An7ar35

December 2017
(Updated 08/11/2019)

Arch Linux installation guide for MacBook Pro Retina mid 2015 model

with KDE

v1.0.6

Contents

1 Foreword

This guide is current for the Linux kernel version 4.15.5-1.

After lots of reading, searching, experimenting, furious late-night shell command typing and do-overs here are the results of replacing OSX with Arch Linux on a mid-2015 MacBook Pro.

The setup is:

- Clean Arch installation, no OSX.
- Encrypted root/home/swap partition (luks),
- GRUB boot-loader (EFI),
- KDE Plasma desktop environment.

I would also advise backing up your drive using a complete bit-to-bit cloning process (unlike me who casually forgot that step... 😱) so that you retain a copy of everything including the recovery partition on the Mac. This way it is a one-step process to restore everything.

This guide assumes a working knowledge of the Linux command line (bash) as well as Arch's `pacman` and `yaourt`$^{\text{AUR}}$ package managers.

2 Nomenclature

`package-name`	Standard package repository - use `pacman` to install.
`package-name`$^{\text{AUR}}$	Arch User Repository package (AUR) - use `yaourt` to install.
$ `...`	Command to type in at the command line prompt.
$TEXT	User defined variable (replace with what you want to define it as).
Fn	Function key.
⌃	Ctrl key.
⌥	Alt/Options key.
⌘	Command (Cmd) key.
⇧	Shift key.

3 Machine Specs

Display	· 15.4” LED-backlit Retina display (2880x1800 at 220ppi)
Processor	· 2.5GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.7GHz) with 6MB shared L3 cache
RAM	· 16GB of 1600MHz DDR3L memory
GPU	· Intel Iris Pro Graphics, · AMD Radeon R9 M370X with 2GB of GDDR5 memory
Storage	· 512GB PCIe-based flash
Webcam	· 720p FaceTime HD camera
Network	· 802.11ac Wi▯Fi wireless networking; IEEE 802.11a/b/g/n compatible, · Bluetooth 4.0 wireless technology

4 Results

Legend: ✅ Works out-of-the-box or with few steps, ☑️ Requires some work, ⚠ Not fully working, ❌ Broken

Item	**Result**	**Note**
Wifi[1]	✅	Works out-of-the-box, some error messages.
Bluetooth	✅	Uses standard `bluez` package.
Display (IPS scaling)[2]	⚠	Mixed results without Wayland
AMD Radeon R9 M370X	✅	Open source driver works like a charm, screen brightness control works now[3].
Intel Iris Pro Graphics[4]	⚠	Disabled by default, screen brightness control not working.
Graphic switching	⚠	Dynamic switching doesn’t work. Manual switching requires reboot and some TLC[5].
Audio	✅	Uses ALSA.
FaceTimeHD camera[6]	⚠	Requires firmware and driver from AUR. Suspend issues.
Keyboard	✅	ISO layout in command line but perfect in KDE.
Keyboard (Backlight)	✅	Works out-of-the-box.
Keyboard Fn keys	✅	Works out-of-the-box in KDE.

[1] Error messages with default kernel module but doesn’t seem to affect connections.

[2] A workable solution can be setting the resolution to 1920x1200. The scale is good for a 15” screen.

[3] Kernels older than 4.15.5-1 require `apple-gmux` to be patched.

[4] When enabled, there are problems with back-lighting.

[5] Pain in the arse and not streamlined at all. Backlight control on Intel’s IGU breaks.

[6] Not all applications seem to work with this camera but it might just be compatibility problems on the app side.

5 Pre-Installation

5.1 Preparations

Backup drive (complete drive clone preferable).

Make a copy of the colour profile file(s) on the mac to a USB stick. It will be useful later on Linux. The profiles are located in `/Library/ColorSync/Profiles/Displays/*`.

