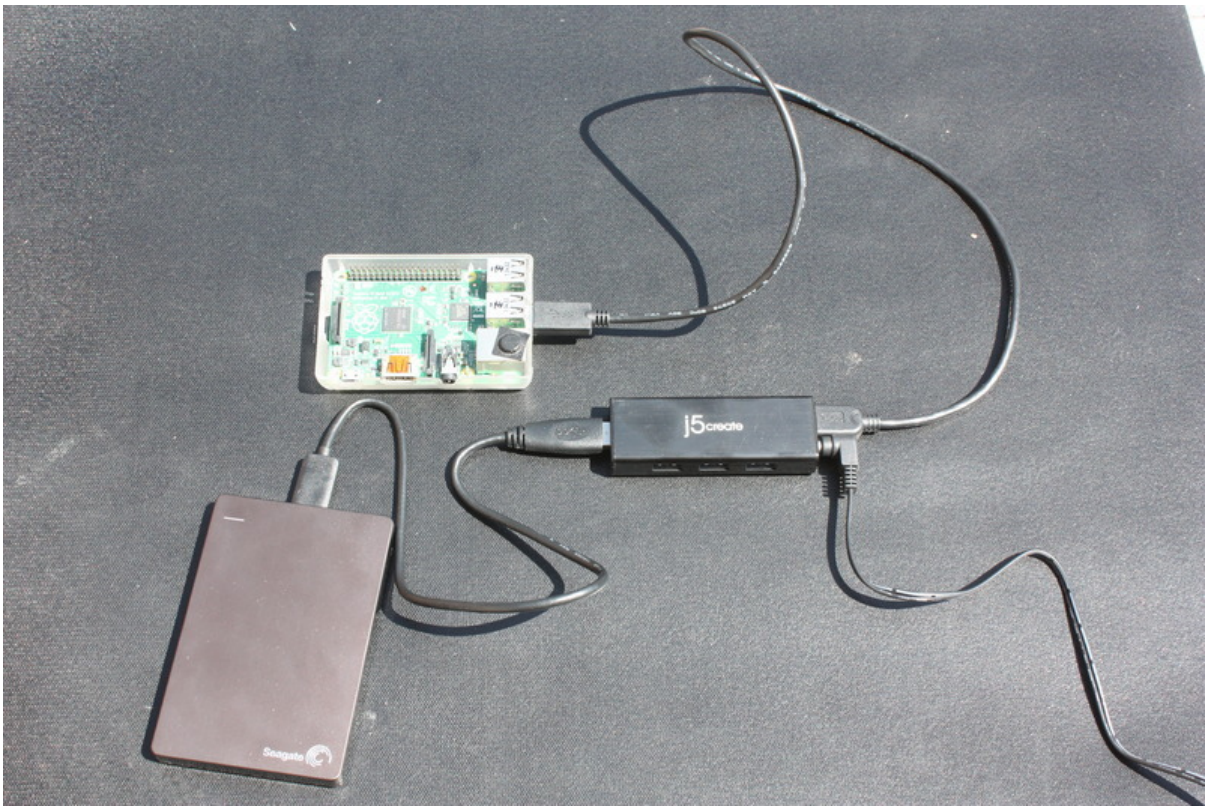
Using an External Drive as a Raspberry Pi Root Filesystem

Created by Brennen Bearnes



https://learn.adafruit.com/external-drive-as-raspberry-pi-root

Last updated on 2024-06-03 01:42:15 PM EDT

Table of Contents

Overview

One of the more noticeable limitations of the Raspberry Pi is using an SD card for its main storage. If you've been using a Pi for a while, you might be looking for more storage, faster access times, or media with a longer lifespan than many SD cards turn out to have.

For example, I'm working on a project to configure a set of custom Raspbian images to support some Adafruit hardware. I'd like to write a script to automate the process, but doing so will take quite a bit of storage space, and copying multi-gigabyte files can be pretty slow on an SD card.

One approach to this is to copy your root filesystem to an external drive, connected via USB, and tell the kernel to use this filesystem instead of one on the SD card.

Other potential uses include:

- A network fileserver where lots of storage is desirable.
- Write-heavy applications like data logging.

For my project, I'm using an external hard drive, but the same approach should work with an SSD or a USB stick.

This guide assumes enough familiarity with the command line to open a terminal, run a script, and maybe edit a few configuration files with Nano. Not sure about any of that? You can always start with one of these:

- What is this "Linux", anyhow? (https://adafru.it/sdn)

- What is the command line? (https://adafru.it/sdo)
- An Illustrated Shell Command Primer (https://adafru.it/CeI)

This tutorial and its associated script borrow heavily from HOWTO: Move the filesystem to a USB stick (https://adafru.it/eWs), by paulv on the Raspberry Pi forums.

