Install EndeavourOS Desktop on Arch Linux ARM

July 4, 2020

The allure of an arm device for everyday use is it's small size and very low power consumption. The arm device could be used for browsing, email, typical office files, listening to music, light image editing such as Gimp, etc. The powerful PCs available today could be reserved for gaming, compiling, rendering image files, and other CPU/GPU intensive chores.

Theoretically you can install EndeavourOS on any ARM device that Arch Linux Arm supports. Just go to the Arch Linux Arm site, https://archlinuxarm.org/ hover over "Platforms" then the Architecture, brand name, etc. to see if your device is supported. Click on your device and get a page with overview and installation tabs. In reality, most ARM devices are not powerful enough to run a full blown EndeavourOS with a Desktop environment, but some are.

A script is now available for installing EndeavourOS running on Arch Linux Arm. In our testing we used an Odroid N2, an Odroid XU4, and a Raspberry PI 4b as the test beds. For running an Operating System with a DE, the Odroid N2 was the best, the Odroid XU4 was second, and the Raspberry Pi 4b third. All three were acceptable for most daily usage. I am sure there are others available, but these are the three that were available for testing. All three were also very proficient for running a headless LAN server, or NAS which is also available with the script.

Flash the micro SD card with the OS and bootloader.

The storage device for the Operating System will be either a Micro SD card or an eMMC card. The micro SD card must be at least 32GB, 64 GB gives more head room for logs and etc. Use a good name brand micro SD such as Samsung or SanDisk ultra, not an off brand. SD cards come in different speed classes, 2, 4, 6, and 10, with 10 being the fastest. Since the advent of 4k devices, there is also UHS class speed 1 and UHS class speed 3. UHS = Ultra High Speed. Get at least a class 10 HC1 device. Here is what to look for.

The speed class is the number 10 inside what looks like a C.
The speed class is followed by A1 which is a new specification I am not familiar with.
The UHS speed is the number 1 inside a U and also referred to as HC1. This one sold for $8.98 USD at amazon including a Micro SD to SD adapter.
The Odroid N2 and Odroid XU4 will also run off of eMMC

On a working x86_64 computer, insert the latest USB EndeavourOS ISO installer. Use the USB EndeavourOS ISO because it includes GParted, and cleaning up directories and files created during the flash process is not necessary as the ISO is not persistent.

Boot into the msdos/MBR version of the EndeavourOS installer ISO.
Insert the USB card reader containing the Micro SD to SD adapter with the micro SD card.
In the Endeavour welcome screen, select Partition Manager (Gparted)

INSTALL Base ARCH LINUX ARM on an ODROID XU4

Click on "GParted" tab and select the USB SD READER (ensure the right device is selected)
Note the Device Name of the SD READER, such as /dev/sda. Write this down.
Unmount any mounted partitions as indicated by a key symbol. Highlight the partition with the key symbol, right click on it, select “unmount”

Click on "Device" tab, and create a msdos Partition Table.
If an existing partition is mounted, this will fail. Highlight the partition with the key symbol, right click on it, select “unmount” Then repeat with the “Device” tab and create a msdos Partition Table.
Click on "Partition" tab, then new

Free Space preceding MiB: 1	Create as: Primary Partition
New Size MiB: leave as is	Partition name:
Free Space following (MiB): 0	File System: ext4
Align to: MiB	Label: Odroid-XU4

Apply All Operations
Close GParted

Close the Welcome screen
Open a terminal window.
Some of the following commands can take several minutes, so be patient.

```
$ mkdir Odroid
$ cd Odroid                 (isolates the process from the rest of the OS)
$ sudo su
# mkdir MP                  (MP = temporary Mount Point for the USB SD Reader)
# mount /dev/sdX1 MP    (change /dev/sdX1 as noted in Gparted, e.g. /dev/sda1)
# wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-xu3-latest.tar.gz
# bsdtar -xpf ArchLinuxARM-odroid-xu3-latest.tar.gz -C MP
# cd  MP/boot
# sh sd_fusing.sh /dev/sdX      (change /dev/sdX to what's appropriate. e.g. /dev/sda)
# cd ../..
# umount MP       (this could take a while)
# exit
$ exit
```

The Arch Linux ARM Operating System is now installed.
NOTE: To install OS on an emmc card, see "Flash an emmc card" at the end of this tutorial.

