:
#noipv4
## Don't attempt to obtain IPv4LL
noipv4ll
## Don't attempt to obtain IPv6:
#noipv6
## Don't check if obtained IP address is taken by arp (increases speed):
noarp

######################
### IPv4 settings ###
######################
## Use MAC address in format xx:xx:xx and then is encoded as hex
## for interfaces whose MAC > 8 butes, clientid = "", and dhcpcd sends
## default clientid of HW family and HW address:
#clientid
```

```
## OR use DHCP Unique Identifier (DUID):
#duid
## Remove any pre-existing IPv4 addresses when adding IPv4 address:
noalias

######################
### IPv6 settings ###
######################
## Use MAC address when generating SLAAC address for the interface:
slaac hwaddr
## OR generate Stable Private IPv6 Address based from the DUID:
#slaac private

########################
### Default Gateway ###
########################
## Request default gateway (default):
gateway
## Don't obtain default gateway:
#nogateway

############
### DNS ###
############
## https://linux.die.net/man/5/dhcpd-options
## Request DNS servers from a server:
#option domain_name_servers
## Request domain-name for current network:
option domain_name
## Do not write to /etc/resolv.conf:
nohook resolv.conf

#################
### Required ###
#################
## A ServerID is required by RFC2131.
require dhcp_server_identifier
```

8.5 DNS (WIP!!!)

ToDo:

- /etc/resolv.conf

- DNS over HTTPS https://dns.sb/doh/
- DNSSEC https://wiki.archlinux.org/title/DNSSEC

1. **Install:**

```
root# pacman -S unbound [expat]
```

2. **Configure:**
File (/etc/unbound/unbound.conf):

```
server:
    ## Listen on loopback interface [on port 443]:
    interface: 127.0.0.1[@443]
    interface: ::1[@443]
```

3. **Configure DNS server list:**
File: (/etc/resolv.conf):

```
## Uncensored DNS - Denmark - Unicast
#nameserver 89.233.43.71
## CZ.NIC
#nameserver 193.17.47.1
#nameserver 185.43.135.1
## Quad9
#nameserver 1.1.1.1
#nameserver 1.0.0.1

## Use unbound as DNS resolver:
nameserver 127.0.0.1
nameserver ::1
## See https://man7.org/linux/man-pages/man5/resolv.conf.5.html
options trust-ad
```

1. **Install:**
   - **Link:** https://aur.archlinux.org/dnsproxy-adguard.git
   - **See 5.4.1.**
2. **Create SystemD service:**
3. **Run:**

```
dnsproxy-adguard -l <127.0.0.1> -p 53
-u <https://doh.dns.sb/dns-query> -b 185.222.222.222:53
```

8.6 NTP

Not needed!

9. Texteditor and Shell

9.1 Texteditor

9.1.1 Vim

1. **Install:**

```
root# [yes |] pacman -S vim
```

2. **Replace vi:**

```
root# ln -sf </usr/bin/vim> </usr/bin/vi>
```

3. **Set vim as default editor:**

```
user$ export EDITOR=vim
```

4. **Configure vim:** File (~/.vimrc):

```
"""""""""""""""""""""""""
"       SOUND           "
"""""""""""""""""""""""""
"" Turn off all sound:
set noerrorbells
set novisualbell
set t_vb=
set tm=500

"""""""""""""""""""""""""
"      BACKUPS          "
"""""""""""""""""""""""""
"" Turn backup off:
set nobackup
set nowb
set noswapfile

"""""""""""""""""""""""""
"   TAB and SPACES      "
"""""""""""""""""""""""""
```

```
"" Preserve indentation on new line:
set autoindent
"" Do not replace tabs with spaces:
set expandtab
"" Tab = 4 chars:
set tabstop=4
set shiftwidth=4
set softtabstop=4
"" For specific file types set different tab space:
autocmd FileType html setlocal tabstop=2 shiftwidth=2 softtabstop=2
autocmd FileType css setlocal tabstop=2 shiftwidth=2 softtabstop=2

"""""""""""""""""""""""""
"     STATUSLINE      "
"""""""""""""""""""""""""
"" Always show the status line:
set laststatus=2
"" Statusline:
set statusline=%F\ %p%%\ [%l:%c]\ %M\ %R\ %Y\ %y

"""""""""""""""""""""""""
"      SCROLLING      "
"""""""""""""""""""""""""
"" Start scrolling, N lines before reach end:
set scrolloff=4

"""""""""""""""""""""""""
"       SEARCH        "
"""""""""""""""""""""""""
"" Highlight search results:
set hlsearch

"""""""""""""""""""""""""
"    LINE NUMBERS     "
"""""""""""""""""""""""""
"" Number lines:
set number

"""""""""""""""""""""""""
" PROGRAMMING SYNTAX "
"""""""""""""""""""""""""
"" Enable syntax highlighting:
syntax on
"" Highlight trailing whitespace:
```

```
:highlight ExtraWhitespace ctermbg=red guibg=red
:match ExtraWhitespace /\s\+$/

"""""""""""""""""""""""""
" HIGHLIGHT BRACKETS "
"""""""""""""""""""""""""
"" Show other bracket:
"set showmatch

"""""""""""""""""""""""""
"    COLOR THEME      "
"""""""""""""""""""""""""
"colorscheme murphy
```

9.1.2 Bvi

(a) **Install:**
- **Link:** https://aur.archlinux.org/bvi.git
- **See 5.4.1.**

(b) **Configure:**
File (~/**.bvirc**):

```
"" http://bvi.sourceforge.net/set.html
"" Enable edit:
set memmove

"" Show status message bar at the bottom:
set showmode
```

9.2 Zsh (WIP)

1. **Install:**

```
root# [yes |] pacman -S zsh [zsh-autosuggestions]
[zsh-syntax-highlighting]
```

2. **Configure:**
File (~/**.zshrc**):

3. **Set ZSH as default shell:**
   - **User change:**
     ```
     user$ chsh -s /bin/zsh
     ```
   - **Root change:**
     ```
     root# usermod -s /bin/zsh <USER>
     ```

10. Xorg

1. **Install:**

11. i3-gaps

12. i3 programs

13. Power Management

14. Audio

15. Software

15.1 Help

- **Manuals:**
  1. **Install:**
     ```
     root# [yes |] pacman -S man-db man-pages
     ```
- **TLDR:**
  1. **Install:**
     ```
     root# [yes |] pacman -S tldr
     ```
  2. **Update it's cache:**
     ```
     root# tldr -u
     ```

16. System Maintenance

https://wiki.archlinux.org/title/System_maintenance

17. Things to consider in the future

- **Init system:** SystemD -> runit/OpenRC
- **X11 implementation:** Xorg -> Wayland
- **GPU driver:** Nvidia -> Nouveau
- **Virtualization:** use XEN

18. References

- **System Maintenance:**
  https://wiki.archlinux.org/title/System_maintenance
- **App list:**
  https://wiki.archlinux.org/title/List_of_applications
- **Boot Procedure:**
  https://wiki.archlinux.org/title/Arch_boot_process
- **Partition Optimal:**
  Partitioning
- **Booted from UEFI:**

```
root# ls /sys/firmware/efi/efivars
bootctl status
```

WIKI