What You'll Need

Powered USB Hub + External Hard Drive or SSD

First, decide what you'll be using for external storage. I picked a 1 terabyte USB hard drive from Seagate, because I had it laying around as a spare backup drive. You can probably find the equivalent on Amazon (https://adafru.it/eWt) for $60-70.

The Raspberry Pi doesn't have enough power to run most external hard drives or SSDs, so unless you have a drive with its own power supply, you'll probably need a powered USB hub.

I picked up a j5create JUH340 4-port hub (https://adafru.it/eWu) a while ago at a nearby Best Buy for about $25, and so far haven't had any problems with it. Adafruit offers a USB 2.0 powered hub (http://adafru.it/961) which has been tested with the Pi and should work well. You'll want to make sure you get something that supports USB 2.0.

Be wary of cheap, low-end hubs! If in doubt, it's best to use something that others have already tested with the hardware.

USB Thumb Drive

It won't offer all the benefits of a "real" drive, but if you've got a decent-sized USB thumb drive available, you can use it instead. This probably won't require a powered hub.

An SD Card for the Boot Partition

Unfortunately, no matter what you do, the Pi has to boot off an SD (microSD, in the case of the B+/Pi 2) card. Assuming you already have a Pi up and running, your existing card should work just fine, but you might want to use a spare just in case.

The card should have a standard Raspbian install already present. I started with Adafruit's 4GB SD card with a pre-installed Raspbian image (http://adafru.it/1121), but you should be able to use just about any card with an existing install.

If needed, you can follow our guide on preparing an SD card for the Pi (https://adafru.it/jd1).

This guide assumes that you're doing everything from the Pi!

You can certainly set up the SD card and external drive from another machine, but we'll do everything here from the command line on the Pi. You shouldn't need another computer as long as your Pi already boots into Raspbian.

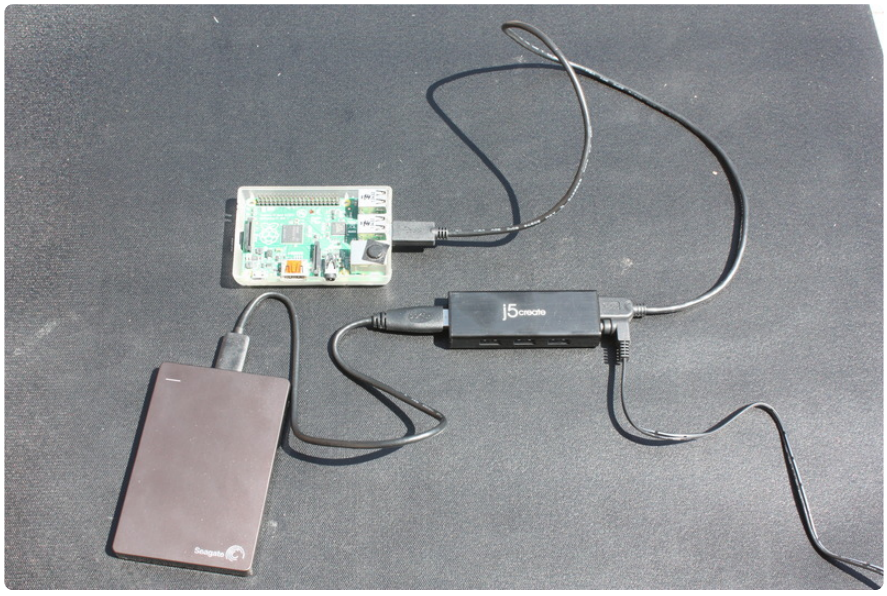
Configuring the Drive

Plug in the Drive

Hardware hookup here is pretty simple:

1. Plug USB hub into a free USB port on the Pi.
2. Plug power cable into powered USB hub.
3. Plug external drive into USB hub.

In the case where you're using a USB thumb drive, you can likely just plug right into the Pi.



Power on the Pi, if you haven't already, and bring up a terminal. Next, let's see if the drive showed up. First, try `lsusb`:

The line `Bus 001 Device 005: ID 0bc2:ab24 Seagate RSS LLC` looks promising. You'll likely see something similar, although the device isn't guaranteed to have a manufacturer string.

So next let's figure out where the drive actually shows up on the filesystem. Try using `dmesg`:

Here're the lines of interest:

```
[    8.822011] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
[    8.849001]  sda: sda1
[    8.862459] sd 0:0:0:0: [sda] Attached SCSI disk
```

Your output will probably look a little different, but unless you have another USB drive plugged in, your drive is almost certain to be `/dev/sda`.