Shut down the x86_64 computer, and remove the uSD card from the USB SD Reader.
Set the boot switch selector on the Odroid-XU4 board, next to the HDMI jack, to the uSD position (to the left).
Insert the micro SD card into the XU4. Connect a monitor, keyboard, ethernet, & apply 5VDC

INSTALL Base ARCH LINUX ARM on an ODROID N2

Click on "GParted" tab and select the USB SD READER (ensure the right device is selected)
Note the Device Name of the SD READER, such as /dev/sda. Write this down.
Unmount any mounted partitions as indicated by a key symbol. Highlight the partition with the key symbol, right click on it, select “unmount”

Click on "Device" tab, and create a msdos Partition Table.
Click on "Partition" tab, then new
Free Space preceding MiB: 1 Create as: Primary Partition
New Size MiB: 256 Partition name:
Free Space following (MiB): XXXX File System: fat32
Align to: MiB Label: BOOT
Create a second partition
Free Space preceding MiB: 0 Create as: Primary Partition
New Size MiB: XXXXXX Partition name:
Free Space following (MiB): 0 File System: ext4
Align to: MiB Label: ROOT
Apply All Operations
Close GParted

Close the Welcome screen
Open a terminal window.
Some of these commands can take several minutes, so be patient.

```
$ mkdir OdroidN2
$ cd Odroid                  (isolates the process from the rest of the OS)
$ sudo su
# mkdir MPboot               (MP = temporary Mount Point for the USB SD Reader)
# mount /dev/sdX1 MPboot     (change /dev/sdX1 as noted in Gparted, e.g. /dev/sda1)
# mkdir Mproot
# mount /dev/sdX2  MProot
 # wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-n2-latest.tar.gz
# bsdtar -xpf ArchLinuxARM-odroid-n2-latest.tar.gz  -C MProot
# mv MProot/boot/*   MPboot
# dd if=MPboot/u-boot.bin  of=/dev/sdX conv=fsync,notrunc  bs=512 seek=1
# vi or nano  MPboot/boot.ini              (this is optional to set screen resolution)
    copy desired resolution into the hdmimode variable
    setenv monitor_onoff “true”        (close file)
    *** For eMMC Edit  MProot/etc/fstab    change boot partition to /dev/mmcblk0p1 ***
# umount MProot   MPboot
# exit
$ exit
```

The Arch Linux ARM Operating System is now installed.
Shut down the x86_64 computer, and remove the uSD card from the USB SD Reader.
Set the boot switch selector on the Odroid N2 board to MMC
Insert the micro SD card into the N2. Connect a monitor, keyboard, ethernet, & apply 5VDC

Install EndeavourOS on ARM

The following SHOULD work with any ARM SBC (Single Board Computer) once Arch Linux ARM base is installed on the device.

The default user is *alarm* with the password *alarm,* the default root password is *root.*
FYI, alarm = Arch Linux ARM and it is also the default hostname.

Anytime pacman asks for an option, just press Enter until you get to the Proceed with installation? then enter y. This is recommended for the first install of XFCE4 on the Arm device. Once you are comfortable with everything, a re-install can be done with any tweaking you desire.

After the SBC boots up:
Login as root (enter root for the username and root for the password)
use vi or nano to edit /etc/pacman.d/mirrorlist
comment out the server under "## Geo-IP based mirror selection and load balancing"
un-comment servers near you
pacman-key --init
pacman-key --populate archlinuxarm
pacman -Syy
pacman -Syu
pacman -S git
reboot & Login as root

RUN SCRIPT TO INSTALL EndeavourOS

git clone https://github.com/pudges-place/EndeavourOS-ARM.git
cd EndeavourOS-ARM

Make endeavour-ARM-install-V1.0.sh executable
chmod 774 endeavour-ARM-install-V1.0.sh
ls -l
-rwxrwxr-- 1 root root 5607 May 5 01:39 endeavour-ARM-install-V1.0.sh
sh endeavour-ARM-install-V1.0.sh (run the installer script)

When installing x86_64 EndeavourOS with the live ISO, at the "Packages" section, "base base-devel and commonly used packages" is selected by default. The first thing the ARM script does is install all these files so that EndeavourOS ARM matches EndeavourOS x86_64.

Next a bunch of configuring is done. Then the script asks which DE you want installed.
The packages are installed as per the package lists from x86_64 EndeavourOS. The ARM install is as much as possible the same as the X86_64 install, package for package.