5.2 Making a bootable USB

5.2.1 From Linux

```
$ dd if=archlinux.iso of=/dev/sdX bs=16M && sync
```

> Where `X` is your target USB drive letter (use `lsblk` for an overview of all connected drives to find out)

5.2.2 From a Mac

```
$ diskutil unmountDisk diskX
```

> Where `X` is your target USB drive number (use `diskutil list` for an overview of all connected drives to find out).

```
$ sudo dd if=/Users/$USERNAME/Downloads/archlinux.iso of=/dev/diskX bs=16M
```

> Replace $USERNAME with your user-name on the mac and replace `X` with the USB drive's number.

6 Installation

6.1 Booting from the USB stick

Simply plug in the USB in the MBP and press the ⌥ key during start up to reach the boot menu.

6.2 Preliminary setup

```
$ loadkeys uk
```

> Load your keyboard layout. Replace 'uk' with whichever you have on your machine.

```
$ wifi-menu
```

> Connect up to the wifi network. On the MPB Pro 2015 the wifi gets detected and works out of the box at this stage.

6.3 Partitioning and Crypto volume setup

The assumption here is that the hard drive where everything will be installed to is `/dev/sda`. The partition structure will thus be:
`/dev/sda1` for the EFI partition,
`/dev/sda2` for the Boot partition and
`/dev/sda3` for the encrypted volume where the root, home and swap will be located.

```
$ cgdisk /dev/sda
```

> **Partition layout with just Linux**
> *Warning: this will erase every thing including the recovery partition for OSX.*
> *No easy way to go back after this without a bit-to-bit HDD clone*
>
> 1. 100M partition 'EFI' (Hex #`ef00`)
> 2. 250M partition 'Boot' (Hex #`8300`)
> 3. 100% remainder of the space for the crypto volume (Hex #`8300`)
>
> The root and home partition will be created later in the crypto volume

```
$ cryptsetup --verbose --verify-passphrase --cipher aes-xts-plain64 --key-size
  512 --hash sha512 --iter-time 5000 --use-random luksFormat /dev/sda3
```

> Sets up the crypto volume.

```
$ cryptsetup open --type luks /dev/sda3 mctoasty
```

> `mctoasty` is the mounted name of the crypto volume. Can be changed to whatever is preferred.

6.3.1 Create crypto volume partitions

```
$ pvcreate /dev/mapper/mctoasty
$ vgcreate vg0 /dev/mapper/mctoasty
$ lvcreate --size 16G vg0 --name swap
```

> Generally swap size value is set to the amount of RAM.

```
$ lvcreate --size 50GB vg0 --name root
```

> Adjust root's size as required.

```
$ lvcreate -l +100%FREE vg0 --name home
```

> Creates the home parition with the remainder of the free space.

6.3.2 Formatting all the partitions

```
$ mkfs.vfat -F32 /dev/sda1
```

> EFI partition

```
$ mkfs.ext2 /dev/sda2
```

> Boot partition

```
$ mkswap /dev/mapper/vg0-swap
$ mkfs.ext4 /dev/mapper/vg0-root
$ mkfs.ext4 /dev/mapper/vg0-home
```

6.3.3 Mount all the partitions

```
$ mount /dev/mapper/vg0-root /mnt
$ swapon /dev/mapper/vg0-swap
$ mkdir /mnt/boot
$ mount /dev/sda2 /mnt/boot
$ mkdir /mnt/boot/efi
$ mount /dev/sda1 /mnt/boot/efi
$ mkdir /mnt/home
$ mount /dev/mapper/vg0-home /mnt/home
```

7 System Setup I: Base System Installation

```
$ pacstrap /mnt base base-devel grub-efi-x86_64 git efibootmgr bash-completion
  dialog wpa_supplicant dosfstools mtools
```

> Base packages along with GRUB and the EFI boot manager, git (will be useful later), bash completion and the stuff needed to keep the Wifi device in working order after reboot.