Configure a New Root Filesystem

Once you know what drive to target, you can use a helper script we've written called `adafruit-pi-externalroot-helper`. The source for this script can be found here on GitHub (https://adafru.it/eWv).

If you're already using our package repository, you can install the script and run it like so:

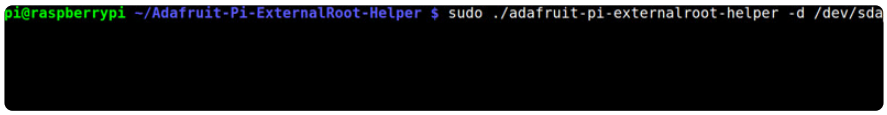
```
sudo apt-get install adafruit-pi-externalroot-helper
sudo adafruit-pi-externalroot-helper -d /dev/sda
```

Want to use our packages? Check out this guide to using the Adafruit Raspberry Pi package repository (https://adafru.it/Cfw).

If you're using a stock Raspbian install, and would rather not mess around with the package manager, try this instead:

```
git clone https://github.com/adafruit/Adafruit-Pi-ExternalRoot-Helper.git
cd Adafruit-Pi-ExternalRoot-Helper
./adafruit-pi-externalroot-helper -d /dev/sda
```

You'll be prompted to continue - double-check that the script is pointed at the right drive, and press **y** to continue.



This will take a long time.

It first creates a new filesystem on the target drive and copies everything on your root filesystem there using `rsync`. Lastly, it alters the kernel boot parameters in `/boot/cmdline.txt` and modifies a file called `/etc/fstab` to stop mounting the old root partition on the SD card.

adafruit-pi-externalroot-helper WILL destroy the existing contents of its target drive - make sure you want to proceed here!

Check the New Configuration and Reboot

Once the script is finished, you might want to check the contents of the following files:

`/mnt/etc/fstab`: the old `/dev/mmcblk0p2` entry for root should be commented out, and there should be a new one for your external device, identified by a path including a long unique id. Mine looks like this:

```
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
#/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
# a swapfile is not a swap partition, so no using swapon|off from here on, use
dphys-swapfile swap[on|off]  for that
/dev/disk/by-uuid/8bc7fb29-43c4-4179-9a32-89df37badad3    /   ext4
defaults,noatime  0       1
```

`/boot/cmdline.txt`: Instead of `root=/dev/mmcblk0p2`, you should see a `root=PARTUUID=...` parameter. Mine looks like so:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1
root=PARTUUID=ED5E1D73-8EAA-4C40-99A6-1E3A3795F98C rootdelay=5 rootfstype=ext4
elevator=deadline rootwait fbcon=map:10 fbcon=font:VGA8x8
```

When you reboot, `/mnt/etc/fstab` will be the new `/etc/fstab`. (The Arch Linux wiki has a pretty good explanation (https://adafru.it/eWx) of what this file does.)

If everything looks in order here, go ahead and reboot like so:

```
sudo reboot
```

Verify You're Using the New Drive

Once your Pi boots, you can log in like normal and try a couple of commands.

```
df -h
```

This should show `/dev/root` with its amount of **d**rive space **f**ree, which should correspond to what's available on your new drive. This is probably a good enough indication, if you're using a drive bigger than your SD card, but you can also check that `/dev/root` points to `/dev/sda1`:

```
readlink /dev/root
```

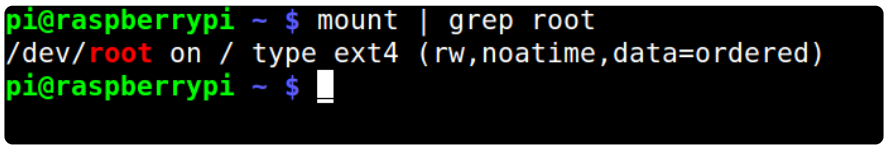
raspberry_pi_screencast-2015-04-12-23_06_52.gif

...and you should be able to look in `/dev/disk/` and see that it's available by its unique ID:

```
ls -l /dev/disk/by-uuid
ls -l /dev/disk/by-label
```

...and verify that it's mounted on `/`:

```
mount | grep root
```



These should show that `/dev/sda1` is

Recovering from a Failed Boot

If for some reason your Pi doesn't boot, you can restore things by opening the first partition on the SD card from another machine (Windows or OS X will work just fine) and editing `cmdline.txt`.

Just replace `root=PARTUUID=... rootdelay=5` with `root=/dev/mmcblk0p2`, which will point the root partition back to the second partition on your SD card, looking something like this:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait fbcon=map:10 fbcon=font:VGA8x8
```

Put the card back into the Pi, and it should boot like normal.

