Arch Linux - Installation Guidelines

horizontal line

First version: MARCH 25, **20**22 / 6:22 PM CT  
Last update: MARCH 27, 2022 / 7:13 PM CT  
  
00 Arch Linux ISO file

1. I’m assuming you already downloaded your Arch ISO, the image can be burned to a CD/DVD, mounted as an ISO file, or be directly written to a USB stick: <https://archlinux.org/download/>
2. Increase the font size:  
   **# setfont ter-132n**
3. Once the ISO is loaded, and you see the zsh command prompt, if you are installing Arch Linux on a VM, you can connect remotely via ssh, first, you will need to install the following packages:  
   ***# sudo pacman -Syy reflector openssh virtualbox-guest-utils***
4. You need to enable/start the ssh service:  
   ***# systemctl start sshd***
5. You need to set the password for the root user with:  
   ***# passwd***
6. Next, you need to find out what’s your IP address:  
   **# ip -c a**  
     
   If you need to connect to Wi-Fi, you’ll connect to the wireless networks with iwd following this process (<https://wiki.archlinux.org/title/Iwd#iwctl>):  
   **# iwctl**  
   -- to find the name of our Wireless adapter:  
   **[iwd]# device list**  
   -- next, we need to scan the networks available to Us (for this example the device name found on the previous step was wlan0):  
   **[iwd]# station wlan0 scan**  
   -- next, we need to display the networks available to Us:  
   **[iwd]# station wlan0 get-networks**  
   -- next, to connect to the desired network, you’ll be requested to enter the password for this network:  
   **[iwd]# station wlan0 connect <network-name>**  
   -- Exit the Wireless connection tool to go back to the root ISO:  
   **[iwd]# exit**  
     
   Once you know your IP, you can continue the installation of Arch Linux from your favorite terminal, something like this should do the ssh connection, per instance:  
   ***# ssh root@192.168.170.85***

## 01 Change your keyboard layout, if needed.

1. By default, you have english, if you need to change your layout to, let’s say, spanish, you can find your locale with:   
   # ***localectl list-keymaps | grep ES***
2. From the results, pick up the best that suits your needs, for example, to load the spanish keyboard layout in memory, execute something like:  
   # ***loadkeys es***

## 02 Sync your Network Time Protocol

**# timedatectl set-ntp true**

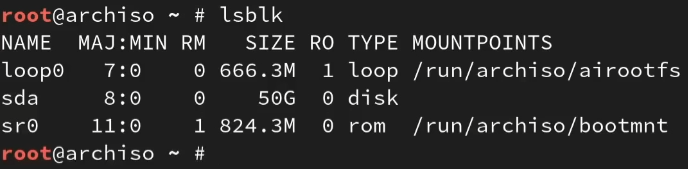
## 03 Update your mirrorlist with [reflector](https://wiki.archlinux.org/title/Reflector)

1. To list all available Arch Linux mirrors world-wide, run:  
   **# reflector --list-countries**
2. The following command retrieves the 15 fastest HTTPS mirrors located in the United States, sorts them by download speed, and overwrites the file /etc/pacman.d/mirrorlist file. **# reflector -c US -a 6 --f 15 --protocol https --sort rate --save /etc/pacman.d/mirrorlist**
3. Finally, you can synchronize the servers: **# pacman -Syy**

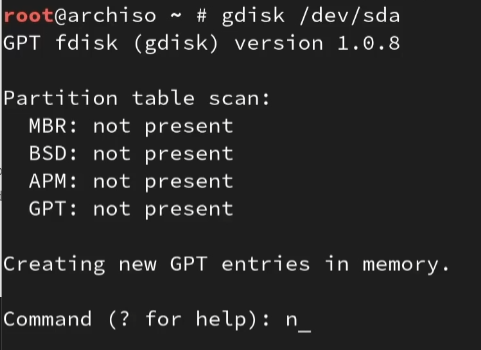
## 04 We will work on disk partition now…

First, we can try with the BTRFS file system, there are like a dozen different ways to do this, for this example we will install Arch on a disk called “sda”, so your disk could be something else, make sure to use the correct name:

