

Arch Linux Install/Bonus Info

Nathan Warner

Feb 23, 2023

Pre Installation:

1 - Obtain Iso and Verify Sig

Visit the [Download Page](#) and acquire an iso image file. It is recommended to verify the image signature before use, especially when downloading from an HTTP mirror, where downloads are generally prone to be intercepted to serve malicious images.

On a system with GnuPG installed, do this by downloading the ISO PGP signature (under Checksums in the page Download) to the ISO directory, and verifying it with:

- `gpg --keyserver-options auto-key-retrieve --verify archlinux-version-x86_64.iso.sig`

Alternatively, from an existing Arch Linux installation run:

- `pacman-key -v archlinux-version-x86_64.iso.sig`

Note:-

The `gpg` command will output the fingerprint of the public key.

2 - Basic Input Output (BIOS) Tweaks

If your coming from windows you'll likely have to make adjustments in the BIOS settings before you proceed into the live enviornment.

- Make sure Intel Rapid Storage Technology (RST) is Off. Linux does not support RST.
- Make sure SATA mode is on AHCI
- Turn off *Fast Boot* and *Secure Boot*
- Make sure the boot sequence is UEFI

Note:-

If you boot into the `tty` and your system does not detect your block device, reboot into the `tty` on SATA mode: RST instead of AHCI, just be sure the change it back before you boot from disk after the install in complete.

Boot Live Enviornment and Install Steps

0.5 - Verify EFI Boot

Run:

- `ls sys/firmware/efi/efivars`

If you see a bunch of output, you are in UEFI mode

1 - Test Internet Connection

To test if wifi is connected, ping a webpage like archlinux.org

- `ping -c 3 archlinux.org`

If wifi is not connected, run:

- `iwctl`

Once you are in the **iwctl** shell, run the following commands:

- `device list`

If the device or its corresponding adapter is turned off, turn it on with:

- `device device set-property Powered on`

Or:

- `device device set-property Powered on`

Next, Initiate a scan for available networks:

- `station device scan`

You can then list all available networks:

- `station device get-networks`

Finally, to connect to a network:

- `station device connect SSID`

Once you have connected to wifi, rerun ping to ensure you are properly connected:

- `ping -c 3 archlinux.org`

2 - Set The Console Keyboard Layout

The default console keymap is US. Available layouts can be listed with either:

- `ls /usr/share/kbd/keymaps/**/*.map.gz`

Or:

- `localectl list-keymaps`

To set the keyboard layout, pass a corresponding file name to `loadkeys(1)`, omitting path and file extension.

- `loadkeys dvorak`

The current layout can be listed with:

- `localectl status`

3 - Update the system clock

In the live environment systemd-timesyncd is enabled by default and time will be synced automatically once a connection to the internet is established. Use timedatectl to ensure the system clock is accurate:

- `timedatectl set-ntp true`
- `timedatectl status`
- `timedatectl list-timezones`
- `timedatectl set-timezone America/Chicago`

4 - Partition the disks

When recognized by the live system, disks are assigned to a block device such as `/dev/sda`, `/dev/nvme0n1` or `/dev/mmcblk0`. To identify these devices run:

- `lsblk`

Or:

- `fdisk -l`

Note:-

If you want to create any stacked block devices for LVM, system encryption or RAID, do it now.

Once you have located the block name of the partition you wish to install onto:

use one of these options to create partitions:

- `fdisk`
- `parted`
- `cfdisk`

Dual Boot:

Enter ***cfdisk*** with the desired block name and do the following:

1. Locate the block in which you want to install arch onto, hover over it and click enter, from here, enter the size, but make sure to leave some space for ***Swap***, you will see that now you have the desired space as ***Linux File Space***
2. Hover over the free space that you left for ***Swap***, click enter and input the size, from here you will see you have created another linux filesystem, hover over ***type***, and change the type to ***Swap***
3. Now Choose ***Write***, and ***Quit***

Note:-

For ***Swap***, it is recommended you provide over **512mb**. By convention, if your RAM is less than **2gb**, swap should be **Double the RAM size**, For RAM size greater than **2gb**, it should be **RAM size, +2gb**

Non Dual Boot:

Enter ***cfdisk*** with the desired block name and do the following:

1. If your replacing your current linux system and you already have partitons here, be sure to delete them first
2. Click enter on **new**, create a efi system (for ***boot***), recommended size is **525mb**, but if multiple kernels are to be installed, no less than **1gb**
3. Next, create new partiton for **Swap**, See above note for size guide.
4. Next, create the **Root** partition, use the rest of the available space for this