7.1 Generating fstab

```
$ genfstab -pU /mnt > /mnt/etc/fstab
```

> **Important!**
> This generates the fstab. i.e.: It saves our mounted partitions and swap configuration for persistence after a reboot. If you miss this step you'll have to reboot with the USB, mount

> everything again and then generate the fstab file.

```
$ arch-chroot /mnt /bin/bash
```

7.2 System clock

```
$ ln -s /usr/share/zoneinfo/$ZONE/$REGION /etc/localtime
```

> Replace $ZONE with yours from `/usr/share/zoneinfo/`
> Replace $REGION with yours from `/usr/share/zoneinfo/$ZONE/`
> e.g.: `/usr/share/zoneinfo/Europe/London`

```
$ hwclock --systohc --utc
```

7.3 Hostname

```
$ echo $HOSTNAME > /etc/hostname
$ nano /etc/hosts
```

> Add the hostname at the end of each of the relevant lines in this file for completion's sake.
> Replace $HOSTNAME by the hostname you want to computer to have.

7.4 Basic fonts

```
$ pacman -S terminus-font ttf-dejavu ttf-liberation
```

> The terminus font will be used to make the console font more readable on the HiDPI display.

7.5 Locale

```
$ locale-gen
```

> Generates the locale file

```
$ nano /etc/local.gen
```

> Edit the locale configuration and delete the hash in front of the desired locale.

```
$ locale-gen
```

> Needs to run again to apply the changes made in the previous step.

```
$ nano /etc/locale.conf
```

> To set permanent locale settings add the lines:
>
> ```
> LANG=en_GB.UTF-8
> LANGUAGE=en_GB
> LC_ALL=C
> ```

7.6 Console keymap and font

To see a list of all available keymaps: `find /usr/share/kbd/keymaps/ -type f | more`
To see a list of all installed console fonts: `ls /usr/share/kbd/consolefonts/`
For font maps check the Arch Wiki and the Wikipedia entries.

```
$ nano /etc/vconsole.conf
```

> To set permanent setting for the console add the lines:
>
> ```
> KEYMAP=uk
> FONT=ter-228n
> FONT_MAP=8859-1
> ```

7.7 Root/User Accounts

```
$ passwd
```

> Sets the root password.

```
$ useradd -m -g users -G wheel -s /bin/bash $USERNAME
```

> Adds a user. Replace $USERNAME with whatever user name you which to use.

```
$ passwd $USERNAME
```

> Sets the user's password.

7.7.1 Sudoers

```
$ groupadd sudo
$ usermod -a -G sudo $USERNAME
```

> Creates 'sudo' group then adds $USERNAME to it.

Make sure to un-comment the `%sudo ALL=(ALL) ALL` line in `/etc/sudoers`:

```
## Uncomment to allow members of group sudo to execute any command
# %sudo ALL=(ALL) ALL
```

7.8 Initial RAM Environment Configuration

`mkinitcpio` is useful to create the initial ramdisk environment (early userspace).

```
$ nano /etc/mkinitcpio.conf
```

> In the file do the following:
>
> - In ’MODULES:
>   1. Add `ext4`, `amdgpu`[7], `radeon`[7] and `bcm5974`[8]
>
> MODULES should be ordered as such in the end:
> `MODULES=(ext4 amdgpu radeon bcm5974)`
>
> - In ’HOOKS’:
>   1. Add `encrypt` and `lvm2` before `filesystems`
>   2. Add `consolefont` right after `autodetect`
>   (to avoid squinting at the crypto volume password prompt)
>   3. Move `keyboard` right after `consolefont`
>
> HOOKS should be ordered as such in the end:
> `HOOKS=(base udev autodetect consolefont keyboard modconf block encrypt lvm2 filesystems fsck)`

```
$ mkinitcpio -p linux
```

> Regenerates the initrd image.

7.9 GRUB boot loader

```
$ grub-install
$ nano /etc/default/grub
```

> At line with `GRUB_CMDLINE_LINUX` add the arguments so that it becomes
> `GRUB_CMDLINE_LINUX="cryptdevice=/dev/sda3:luks:allow-discards"`

```
$ grub-mkconfig -o /boot/grub/grub.cfg
```

7.10 Dismount and reboot

```
$ umount -R /mnt
$ swapoff -a
```

Remove the USB stick then `reboot`.