What the Helper Script Does

At this point, hopefully you're good to go with a working system. **You can safely skip this section**, but it might be helpful to know the basics of what's changed on your system.

We won't go over `adafruit-pi-externalroot-helper` line by line, but let's hit the high points. (For more detail, there are extensive comments in the script itself (https://adafru.it/eWy).)

Here's what the script does:

Install Dependencies

```
apt-get install gdisk rsync parted
```

We'll need a handful of utilities:

- `gdisk` is GPT fdisk (https://adafru.it/eWz), a tool for working with GUID Partition Tables (https://adafru.it/eWA).
- `rsync` is a utility for fast file transfers (https://adafru.it/eWB). It's usually used to copy and synchronize collections of files over the network, but also works between filesystems. Here it's used to copy your existing filesystem from the SD card to the external drive.
- `parted` is GNU Parted (https://adafru.it/eWC), another utility for working with partition tables. We'll use it to actually create a new partition on the external drive.

Create Target Partition & Filesystem

```
parted --script "${target_drive}" mklabel gpt
parted --script --align optimal "${target_drive}" mkpart primary ext4 0% 100%
mkfs -t ext4 -L rootfs "${target_partition}"
```

These commands create a new partition on our target drive (usually `/dev/sda`), using the entire drive, and then build a new ext4 (https://adafru.it/eWD) filesystem on that partition.

Find Partition UUID and Partition Unique GUID

```
eval `blkid -o export "${target_partition}"`
export target_partition_uuid=$UUID
export partition_unique_guid=`echo 'i' | sudo gdisk /dev/sda | grep 'Partition
unique GUID:' | awk '{print $4}'`
```

`blkid -o export /dev/sda1` gives a list of attributes for that partition, including a UUID, which we'll use later in `/etc/fstab` to make sure that the root partition is mounted from the external drive.

We could just use `/dev/sda1`, but the drive isn't guaranteed to show up as that device if other drives are plugged in. The UUID should be unique to the root partition on the external drive.

`echo 'i' | sudo gdisk /dev/sda | grep 'Partition unique GUID:' | awk '{print $4}'` gives us a similar value which we can put in `/boot/cmdline.txt` to tell the kernel what partition to treat as root.

Copy Filesystem to External Drive

```
mount "${target_partition}" /mnt
rsync -ax / /mnt
```

Here we attach `/dev/sda1` (or other target drive) to `/mnt` on the current filesystem, and use `rsync` to copy all the files across. The -a option means "archive", which should preserve file permissions, modification times, etc., and the -x option means "don't cross filesystem boundaries", so it'll skip `/boot` and `/mnt`.

Configure Pi to Boot With New Filesystem

```
cp /boot/cmdline.txt /boot/cmdline.txt.bak
sed -i "s|root=\/dev\/mmcblk0p2|root=PARTUUID=${partition_unique_guid}
rootdelay=5|" /boot/cmdline.txt

sed -i '/mmcblk0p2/s/^/#/' /mnt/etc/fstab
echo "/dev/disk/by-uuid/${target_partition_uuid}    /   ext4    defaults,noatime
0       1" &gt;&gt; /mnt/etc/fstab
```

These commands tweak `/boot/cmdline.txt` to boot from the new root partition (identified by the GUID from `gdisk`) and the new partition's `/etc/fstab` to mount it as root (identified by the UUID from `blkid`) instead of the old root partition from the SD card.

If something goes wrong here, you can copy `cmdline.txt.bak` over `cmdline.txt` from another machine, and your Pi should boot like before.

Further Reading

The `adafruit-pi-externalroot-helper` script is mostly an implementation of the process outlined in paulv's HOWTO: Move the filesystem to a USB stick (https://adafru.it/eWs) on the Raspberry Pi forums, with additional help from the following sources:

- The Arch Linux wiki, GUID Partition Table (https://adafru.it/eWE) for what a GPT is and some `parted` commands.
- GUID Partition Table (https://adafru.it/eWA), Wikipedia.

- Booting from an external USB drive (https://adafru.it/eWF), from the Raspberry Pi StackExchange.
- Booting a Raspberry Pi reliably from USB in the presence of multiple USB drives (https://adafru.it/eWG), by Stefan Krastanov.
- Speed up your Pi by booting to a USB flash drive (https://adafru.it/eWH), by Sam Hobbs.
- This comment (https://adafru.it/eWI) on `root=` in `init/do_mounts.c` in the Linux source tree.

While the steps here are fairly straightforward once you put them together, this can be a finicky process. If you run into any problems, please consider filing an issue on the GitHub project (https://adafru.it/eWJ).