The script also installs yay. Arch Linux ARM uses the same AUR as regular Arch. If you are interested in a package, go to the AUR, find the package, view the PKGBUILD for the package. The PKGBUILD has the line arch= which lists the architectures it will build for.
arch=('i686' 'x86_64' 'arm' 'armv7h' 'armv6h' 'aarch64')
In this case ‘armv7h’ and ‘aarch64’ are both listed so it will install with yay in ARM.
If it says arch=(‘any’) then it will also install with yay in ARM

After the script runs, it should reboot into Lightdm, or GDM depending on DE
If it is Lightdm, be sure to use lightdm-gtk-greeter-settings to customize your login screen.

I believe that some ARM devices do not support sleep mode or hibernation. It is recommended that in power management for your DE, to set everything to never and if possible disable sleep and hibernation.

Also, the storage capabilities of micro SD cards is limited. There are two ways to remedy this.

1 Keep your data on a LAN file server. The ARM server installation instructions are at:
https://github.com/pudges-place/EndeavourOS-ARM/blob/master/EOS-server-instructions.pdf
Click on Download and the PDF will be downloaded to your browser’s specified directory.
This is a guide to build an ARM LAN file server. If you already have a LAN file server with SSH enabled, it will guide you through preparing your new EndeavourOS ARM computer to access the LAN server using SSH and FUSE. Go to page seven SET UP A LINUX CLIENT COMPUTER, and follow the guide.

2 Consider installing a SSD in a USB3 enclosure for an external /home directory. It makes your data more transportable between devices, and it is also easy to back up your data to another USB3 SSD using rsync. Since the data and config files are separate from the uSD card, you can re-install and not touch the data. All you have to do is, after the re-install edit the fstab file to mount your USB SSD enclosure on /home. Because all of your dotfiles are in your home directory, when you launch your various applications, they’ll find all of your settings, preferences, and data.

If you want to add an external USB SSD enclosure as the /home directory, it is best to do so immediately after installation.

Installing an external SSD as the /home directory is described on the next page.

Have fun customizing your vanilla DE.

Manually Installing a SSD for the /home Partition

Install a SSD in a USB 3 enclosure as an external home partition. This SSD can be any size you want. Then this USB 3 SSD can be mounted during boot up in /etc/fstab as /home. This also makes this data easily backed up with rsync along with config files.

Power off the computer and connect a USB 3 external enclosure with a SSD installed.
IMPORTANT: To transfer the contents of /home then empty /home, there should not be anyone logged into a Desktop Environment. Way too many conflicts if someone is logged in.

Boot the computer, do NOT login. Press Ctrl-ALT-F2 to open a Console window (tty2).
Login to the console window as root and enter your root password. Still no user logged in.

```
# lsblk                          (should get something similar to this)
NAME          MAJ:MIN    RM     SIZE     RO  TYPE    MOUNTPOINT
sda             8:0       0    232.9G    0    disk
└─sda1          8:1       0    232.9G    0    part
mmcblk1       179:0       0     29.8G    0    disk
 |─mmcblk1p1  179:1       0      256M    0    part     /boot
 |--mmcblk1p2  179:2       0      28.4G   0    part     /
```

/dev/mmcblk1p1 is our OS device. The newly added device is /dev/sda1 If the device is brand new and has never been partitioned it may look different. Now we have determined that /dev/sda is our new device (your setup may be different such as /dev/sdb etc)

/dev/sda1 should not show a MOUNTPOINT. It is remotely possible /dev/sda1 will be mounted and show a mount point. If so, we have to un-mount it, and any other partition on /dev/sda that shows up as being mounted. Such as possibly /dev/sda2, /dev/sda3, etc. until all are un-mounted. Of course, do not un-mount anything on mmcblk1.

umount /dev/sda1 and possibly # umount /dev/sda2 etc.

Partition the USB SSD as one partition that uses all the space in the SSD
CAUTION: This WILL erase ALL DATA on your SSD.

```
# fdisk /dev/sda                ( or adjust to /dev/sdb or whatever is relevant )
Command  o                      (That's lower case o...create a new empty DOS partition table)
Command  n                                  (add a new partition)
  Partition type: p                         (p = primary)
 partition number: 1
 First sector:  enter to accept default
 Last sector: enter to accept default
  Partition #1 contains a ext4 signature.       (this warning may not appear, if so yes)
  do you want to remove the signature?  yes
Command: w                                  (write table to disk and exit)
```

mkfs.ext4 -L HOME /dev/sda1 (format our new partition to ext4, -L HOME is the label)

```
# lsblk
NAME         MAJ:MIN    RM     SIZE     RO  TYPE    MOUNTPOINT
sda            8:0       0    232.9G    0    disk
└─sda1         8:1       0    232.9G    0    part
mmcblk1      179:0       0     29.8G    0    disk
 |─mmcblk1p1  179:1       0      256M    0    part    /boot
 |--mmcblk1p2  179:2      0      28.4G    0    part    /
```