[7] Graphic modules to avoid the SDDM launch failure.

[8] Trackpad module for the MBP. It avoids the annoying issue with KDE ‘forgetting’ the user settings for it between boots.

Note: If you want a break (☕) this is the place to take it. Instead of rebooting just shutdown the machine.

8 System Setup II: Hardware and Tools

Reconnect to your wifi with `sudo wifi-menu`

8.1 Enabling the multilib package repository

```
$ nano /etc/pacman.conf
```

> Uncomment the multilib lines (i.e. remove the leading `#`):
>
> ```
> #[multilib]
> #Include = /etc/pacman.d/mirrorlist
> ```

```
$ pacman -Syuu
```

> Updates the cache/packages

8.2 AUR package manager - Yaourt

There are two ways of doing this.

a. Through Pacman

```
$ sudo nano /etc/pacman.conf
```

> Add the folowing lines to the end of the file:
>
> ```
> [archlinuxfr]
> SigLevel = Never
> Server = http://repo.archlinux.fr/$arch
> ```

```
$ sudo pacman -Syuu
$ sudo pacman -S yaourt
```

b. Manual installation

From your home directory (`cd ~`):

```
$ mkdir -p git-repos/system-packages
$ cd git-repos/system-packages
```

> Or whatever directory chosen to clone the repositories into.

```
$ git clone https://aur.archlinux.org/package-query.git
```

> Required for Yaourt

```
$ git clone https://aur.archlinux.org/yaourt.git
```

For each of the 2 packages, `cd` into their respective directories and run the following command to install:

```
$ makepkg -si
```

> e.g.: `cd package-query` then `makepkg -si`

Finally

To build AUR packages with all 4 cores of the processor we need to modify the `makepkg.conf` file.

```
$ sudo nano /etc/makepkg.conf
```

> Find the line:
>
> ```
> #MAKEFLAGS="-j2"
> ```
>
> Uncomment the line and change the flag to `"-j4"`:
>
> ```
> MAKEFLAGS="-j4"
> ```

8.3 Console-Fu

```
$ yaourt -S hstr-git
```

> This is a replacement on steroids for the console's ^ + R

```
$ hh --show-configuration > ~/.bashrc
```

> Adds the config options for HSTR to your bash profile and auto-starts hh on login.

```
$ pacman -S powertop htop
```

> Installs 2 of the most basic and useful monitoring tools in linux.

8.4 Power management

```
$ yaourt -S laptop-mode-tools
```

> All the laptop-centric power saving goodies

```
$ sudo systemctl enable laptop-mode.service
```

> Turns on the service

```
$ pacman -S acpid
```

> *(Optional)* Daemon for delivering ACPI events.

8.5 Fans

```
$ yaourt -S mbpfan-git
```

> mbpfan-git[AUR] controls the MacBook's fans.

```
$ sudo systemctl enable mbpfan.service
```

> Turns on the service

The configuration file is located in `/etc/mbpfan.conf`. Go there to customise the temperature limits and fan speeds.

8.6 Processor

```
$ pacman -S intel-ucode
```

> For intel's microcode update.

```
$ grub-mkconfig -o /boot/grub/grub.cfg
```

> Need to update grub so that it loads the Microcode updates at boot

8.6.1 Turbo Boost

I would recommend disabling this feature if usage includes sustained load-heavy computation such as rendering, compiling, etc...

Turbo boost automatically increases the operating frequency of the cores depending on the task load. This allows for greater performance under demanding conditions but also causes heating issues. These are exasperated by the design constraint of laptops where large heat-sinks + fans or water-cooling is not practical.

Turning off Intel's CPU turbo boost makes a sizeable difference to temperature[1] both on idle and under load. A cooler processor will also help increase the lifespan of the machine.

The table below shows the different temperature reached with and without the turbo boost enabled on my machine. The load was from compiling a small C++/Qt application. The maximum turbo boost temperature was reached 10 seconds inside the compiling job and stayed at $100\,^{\circ}$C until completion

[1] Odly enough, I found that on my MBP the temperatures reached on OSX are a little worse than on Linux.

with the fans going full blast.