1. **# lsblk**

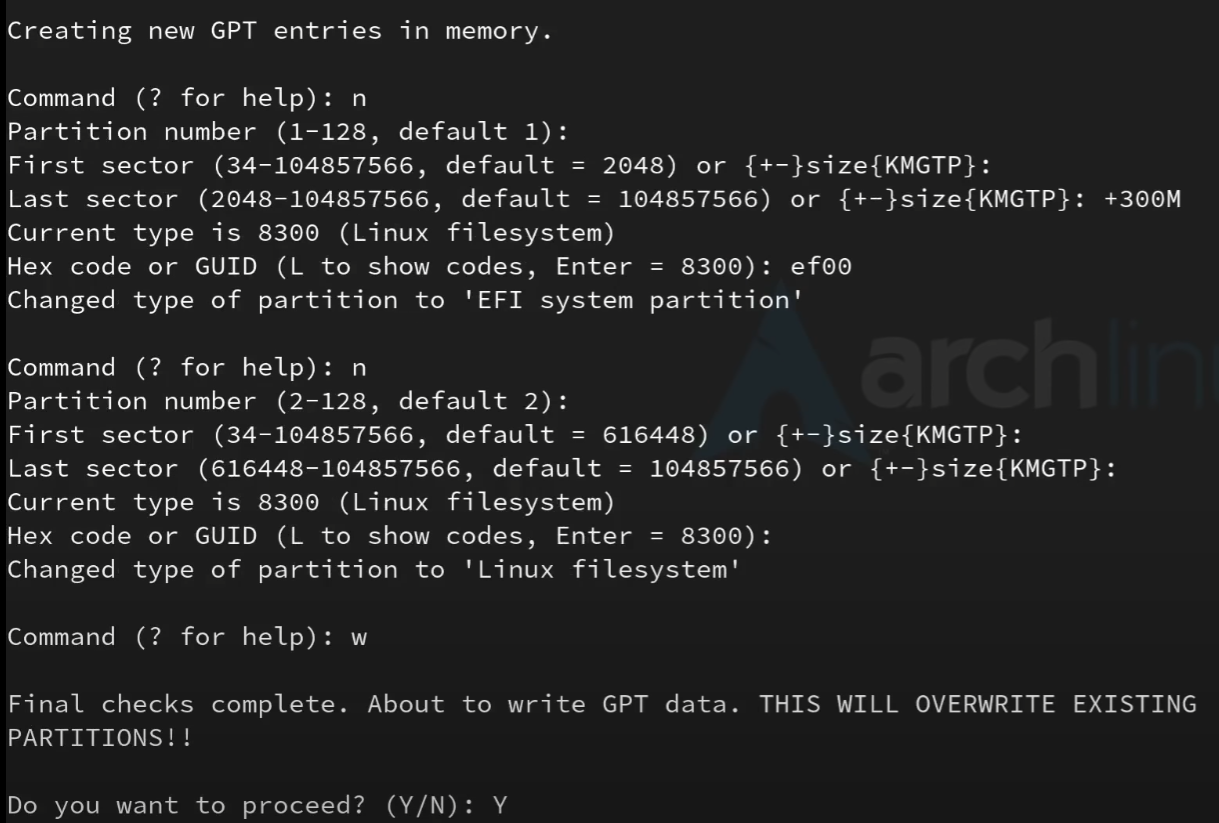


1. Let’s do the disk partition with   
   # **gdisk <the path of the disk>** and enter “n” for new.

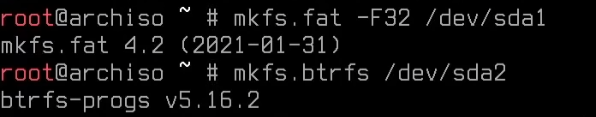


1. This is just an example of how to do your disk partition:

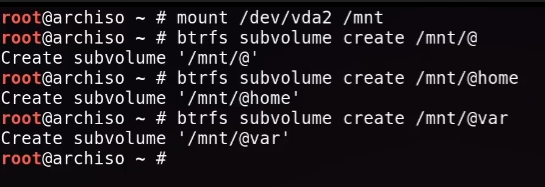
I am doing the partition on a brand-new disk, the first partition I am creating is for EFI (350Mb), the second will have the rest of the disk, because we will use ZRam for the swap and BTRFS with multiple sub-volumes:



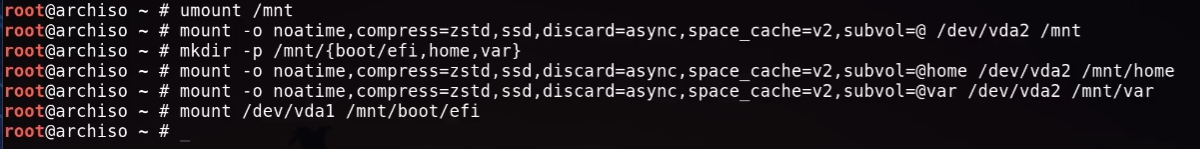
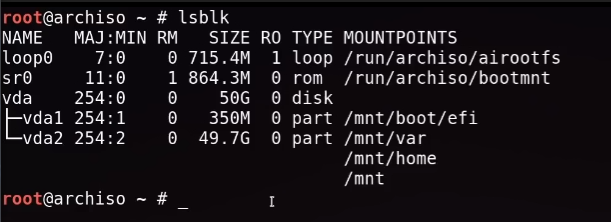
1. The next step is formatting the disk partitions (for this example, vda1 is the EFI partition, vda2 is the BTRFS file system):

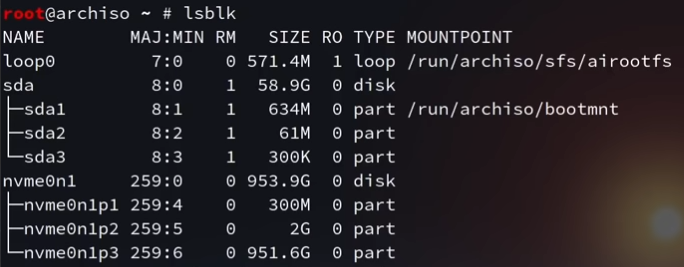


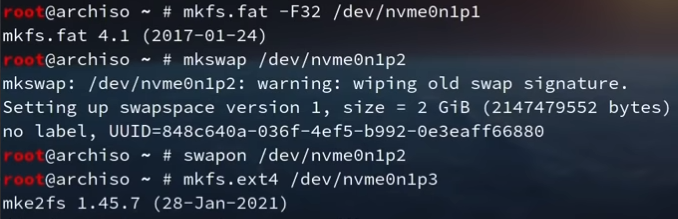
1. Next we need to mount our partitions, first we need to mount the root partition, then create our BTRFS sub-volumes and mount them with specific options (we will take snapshots only of the root sub-volume), as always, feel free to change this according to your needs:

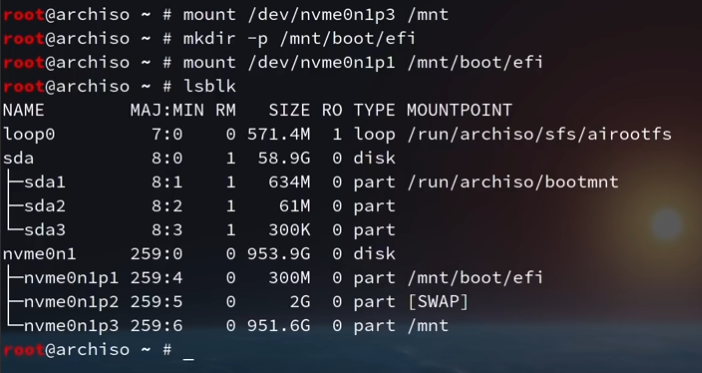
  
**# mount /dev/vda2 /mnt  
# btrfs subvolume create /mnt/@  
# btrfs subvolume create /mnt/@home  
# btrfs subvolume create /mnt/@var ??**

1. Next, we need to unmount our /mnt directory because later we need to re-mount the sub-volumes into their respective directories using some options:

  
**# umount /mnt  
# mount -o noatime,compress=zstd,ssd,discard=async,space\_cache=v2,subvol=@ /dev/vda2 /mnt  
# mkdir -p /mnt/{boot/efi,home,var}  
# mount -o noatime,compress=zstd,ssd,discard=async,space\_cache=v2,subvol=@home /dev/vda2 /mnt/home  
# mount -o noatime,compress=zstd,ssd,discard=async,space\_cache=v2,subvol=@var /dev/vda2 /mnt/var ??  
# mount /dev/vda1 /mnt/boot/efi**  
  
  
**# lsblk**  


As an alternative, we can use the classic ext4 file system, if you use this alternative, you will have to skip/ignore all the configuration related to snapshots and everything related to BTRFS that is present on this document below, for this example we are using a disk called nvme0n1:  


Formatting the partitions:  


Let’s mount these partitions:  


## 05 Installation of the base packages on the new system:

**# pacstrap /mnt base linux linux-firmware vim amd-ucode btrfs-progs**  
**06 Generate the file system table**

**# genfstab -U /mnt >> /mnt/etc/fstab  
# cat /mnt/etc/fstab**

**07 Moving to the installation as root**

**# arch-chroot /mnt**

**08 Determine and set the timezone**

1. **# timedatectl list-timezones | grep US**
2. **# ln -sf /usr/share/zoneinfo/US/Central /etc/localtime**
3. **# hwclock --systohc**

**09 Work on the locale**

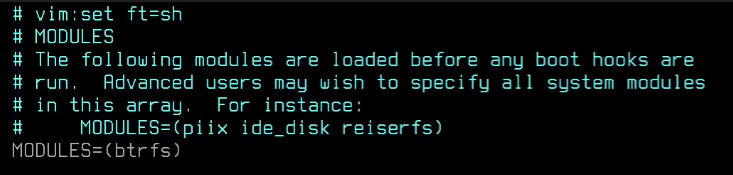
1. **# vim /etc/locale.gen**
2. Uncomment the line for English US (or whatever you need), then, save the file and exit VIM: **# en\_US.UTF-8 UTF-8**
3. Generate the locale:  
   **# locale-gen**
4. Create the locale file with this information:  
   **# echo “LANG=en\_US.UTF-8” >> /etc/locale.conf  
   # echo “KEYMAP=us” >> /etc/vconsole.conf** (optional)

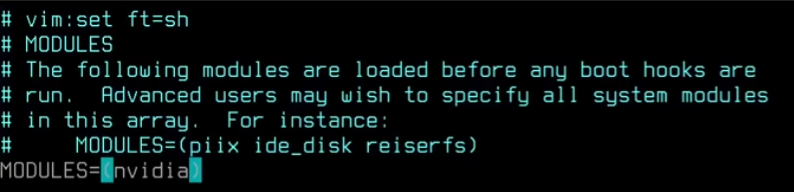
**10 Work on the hostname & hosts files**

1. **# echo “arch” >> /etc/hostname**
2. Also, you need to set the hosts file: **# vim /etc/hosts**
3. Then, you enter the following, then, save the file and exit VIM (:x) **127.0.0.1 localhost  
   ::1 localhost  
   127.0.1.1 arch.localdomain arch**
4. You can set the root password if not done already:  
   **# passwd**

**11 Install the bootloader and some other packages**

**# pacman -S grub efibootmgr networkmanager network-manager-applet dialog os-prober wpa\_supplicant snapper mtools dosfstools base-devel linux-headers bluez bluez-utils cups reflector openssh git xdg-utils xdg-user-dirs alsa-utils pulseaudio pulseaudio-bluetooth inetutils bash-completion terminus-font rsync dnsutils nfs-utils nvidia nvidia-utils nvidia-settings *xf86-video-amdgpu sof-firmware ntfs-3g***

1. Before installing the bootloader, we need to change something into the cpio file because we are using the BTRFS file system:  
   **# vim /etc/mkinitcpio.conf**   
     
     
   \*\* For some other configurations, depending on the hardware you have, if you have an NVidia card you would need:



1. Then, save the file and exit VIM.
2. Then, re-generate the kernel image with the BTRFS module included:  
   **# mkinitcpio -p linux**

**12 Install the GRUB loader and generate the GRUB config file**

**# grub-install --target=x86\_64-efi --efi-directory=/boot/efi --bootloader-id=GRUB --removable** (and make sure to enable UEFI mode from bios, otherwise it will not boot).

