Arch Linux

AISK

January, 2022

Contents

1. Flash USB

1.1 Download

1. **Download Arch ISO from:**
   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>
```

2. **Print change:**

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

3. **Quit parted:**

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

1.3 Flash ISO to USB

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

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

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 ISO specific

- **Remove pcspkr module:**

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

3.2 Disk Partitioning

3.2.1 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>
```

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

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

4. **Create Partitions**
   (a) **See partitions and free space:**

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

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

```
(parted) mkpart primary <2048s> <1050623s>
(parted) name 1 efi
(parted) set 1 boot on
(parted) set 1 esp on
```

   (c) **Partition - LVM**

```
(parted) mkpart primary <1050624s> 100%
(parted) name 2 lvm
(parted) set 2 lvm on
```

(d) **Quit parted:**

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

5. **Create EFI Filesystem:**

```
root# mkfs.vfat </dev/sdX1>
```

6. **Encrypted LVM**

(a) **Encrypt LVM partition:**

```
root# cryptsetup luksFormat </dev/sdX2>
> YES
> <PASSWORD>
> <PASSWORD (VERIFY)>
```

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

```
root# cryptsetup open --type luks </dev/sdX2> <lvm>
> <PASSWORD>
```

7. **OPTIONAL LUKS stuff:**

- **Close LUKS:**

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

- **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>
```

8. **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>
```

9. **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>
   ```

3.3 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/sdX1> </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. **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
   ```

3.5 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
   ...
   ```
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:**

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

3.6 Install bootloader

1. **Download packages:**

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

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

3. **Install GRUB:**

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

4. **Make 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# reboot
   ```

WIKI

4. References

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

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