Turbo Boost state	Idle Temp.	Load Temp.
Enabled	60 → 65 °C	100 °C
Disabled	48 → 50 °C	76 °C

Requirements

MSR-Tools$^{\text{AUR}}$ provides utilities to access the processor MSRs and CPU ID directly. It is called by the turbo-boost disabler script.

```
$ yaourt -S msr-tools
```

Disabling Turbo Boost

Clone the arch-scripts repository into the `~/git-repos/system-packages` directory previously created.

```
$ cd ~/git-repos/system-packages
$ git clone https://github.com/An7ar35/arch-scripts
$ cd arch-scripts/coreboost/
$ sudo ./install.sh
```

> The installation copies the `coreboost.sh` script into `/usr/local/bin/` and the systemd service file to launch that script into `/etc/systemd/system/`.
> It then enables the service and loads it up. The turbo-boost disabler will be called upon at boot and after resuming suspend automatically.

8.7 Sound

Alsa works without issues.

```
$ sudo pacman -S alsa-utils alsa-plugins
$ alsamixer
```

> Make sure your current sound card is the ”HDA Intel PCH” and that your master volume is up and unmuted (mute=MM, unmuted=00 at the bottom of the volume bar. You can use the M key on the keyboard to toggle mute).

```
$ speaker-test -c 2
```

> To make sure the sound works.

Note: The internal speaker might not be disabled when using the headphone jack. To solve this, enable ”Auto-mute” in `alsamixer`.

8.7.1 Audio recording

For recording audio (e.g.: music instruments) or using real time processors like Guitarix on your input a real-time kernel would be advised.

realtime-privileges should be installed in any case and users that require it need to be added to the `realtime` group..

```
$ sudo pacman -S realtime-privileges
$ usermod -a -G realtime $USERNAME
```

8.8 Bluetooth

Bluetooth works out-of-the-box with the standard packages.

```
$ sudo pacman -S bluez bluez-utils
$ modprobe btusb
$ sudo systemctl enable bluetooth.service
```

8.9 Wifi

Broadcom Limited BCM43602 802.11ac Wireless LAN SoC (rev 01)

As with most Broadcom chipsets, there are going to be hiccups. The `brcmfmac` driver is automatically loaded and works for the most parts (lucky!). Some have reported connectivity issues with this driver. Here is a run-down that includes my experience:

- ✅ 2.4Ghz channels: works without issues.
- ✅ 5Ghz channels: both 2.4Hgz and 5Ghz capabilities are listed in `iw list`. Reports say that they are not detected but that's for *rev 2* of the chipset..
- ⚠ Low sensitivity: Lower than in OSX but not as bad as some reports.
- ✅ Failure after 10-15mns has not occurred in my case. Might be a *rev 2* only issue.

8.10 Video

8.10.1 AMD Radeon

Using the dedicated GPU means that the battery will drain faster than when using Intel's IGU but getting the backlight control to work has been relatively pain free.

```
$ sudo pacman -S mesa xf86-video-amdgpu vulkan-radeon lib32-mesa
```

> Installs the open source drivers for the ATI GPU along with the 32bit libs and the Vulkan drivers.
> The `lm_sensors` package used for temperature monitoring is a dependency for `mesa` so will be installed with it.
> Run `sensors` to see all the temperatures.

```
$ yaourt -S radeontop
```

> *(Optional)* Monitoring utility for Radeon GPU cards.