If we have other operative system installed already, and we need to boot from multiple OSs, we need to tell GRUB to “do not disable” os-prober executing the following steps to update the configuration file for GRUB:  
**# vim /etc/default/grub**  
  
If not present, add this entry at the end of the file, save the file and exit VIM::  
**GRUB\_DISABLE\_OS\_PROBER=false**  
  
Lastly, we run the GRUB configuration file  
**# grub-mkconfig -o /boot/grub/grub.cfg**

**13 Enable some services**

**# systemctl enable NetworkManager  
# systemctl enable sshd  
# systemctl enable cups  
# systemctl enable bluetooth  
# systemctl enable fstrim.timer**  
**14 Add another user**

**# useradd -mG wheel <username>  
# passwd <username>**

**15 Give <username> sudo privileges**

**# EDITOR=vim visudo**

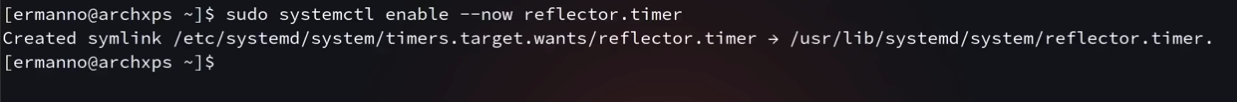
1. Go down until you find the wheel group and uncomment this line, later, save the file and exit VIM:  
   **%wheel ALL=(ALL) ALL**

**16 Exit the installation and return to the installer**

**# exit (to go back to the ISO)  
# umount -a  
# reboot**

## 17 Login with your user and install the XFCE desktop environment

1. If needed, we can connect to the Wi-Fi using the Network Manager terminal user interface, where you will be able to activate a connection and enter the password for the same:  
   **$ nmtui**
2. Just to be safe, please repeat steps 2, 3, 8.3
3. Let’s enable the reflector timer:



1. Let’s proceed to install all the packages we need, first the graphics driver:  
   **$ sudo pacman -S xorg lightdm lightdm-gtk-greeter xfce4 xfce4-goodies xfce4-terminal firefox-developer-edition materia-gtk-theme papirus-icon-theme simplescreenrecorder vlc dina-font tamsyn-font ttf-bitstream-vera ttf-croscore ttf-dejavu ttf-droid gnu-free-fonts ttf-ibm-plex ttf-liberation ttf-linux-libertine noto-fonts ttf-roboto tex-gyre-fonts ttf-anonymous-pro ttf-cascadia-code adobe-source-code-pro-fonts cantarell-fonts ttf-opensans adobe-source-han-sans-otc-fonts adobe-source-han-serif-otc-fonts lolcat archlinux-wallpaper xreader**
2. Enable the lightdm display manager and reboot: **$ sudo systemctl enable lightdm**
3. Configure the lightdm-slick-greeter  
   **$ sudo vim /etc/lightdm/lightdm.conf**  
   Search for this line and make the corresponding fix, so we have:  
   **$ greeter-session=lightdm-slick-greeter**
4. Installing PARU AUR Helper  
   **$ sudo pacman -S --needed base-devel  
   $ git clone** [**https://aur.archlinux.org/paru.git**](https://aur.archlinux.org/paru.git) **$ cd paru  
   $ makepkg -si**
5. Installing timeshift and zramd  
   **$ paru -S timeshift timeshift-autosnap zramd lightdm-settings**Configure timeshift:  
   **$ sudo timeshift-gtk**Enable zramd:  
   **$ sudo systemctl enable --now zramd.service  
   $ lsblk** (just to make sure is active and running)
6. **$ reboot**
7. Finally, the lightdm display manager should display the login screen, after entering your user and password you should be able to see the XFCE4
8. Anytime that you need to review the hardware specs of your PC, you can do it with:  
   **$ lspci**

\*\* NOTICE: I cannot and will not be liable responsible for any lost of data or information, nor any damage to your hardware. THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. \*\*