5.1 - Format Partitions and Mount (Dual boot)

Once you have exited cfdisk with your newly created partitons, for **root**, run:

- `mkfs.ext4 /dev/partitionname` (for linux filesystem partiton)

Next, to activate the swap, run:

- `mkswap /dev/partitionname` (for swap partiton)
- `swapon /dev/partitionname`

Now you want to mount these partitons, for **root**, remember this is the partiton as type ***Linux Filesystem***:

- `mount /dev/partitionname /mnt`

For boot, since we are dual booting, we didn't need to make a partiton as ***Efi System***, since we can reuse the one for windows. So find out the partiton name for the type **Efi System**, and run:

- `mkdir /mnt/efi`
- `mount /dev/partitionname /mnt/efi`

5.2 Format Partitions and Mount (Non Dual Boot)

Once you have exited cfdisk with your newly created partitons, for **boot (Efi System)**, run:

- `mkfs.fat -F32 /dev/partitionname`

Next, to activate the **swap**, run:

- `mkswap /dev/partitionname` (for swap partiton)
- `swapon /dev/partitionname`

Next, for **root**, run:

- `mkfs.ext4 /dev/partitionname`

Now, we want to **mount** these partitions, for **root**, run:

- `mount /dev/partitionname /mnt`

For **boot (EFI SYSTEM)**, run:

- `mkdir /mnt/boot`
- `mkdir /mnt/boot/efi`
- `mount /dev/partitionname /mnt/boot/efi`

5 - Extra - Separate Home Partiton

To make a separate home partiton from root, run:

- `mkdir /mnt/home`
- `mount /dev/partitionname /mnt/home`

Note:-

The partiton name here is the partiton as type *Linux Filesystem*

6 - Pacstrap

To install the essential packages to make our system run, use the command:

- `pacstrap /mnt base linux linux-firmware`

7 - Genfstab

genfstab helps fill in an fstab file by autodetecting all the current mounts below a given mountpoint and printing them in fstab-compatible format to standard output. It can be used to persist a manually mounted filesystem hierarchy and is often used during the initial install and configuration of an OS.

- `genfstab -U /mnt » /mnt/etc/fstab`

8 - Chroot (Change root into new system)

To enter into the newly created file system, run:

- `arch-chroot /mnt`

9 - Set Timezone/Sync Hardware Clock

To link the timezone, run:

- `ln -sf /usr/share/zoneinfo/US/Central /etc/localtime`

To sync the hardware clock, run:

- `hwclock --systohc`

10 - Localization

Before we begin with this, we first need to install a text editor, Let's go with neovim:

- `pacman -Sy neovim`

Now edit:

- `nvim /etc/locale.gen`

Uncomment `en_US.UTF-8.UTF-8` and other needed locales, generate the locales by running:

- `locale-gen`

Create a new file in `/etc/locale.conf`:

- `nvim /etc/locale.conf`
- `LANG=en_US.UTF-8` (in the file)

If you set the console keyboard layout, make the changes persistent in `vconsole.conf`:

- `nvim /etc/vconsole.conf`
- `KEYMAP=dvorak` (in the file)

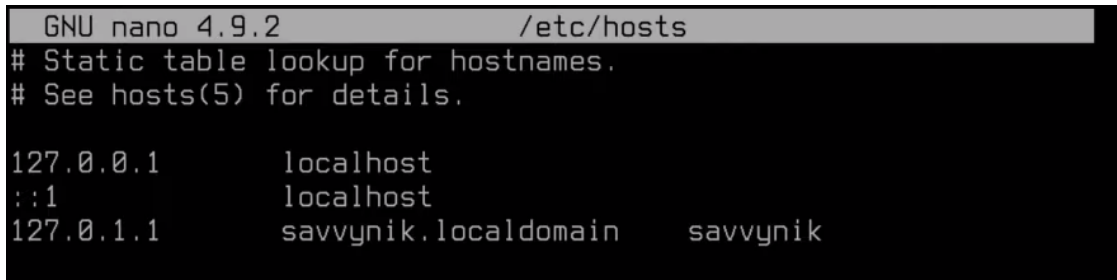
11 - Hostname

To create the hostname, run:

- `nvim /etc/hostname`
- `catsoup` (in the file)

Now setup `/etc/hosts`:

- `nvim /etc/hosts`



```
GNU nano 4.9.2 /etc/hosts
# Static table lookup for hostnames.
# See hosts(5) for details.

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

12 - Root Password

To set the root password, run:

- `passwd`

And enter a password

13 - Create a User

To create a user, run:

- `useradd -G wheel,audio,video -m username`
- `passwd username`

To give permissions to this user, run:

- `pacman -Sy vi`
- `visudo`

```
##
## User privilege specification
##
root ALL=(ALL:ALL) ALL
catsoup ALL=(ALL:ALL) ALL
## Uncomment to allow members of group wheel to execute any command
%wheel ALL=(ALL:ALL) ALL
```

14 - Install Packages (Part 1)

List of packages to install:

- `sudo`
- `networkmanager`
 - `systemctl enable NetworkManager`
- `grub`
- `efibootmgr`
- `os-prober`

15 - Grub

To setup grub, run:

- `grub-install --target=x86_64-efi --efi-directory=/efi/ --bootloader-id=GRUB`
- `grub-mkconfig -o /boot/grub/grub.cfg`

16 - Install Packages (Part 2)

List of packages to install:

- `xorg` (display server)
- `plasma` (Desktop Environment)
- `plasma-wayland-session`
- `kde-applications`
- `sddm`
 - `systemctl enable sddm`
- `git`

- intel-ucode
- bluez
- pulseaudio
- nvidia
 - nvidia-settings
 - nvidia-utlis
 - nvidia dkms
 - nvidia-libgl
 - lib32-nvidia-utils
- thunar
- qt5-multimedia
- vlc
- media-player-info

17 - Reboot

After installing the packages you need, reboot the system and make sure it boots grub

Note:-

If you are launched back into the tty, you may need to manually start your display manager, if your using *sddm*, run: **systemctl start sddm**

Bonus - AUR Helper (yay)

To install yay, do the following:

- `pacman -Sy -needed-base-devel git`
- `cd /opt`
- `sudo git clone https://aur.archlinux.org/yay-git.git`
- `sudo chown -R username ./yay-git`
- `cd yay-git`
- `makepkg -si`
- `sudo yay -Syu`

Bonus - Swap KWIN

To switch out kwin for a tiling window manager, run the following commands:

- `pacman -Sy qtile` (or whatever WM you desire)

To replace KWin in this startup, you must first mask the **plasma-kwin_x11.service** for the current user to prevent it from starting.

- `systemctl --user mask plasma-kwin_x11.service`

Then, create a new systemd user unit to start your preferred WM

```
~/.config/systemd/user/plasma-custom-wm.service
```

```
[Install]
WantedBy=plasma-workspace.target

[Unit]
Description=Plasma Custom Window Manager
Before=plasma-workspace.target

[Service]
ExecStart=/path/to/other/wm
Slice=session.slice
Restart=on-failure
```

Next, To use it, do (as user units) a **daemon-reload**, make sure you have masked **plasma-kwin_x11.service** then enable the newly created **plasma-custom-wm.service**

- `systemctl --user daemon-reload`
- `systemctl enable plasma-custom-wm.service`

Finally, logout and make sure **Plasma x11** is selected as **Session**

Bonus - Add Windows BootMGR to Grub

To add *Windows Boot Manager to grub*, first ensure that you have the package **os-prober**, then, run the command:

- `os-prober`

Next, you want to call `update-grub`, however, if `update-grub` is not a recognized command, you must make it yourself, to do this, do the following:

- `touch /usr/sbin/update-grub`
- `chmod +x /usr/sbin/update-grub`

Open this new file with a text editor and type in the following:

```
2 #!/bin/sh
1 set -e
3 exec grub-mkconfig -o /boot/grub/grub.cfg "$@"
1
```

Now all you need to do is execute this file. Once you restart your system, you should see that windows boot manager has been added to grub.

Bonus - Fix Booting Into Grub Rescue After Reinstalling Windows

Once in grub rescue, do the following commands:

- `ls`

From here you will see a list of partitions, type:

- `ls partition`

So for example:

- `ls (hd0, gpt1)`

Once you get one that outputs anything ***besides Unknown Filesystem***, do the following commands with that partition:

- `set boot=(hd0, gpt1)`
- `set prefix=(hd0, gpt1)/boot/grub`
- `insmod normal`
- `normal`

Note:-

Here I am using (hd0,gpt1), but make sure to replace it with the partiton you found.

Now you will see that grub boots normally, after you boot into your system, do the following:

- `grub-update`

If you dont have the grub-update script, see ***Bonus - Add Windows BootMGR to Grub*** to make the script, after you run update-grub, run the next command:

- `grub-install /dev/partition`

Note:-

replace *partition* with whatever partiton is **your efi system**. If you don't know which one is your **efi system**, run ***lsblk*** to find out

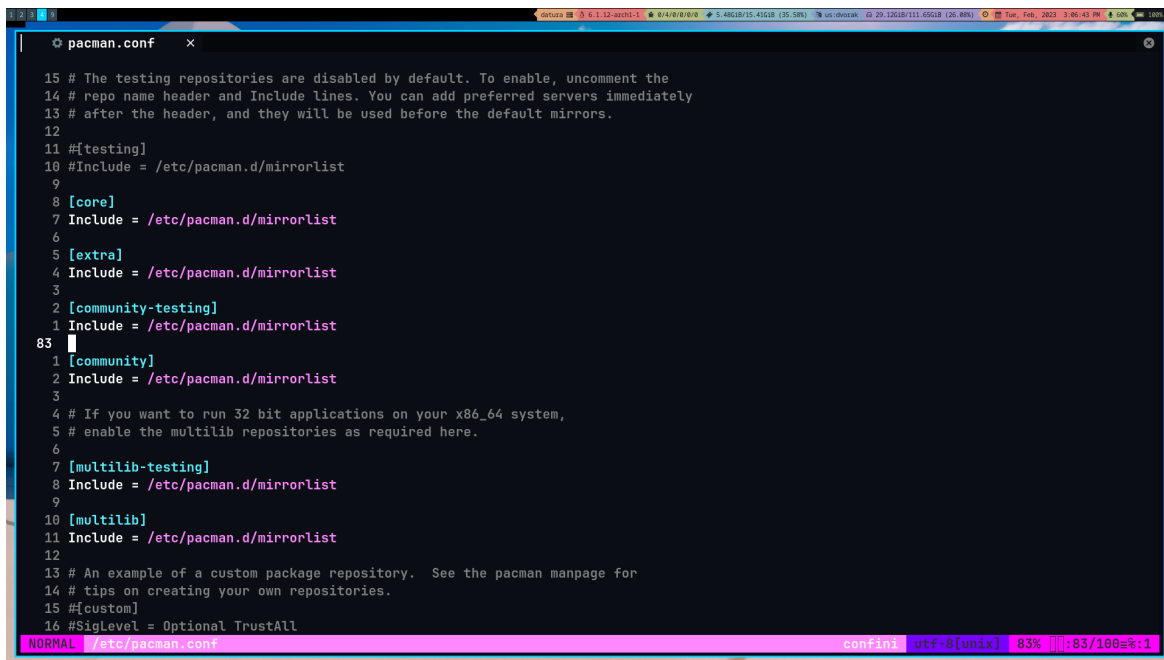
What to do if grub-install says that it cannot find efi directory:

This happens if your efi system is ***not*** mounted in ***/boot/efi***, if this is the case, do the following:

- `mkdir /boot/efi`
- `umount /dev/partition`
- `mount /dev/partition /boot/efi`
- `grub-install`

Remember, replace *partition*, with whatever partiton is your **efi system**, after you do these steps, the next time you reboot your machine, you should notice that grub is back to normal.

Bonus - pacman.conf



```
15 # The testing repositories are disabled by default. To enable, uncomment the
14 # repo name header and Include lines. You can add preferred servers immediately
13 # after the header, and they will be used before the default mirrors.
12
11 #{testing}
10 #Include = /etc/pacman.d/mirrorlist
9
8 [core]
7 Include = /etc/pacman.d/mirrorlist
6
5 [extra]
4 Include = /etc/pacman.d/mirrorlist
3
2 [community-testing]
1 Include = /etc/pacman.d/mirrorlist
83
1 [community]
2 Include = /etc/pacman.d/mirrorlist
3
4 # If you want to run 32 bit applications on your x86_64 system,
5 # enable the multilib repositories as required here.
6
7 [multilib-testing]
8 Include = /etc/pacman.d/mirrorlist
9
10 [multilib]
11 Include = /etc/pacman.d/mirrorlist
12
13 # An example of a custom package repository. See the pacman manpage for
14 # tips on creating your own repositories.
15 #[custom]
16 #SigLevel = Optional TrustAll
NORMAL /etc/pacman.conf conf:ini utf-8[unix] 83% 83/180a:1
```

Note:-

If some of these lines are commented, uncomment them and wq