8.10.2 Intel's IGU

Getting Intel's IGU requires some work and is not perfect. First, the drivers:

```
$ sudo pacman -S xf86-video-intel
```

Enabling the Intel IGU

We need to get the apple_set_os.efi binary from the repository's release section and unzip it. Then we need to copy it to the EFI partition:

```
$ sudo mkdir /boot/efi/EFI/custom
$ sudo cp apple_set_os.efi /boot/efi/EFI/custom
```

Then, `apple_set_os.efi` needs to be placed in GRUB's bootloader chain. I decided for a menu item in order to make triggering `apple_set_os.efi` optional.

```
$ lsblk --output MOUNTPOINT,LABEL,UUID
```

> Look for `/boot/efi` and **make a note of its UUID**.

```
$ sudo nano /etc/grub.d/40_custom
```

> Add these lines to the file replacing $UUID with the UUID noted previously:
>
> ```
> menuentry {
>     search --no-floppy --set=root --fs-uuid $UUID
>     chainloader (${root})/EFI/custom/apple_set_os.efi
> }
> ```

```
$ sudo grub-mkconfig -o /boot/grub/grub.cfg
```

> Updates GRUB with the changes.

8.10.3 Switching GPUs

This section assumes both the AMD and Intel installations have been done. An extra package (gpu-switch^{AUR}) is required to be able to switch between the integrated and dedicated GPUs on the MBP 11,5.

```
$ yaourt -S gpu-switch
```

Switching to the integrated GPU (intel):

```
$ sudo gpu-switch -i
```

Then use the "*apple_set_os*" menu option in GRUB at boot before your normal "*Arch*" menu option.

Switching to the dedicated GPU (ATi):

```
$ sudo gpu-switch -d
```

Then **only** use your normal "*Arch*" menu option in GRUB at boot (default).

8.10.4 Display brightness

Note: this only applies to the AMD Radeon on kernels **older than 4.15.5-1**. Brightness control on these does not work on the Intel IGU aside from turning it either on/off completely.

To get brightness control to work on these older kernels, a patched version of the `apple-gmux` kernel module is required. The vanilla module does not work for the current Linux kernel. Perhaps later versions will eventually.

An easy installer script is available in the arch-script repository.

If you haven't previously cloned the repository (Turbo Boost section):

```
$ cd ~/git-repos/system-packages
$ git clone https://github.com/An7ar35/arch-scripts
```

Installing the patched module

```
$ cd arch-scripts/mbp-brightness-patch/
$ sudo ./install.sh
```

Then restart the machine.

Kernel updates

If the brightness control breaks after a kernel update the patch installer must be run again to get the functionality back.

8.11 Keyboard

All of these tweaks are optional and based on personal choice. The one's I have not tested are marked as such.

Switch function keys on by default

To make function keys be used as first keys. i.e.: Pressing F1 key alone will behave like F1 and pressing Fn + F1 will act as the special key (brightness down).

```
$ sudo nano /etc/modprobe.d/hid_apple.conf
```

> Add the following line to the file:
>
> ```
> options hid_apple fnmode=2
> ```

```
$ sudo mkinitcpio -p linux
```

> Updates the *initramfs* with the new configuration.

Swap the ⌥ and ⌘ keys (untested)

```
$ sudo nano /etc/modprobe.d/hid_apple.conf
```

> Add the following line to the file:
>
> ```
> options hid_apple swap_opt_cmd=1
> ```

```
$ sudo mkinitcpio -p linux
```

> Updates the *initramfs* with the new configuration.

8.12 Webcam

The 2015 MBP has Facetime HD. Fortunately there is a reversed-engineered driver but ”PC suspension is not supported if a program that is keeping the camera active is running”.

To make it work, first install the firmware then the driver from AUR:

```
$ yaourt -S facetimehd-firmware
$ yaourt -S bcwc-pcie-git
```

To test the webcam, `mplayer tv://` or the ‘*Qt 4VL Test Utility*’ can be used inside KDE.

Tested applications

App	Version	Works?
Discord	0.0.3-1	✓
Skype	8.13.76.8-1	✓
Kamoso	3.2.4-1	✗

8.13 Touchpad

Basic touchpad support is available with the Linux kernel. If you want to customise your experience, a specialised synaptic driver must be installed. There are 3 options to choose from. Make sure you only have one installed at any one time to avoid conflicts.

Option 1: `libinput`

```
$ sudo pacman -S libinput
```

To add gestures, the `libinput-gestures`$^{\text{AUR}}$ package can be installed:

```
$ yaourt -S libinput-gestures
$ sudo gpasswd -a $USERNAME imput
```

> Adds $USERNAME to the 'input' group.

```
$ libinput-gesture-setup autostart
```

The Arch Wiki states that there may be an issue where settings do not take effect from the KDE vanilla '*System Settings*' touchpad panel.

