# Arch Linux

AISK

January, 2022

# Contents

# 1. Flash USB

## 1.1 Download

1. **Navigate to:**
   https://archlinux.org/download/

2. **Verify Download:**

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

## 1.2 Disk Preparation

1. **Create Partition Table**

```
root# parted [-a <optimal>] </dev/sdX>
(parted) mktable <gpt|msdos>
```

2. **Print change:**

```
(parted) (p)rint [free]
```

3. **Quit parted:**

```
(parted) (q)uit
```

## 1.3 Flash ISO to USB

1. **Download ISO from:**
   `https://www.debian.org/distrib/`

2. **Unmount any mounted FS on HARD DRIVE!**

3. **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 Ket 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

# 2. Pre-Installation

## 2.1 Check Disk for bad sectors

### 2.1.1 Theory

- **Block:** group of sectors, every file must occupy at least 1 block. 0b file occupy whole block.
  - **512b** = good for lot of small files. More blocks = more metadata.
  - **4096b** = good for larger files, less metadata. Waste if there are small files.

### 2.1.2 Disk Info gathering

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

```
user$  lsblk [-ap | -apf]
root#  fdisk -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]>
    ```

### 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 Disk Partitioning

### 3.1.1 GPT UEFI

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

2. **Create Partition Table:**

   ```
   root# parted [-a <optimal>] </dev/sdX>
   (parted) mktable <gpt|msdos>
   ```

3. **Set Unit Size (sectors):**

   ```
   (parted) unit <s>
   ```

4. **Allow Legacy MBR Header for disk:**

   ```
   (parted) disk_set pmbr_boot on
   ```

5. **Create Partitions**

   (a) **See partitions and free space:**

   ```
   (parted) (p)rint free
   ```

   (b) **Partition - GRUB for legacy BIOS (cca 1MB):**
   + scapegoat partition for optimal partition alignment.

   ```
   (parted) mkpart primary <34s> <2047s>
   > (i)gnore
   (parted) name 1 bios
   (parted) set 1 bios_grub on
   ```

   (c) **Partition - EFI (>= 300MB -> 512MB):**

   ```
   (parted) mkpart primary <2048s> <1050623s>
   (parted) name 2 efi
   (parted) set 2 boot on
   (parted) set 2 esp on
   ```

(d) **Partition - LVM**

```
(parted) mkpart primary <1050624s> 100%
(parted) name 3 lvm
(parted) set 3 lvm on
```

(e) **Quit parted:**

```
(parted) (q)uit
```

6. **Create EFI Filesystem:**

```
root# mkfs.vfat </dev/sdX2>
```

7. **Encrypted LVM**

(a) **Encrypt LVM partition:**

```
root# cryptsetup luksFormat </dev/sdX3>
> YES
> <PASSWORD>
> <PASSWORD (VERIFY)>
```

(b) **Open Encrypted LVM partition:**

```
root# cryptsetup open --type luks </dev/sdX3> <lvm>
> <PASSWORD>
```

8. **OPTIONAL LUKS stuff:**

- **Close LUKS:**

```
root# cryptsetup close <lvm>
```

- **LUKS header:**

(a) **See LUKS header:**

```
root# cryptsetup luksDump </dev/sdX3>
```

(b) **Make LUKS header backup:**

```
root# cryptsetup luksHeaderBackup </dev/sdX3>
--header-backup-file <FILE>
```

(c) **Destroy LUKS header :**

```
root# cryptsetup luksErase </dev/sdX3>
```

(d) **restore LUKS header:**

```
root# cryptsetup luksHeaderRestore </dev/sdX3>
--header-backup-file <FILE>
```

9. **LVM Partitions**

(a) **Initialize disk/partition to be used by LVM:**

```
root# lvm pvcreate </dev/mapper/<lvm>>
```

(b) **Create volume group "vg0":**

```
root# vgcreate <vg0> </dev/mapper/<lvm>>
```

(c) **Logical Volume - SWAP (same as RAM size):**

```
root# lvcreate -L 16G -n swap <vg0>
```

(d) **Logical Volume - Root:**

```
root# lvcreate -l 100%FREE -n root <vg0>
```

10. **Create LVM Filesystems**

(a) **SWAP filesystem:**

```
root# mkswap </dev/mapper/vg0-swap>
```

```
root# swapon </dev/mapper/vg0-swap>
```

(b) **Root filesystem:**

```
root# mkfs.ext4 </dev/mapper/vg0-root>
```

### 3.1.2 Mount FS

1. **Mount Root filesystem:**

```
root# mount </dev/vg0/root> </mnt/>
```

2. **Create EFI dir:**

```
root# mkdir </mnt/efi/>
```

3. **Mount EFI partition:**

```
root# mount </dev/sdX2> </mnt/efi/>
```

## 3.2 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:**

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

5. **Install packages:**

- **Install LVM support:**

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

- **Install vim:**

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

6. **Add LVM support to mkinitcpio:**
   File (/etc/mkinitcpio.conf):

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

7. **Recreate initramfs for LVM:**

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

## 3.3 Customize settings

1. **Set Time:**

   (a) **Select timezone:**

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

   (b) **Update HW clock (generate: /etc/adjtime):**

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

2. **Set Locales:**

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

   ```
   ...
   en_US.UTF-8 UTF-8
   ...
   ```

   (b) **Generate locales:**

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

   (c) **Set language:**

   ```
   [root#] touch /etc/locale.conf
   ```

   File (/etc/locale.conf):

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

   (d) **Set keyboard:**

   ```
   [root#] touch /etc/vconsole.conf
   ```

   File (/etc/vconsole.conf):

   ```
   KEYMAP=us
   ```

3. **Set network:**

(a) **Set hostname:**

```
[root#] touch /etc/hostname
```

File (/etc/hostname):

```
<HOSTNAME>
```

(b) **Install network packages:**

```
[root#] pacman -S dhcpcd wpa_supplicant
```

## 3.4 Install bootloader

(a) **Download packages:**

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

(b) **Make sure EFI partition is mounted!**
See section **3.1.2**.

(c) **Install GRUB:**

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

(d) **Make GRUB config file:**

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

## 3.5 Finish installation

### 3.5.1 Root Password

(a) **Create root password:**

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

### 3.5.2 Finish installation

(a) **Exit chroot:**

```
[root#] exit
```

(b) **Umount disk partitions:**

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

(c) **Reboot:**

```
root# reboot
```

# 4. References

- **Boot Procedure:**
  https://wiki.archlinux.org/title/Arch_boot_process
- **Partition Optimal:**
  Partitioning
- **Booted from UEFI:**

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