Now that /dev/sda is partitioned and formatted, we need to mount it on /mnt
mount /dev/sda1 /mnt
ls -al /mnt (should see the directory lost+found created by mkfs.ext4)
rm -rf /mnt/lost+found
ls -al /mnt (should see lost+found has been removed)
cp -rp /home/* /var (make a backup of your current home directory or directories)
cp -rp /home/* /mnt (copy contents of home to /mnt using recursive & preserve options)
ls -al /mnt/username (should see the copied contents of home)
rm -rf /home/* (empty the original home folder to be used as a mount point)
ls -al /home (home should be empty, if not repeat above command)
umount /dev/sda1 (un-mount /dev/sda1 from mount point /mnt)
mount /dev/sda1 /home (mount /dev/sda1 to now empty home folder)
ls -al /home/username (should see the home directories and dot files)

Modify /etc/fstab

Now that the dirty work is done, finish this up by exiting out of the console window.
Switch to the GUI mode by pressing Ctrl-Alt-F7, F6 , or maybe F1. Log into the desktop.
Open a terminal window
$ su then enter root password
# cp /etc/fstab /etc/fstab.orig (backup fstab)

Find the UUID for the home SSD partition.
lsblk -f /dev/sda

```
NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda
└─sda1 ext4  1.0  HOME  b6bd3cd3-c666-4259-9ad9-125a003ff231  221.7G    0%
```

you should see /dev/sda1 with a nice label of "HOME" and its UUID number.
Highlight the UUID number, then right click and click copy.

Using your favorite text editor, add the following line at the end of the /etc/fstab file
gedit /etc/fstab &

```
UUID=PASTE-Your-UUID-Number  /home  ext4   defaults,noatime   0 2
```

mount -a (should show no errors if fstab is correct, if not edit /etc/fstab again)
df -h /dev/sda1 (should show our new /home)

```
  Filesystem  1K-blocks      Used    Available      Use%    Mounted on
  /dev/sda1    245053044    61504  232473040   1%       /home
```

Reboot the computer and log into your Desktop.
If the computer takes a long time to boot up, you made a typo and it can't find the SSD. After it times out, It will say:
You are in emergency mode.
Give root password for maintenance
(or press Control-D to continue):
Type in your root password and do a blkid to check the UUID & device name such as /dev/sdb Edit /etc/fstab and look for typos. When you find your mistake, reboot and see what happens.

Once all is working, I would recommend removing the home directory backup(s) from /var. This frees up storage space and it is better security to not have your home directory in /var.

From now on the home folder will be on the SSD. The SBC device and the USB external SSD enclosure are now married and should always be used together.

Flash an eMMC card on Odroid XU4

On an Odroid XU4, if you opt to use a emmc card as the boot device, then additional steps are necessary. There are many opinions on micro SD VS emmc. The choice is yours.

To create an emmc card, first create a micro SD card as above including booting up the Odroid-XU4, installing keys, and update. Now that everything is working up to this point, place the emmc card on the emmc to microSD converter card. Then into the USB SD READER. Boot up the x86_64 computer with the latest EndeavourOS ISO.
Now repeat the above steps for the micro SD card but with the emmc card. When you shut down the x86_64 computer come back here and do the following.
Set the boot switch on the Odroid-XU4 board next to the HDMI jack to the uSD position (to the left). Ensure the micro SD card is in the XU4 SD slot. On the back side of the Odroid-XU4, Connect the eMMC module to the XU4, ensuring you hear a click when doing so.
Connect a monitor, keyboard, ethernet, & apply 5VDC.
The default user is *alarm* with the password *alarm,* the default root password is *root.*
Login as root

```
# cd /boot
# sh sd_fusing.sh  /dev/mmcblk0
```

fusing should take place without errors. Poweroff the Odroid. Remove the micro SD card. Switch the boot selector switch to emmc (to the right). Power up the Odroid. You should now be running on the emmc card. Login as root.

```
# pacman-key --init
# pacman-key --populate archlinuxarm
# pacman -Syu
```

Now go back to page 4 “Install EndeavourOS on ARM” section and continue installation.