Optionally, a rewritten KCM for all for all input devices supported by `libinput` is available from the AUR repository:

```
$ yaourt -S kcm-pointing-devices-git
```

Option 2: `xf86-input-synaptics`

The standard synaptic driver. KDE will detect it and enable you to change its settings within the 'System Settings' panel. Only supports 2 finger gesture though.

```
$ sudo pacman -S xf86-input-synaptics
```

Option 3: `xf86-input-mtrack`$^{\text{AUR}}$

Multi-touch support but KDE does **not** natively detect it so settings must be set manually in `/etc/X11/xorg.conf.d/10-mtrack.conf`. A full list of supported options is available in the project's Github repository.

```
$ yaourt -S xf86-input-mtrack
```

8.14 Other Drivers

Possible missing firmware module for:

- wd719x
- aic94xx

```
$ pacman -S aic94xx-firmware
$ pacman -S wd719x-firmware
$ sudo mkinitcpio -p linux
```

> To make sure drivers are loaded.

9 System Setup III: Desktop

Wayland support as of Dec 2017 on KDE is beta at best. Scaling in nice but buggy as hell displaying artefacts. Until that gets a lot better Xorg is the default choice as display managers go even if HiDPI support is terrible.

9.1 Display server (Xorg)

```
$ sudo pacman -S xorg, xorg-server xorg-xinit
```

9.2 Desktop Environment (KDE)

A display manager (SDDM) is installed along with KDE to make it easy to automatically run KDE when starting a user session and provide a GUI login too.

```
$ sudo pacman -S plasma kde-applications sddm systemd-kcm
$ sudo systemctl enable sddm.service
$ sudo systemctl start sddm.service
```

Auto-login can be configured inside KDE's 'System Settings' panel ('Startup & Shutdown' → 'Login Screen (SDDM)' → 'Advanced' tab).

```
$ sudo pacman -S xdg-user-dirs
```

> **Optional**
> Create all the default user directories such as 'Downloads' , 'Music' , etc...

9.3 Network manager (Wifi)

```
$ sudo systemctl enable NetworkManager.service
```

> Enables the network service. GUI network connections should work after reboot.

9.4 GUI package manager

Pamac is a nice package manager with AUR support. It can be installed with:

```
$ yaourt -S pamac-aur
```

> AUR support can be explicitly enabled inside the application's settings in the AUR tab.

9.5 Colour Profile

To use the previously saved OSX colour profile(s) copy them/it to your home directory under `.color/icc/`.

If you don't plan on using KDE and/or want an agnostic way of loading colour profiles then install `xcalib`AUR and follow the instructions provided in the Arch Wiki.

For KDE

```
$ sudo pacman -S colord-kde
```

> Then, in 'Systems Settings' under the 'Color Corrections' section, add your profile file wanted to the '*Apple - MacBookPro11,5*' and tick it instead of the pre-given default.

9.6 Peripherals

These are generic package installations required to make most printers and scanners work. Additional steps may (very likely!) be required to get these peripherals working in Linux.

9.6.1 Printing

```
$ sudo pacman -S cups print-manager
$ systemctl enable org.cups.cupsd.service
$ systemctl start org.cups.cupsd.service
```

> Start it right away.

9.6.2 Scanning

```
$ sudo pacman -S sane libksane sane-frontends xsane
$ sudo pacman -S skanlite
```

> Simple image scanning application for KDE.

