

# **Artix Linux**

AI SK

January, 2022

# Contents

<b>1</b>	<b>Flash USB</b>	<b>3</b>
1.1	Download Arch ISO . . . . .	3
1.2	USB Preparation . . . . .	3
1.3	Flash ISO to USB . . . . .	3
1.4	Boot Live Installer . . . . .	3
1.4.1	Secure Boot . . . . .	3
1.4.2	Boot . . . . .	4
<b>2</b>	<b>Pre-Installation</b>	<b>5</b>
2.1	Check Disk for bad sectors . . . . .	5
2.1.1	Theory . . . . .	5
2.1.2	Disk Info gathering . . . . .	5
2.1.3	Check Disk for bad sectors . . . . .	5
<b>3</b>	<b>Installation</b>	<b>6</b>
3.1	ISO specific . . . . .	6
3.1.1	Remove pcspkr . . . . .	6
3.1.2	Connect to WiFi . . . . .	6
3.2	Disk Partitioning . . . . .	7
3.2.1	GPT UEFI . . . . .	7
3.3	Mount FS . . . . .	9
3.4	Install Arch . . . . .	9
3.5	Customize settings . . . . .	10
3.5.1	Time . . . . .	10
3.5.2	Locales . . . . .	11
3.5.3	Network . . . . .	11
3.6	Install bootloader . . . . .	11
3.7	Finish installation . . . . .	12
3.7.1	Root Password . . . . .	12
3.7.2	Finish installation . . . . .	12
<b>4</b>	<b>Easy Installation</b>	<b>13</b>
<b>5</b>	<b>References</b>	<b>15</b>

## 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. Create Partition Table

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

2. Print change:

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

### 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
3. **Select USB entry.**

## 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

#### 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={
```

### 3. INSTALLATION

---

```
    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 UEFI

#### 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) Create EFI Partition (max 512MiB):

```
cfdisk> n
cfdisk> 512MiB
```

```
cfdisk> t
cfdisk> EFI System
```

- (c) Create separate boot Partition (because of LUKS):

```
cfdisk> n
cfdisk> 512MiB
```

- (d) Create Root Partition:

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

- (e) Write Changes:

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

- (f) Quit cfdisk:

```
cfdisk> Q
```

4. Print change:

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

5. Create filesystems:

- (a) Create FAT32 for EFI:

```
root# mkfs.fat [-F 32] [-n "EFI"] </dev/sdX1>
```

- (b) Create EXT4 for boot:

```
root# mkfs.ext4 [-L "BOOT_GRUB"] </dev/sdX2>
```

- (c) Create Encrypted filesystem:

- i. Encrypt root partition:

```
root# cryptsetup [--label "LUKS"] luksFormat </dev/sdX3>
> YES
> <PASSWORD>
> <PASSWORD (VERIFY)>
```

- ii. Open Encrypted root partition:

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

- iii. Create EXT4 for Root:

```
root# mkfs.ext4 [-L "LUKS_ROOT"] </dev/mapper/luks_root>
```

6. OPTIONAL LUKS stuff:

- Close LUKS:

```
root# cryptsetup close <luks_root>
```

- 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>
```

## 3.3 Mount FS

1. Mount Root filesystem:

```
root# mount </dev/mapper/luks_root> </mnt/>
```

2. Create boot dir:

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

3. Mount boot partition:

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

4. Create EFI dir:

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

5. Mount EFI partition:

```
root# mount </dev/sdX1> </mnt/boot/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  
[intel-ucode|amd-ucode]
```

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 VFAT fs support:

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

- Install vim:

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

7. Add LVM support to mkinitcpio:

File (`/etc/mkinitcpio.conf`):

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

8. 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
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:**

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

## 3.6 Install bootloader

1. **Download packages:**

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

2. **Make sure EFI partition is mounted!**

See section **3.3**.

3. **Install GRUB:**

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

4. **Find UUID of encrypted root fs:**

```
[root#] blkid | grep "<cryptsetup>"
```

5. **Edit GRUB config for encryption:**

File (`/etc/default/grub`):

```
GRUB_CMDLINE_LINUX="cryptdevice=UUID=<UUID>:<cryptroot> \
root=</dev/mapper/cryptroot>"
```

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

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

[https://linuxlink.timesys.com/docs/engineering/wiki/HOWTO\\_Install\\_GRUB2\\_with\\_EFI\\_support](https://linuxlink.timesys.com/docs/engineering/wiki/HOWTO_Install_GRUB2_with_EFI_support)

<https://www.tecmint.com/arch-linux-installation-and-configuration-guide/>

## 4. Easy Installation

```
PARTITIONING:
cfdisk:
gpt
/dev/sda1 * 2048 ? 512M
/dev/sda2 ? END REST

mkfs.ext4 /dev/sda1
mkfs.ext4 /dev/sda2

mount /dev/sda2 /mnt
mkdir /mnt/boot
mount /dev/sda1 /mnt/boot

pacstrap /mnt base linux linux-firmware vim

genfstab -U /mnt >> /etc/fstab

arch-chroot /mnt /bin/bash

pacman -S grub

grub-install /dev/sda
grub-mkconfig -o /boot/grub/grub.cfg

passwd root

vim /etc/locale.gen

locale-gen

vim/etc/locale.conf
LANG=en-US.UTF-8

echo archlinux > /etc/hostname
```

#### 4. EASY INSTALLATION

---

```
exit
umount -R /mnt
reboot
```

WIKI

## 5. References

- **Boot Procedure:**  
[https://wiki.archlinux.org/title/Arch\\_boot\\_process](https://wiki.archlinux.org/title/Arch_boot_process)
- **Partition Optimal:**  
[Partitioning](#)
- **Booted from UEFI:**

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