If you have an Epson scanner:

```
$ sudo pacman -S iscan iscan-data
```

10 System Setup IV: Applications

10.1 File system

```
$ sudo pacman -S ntfs-3g exfat-utils
```

> Enables access to NTFS and ExFAT file systems.

```
$ sudo pacman -S gparted
```

> Partition manager. Alternatively there is also `partitionmanager` (KDE)

10.2 Firewall

Install `firewalld` and start the `firewalld.service`

If there is a systemd timeout issue on restart, do:

```
$ sudo nano /etc/firewalld/firewalld.conf
```

> Find the `CleanupOnExit` option in the file and set it to '`no`':
>
> ```
> CleanupOnExit=no
> ```

10.3 Compression, Archiving and Backup

Terminal

- `p7zip`
- `unrar`
- `zip`
- `unzip`

GUI

- `ark` (Compression)
- `deja-dup` (Backup)

10.4 Dolphin extras

```
$ sudo pacman -S kde-thumbnailer-odf kde-thumbnailer-epub
  kde-thumbnailer-gimpsources
```

> Adds thumbnail support to ODF, EPUB and gimp source files. Enable in Dolphin's settings.

10.5 Fonts

Some more fonts for the system. If more are required, a good place to find some is the Font Library which has a large collection of fonts with permissive licenses.

- `adobe-source-code-pro`
- `adobe-source-sans-pro`
- `adobe-source-serif-pro`
- `ttf-mac-fonts`$^{\text{AUR}}$

10.6 Nice extra packages

Package	Environment	Description
`cmus`	Term	Nice little console music player.
`wget`	Term	Command line tool to retrieve files using HTTP, HTTPS, FTP and FTPS.
`yakuake`	KDE	Quake-like tidle console.
`cairo-dock` `cairo-dock-plugins`	GUI	Docks like on a mac.
`redshift` `plasma5-applets-redshift-control`	GUI/KDE	Colour temperature adjuster and KDE controller applet.
`firefox`	GUI	Web browser.
`thunderbird`	GUI	Mail client.

11 Troubleshooting

11.1 Kernel errors

11.1.1 USB suspend error

If you get the following sort of error after suspend/sleep:

```
usb 2-4: usb_reset_and_verify_device Failed to disable LTM
usb usb2-port4: cannot disable (err=-32)
```

The Kernel Bug report #117811's thread suggest disabling USB auto-suspend.

- If you are using TLP:
  `nano /etc/default/tlp` and set `USB_AUTOSUSPEND=0`
- If you are **not** using TLP:
  `sudo echo on | tee /sys/bus/usb/devices/*/power/control`

11.2 Boot partition recognition is slow

This is usually caused by booting a OSX drive externally or creating one with the recovery process (started with with ⌘ + R on the MacBook Pro).

Beware: Before going forward, make sure you have an bootable Arch installation USB drive ready as you will also need to go reinstall GRUB as described in the next section.

To remedy this problem you will need to reset the NVRAM by pressing ⌥ + ⌘ + P + R together and turn on the machine keeping them pressed for about 20 seconds (or until the second boot-up chime).

11.3 Boot partition not detected after NVRAM reset

Boot from the Arch setup USB (see section 'Making a bootable USB').

Mount all the partitions again (see section 6.3.3) without making the directories:

```
$ cryptsetup open --type luks /dev/sda3 mctoasty
$ mount /dev/mapper/vg0-root /mnt
$ mount /dev/mapper/vg0-home /mnt/home
$ mount /dev/sda2 /mnt/boot
$ mount /dev/sda1 /mnt/boot/efi
$ swapon /dev/mapper/vg0-swap
```

Get into the `arch-chroot` environment and reinstall GRUB:

```
$ arch-chroot /mnt /bin/bash
$ run mkinitcpio -p linux
$ run grub-install
$ grub-mkconfig -o /boot/grub/grub.cfg
```

Exit chroot environment and shutdown:

```
$ exit
$ shutdown now
```

Remove the USB stick and boot up.

11.4 Text-to-Speech isn’t working

Check `journalctl -b` for an error relating to `flite` being missing. Installing it should resolve the text-to-speech problem.

```
$ sudo pacman -S flite
```

12 References

- Arch Linux's MacBookPro 11.x Wiki page
- Mattias Lindberg's Encrypted volume Arch install guide
- HowToForge | How to install Arch Linux with Full Disk Encryption
- Ubuntu | AppleKeyboard