Это версия страницы http://www.ideaheap.com/2013/07/stopping-sd-card-corruption-on-a-raspberry-pi/ из кеша Google. Она представляет собой снимок страницы по состоянию на 29 апр 2015 20:33:39 GMT. Текущая страница за прошедшее время могла измениться. Подробнее
Совет. Чтобы искать на странице, нажмите **Ctrl+F** или ⌘**-F** (для MacOS) и введите запрос в поле поиска.

Текстовая версия

in Raspberry Pi

Stopping SD Card Corruption on Raspberry Pi's Raspbian

The following are instructions for minimizing SD card writes for Raspberry Pi's "Raspbian" Distribution.

If you're like me, you've run into a corrupted SD card too many times to not become hell-bent on making it never happen again. I have the following setup, and it seems to be working well for me.

The biggest offender for Filesystem writes on any linux system is logging. If you are like me, you don't really look at /var/log after a recycle anyways. This area, and /var/run, a location where lock files, pid files and other "stuff" shows up, are the most common areas for mess-ups. Take a look at your blinking FS light on the board. Our goal is to make that light stay off as long as possible.

Set up tmpfs mounts for worst offenders. Do other tweaks.

Linux has with it the concept of an in-memory filesystem. If you write files to an in-memory filesystem, they will only exist in memory, and never be written to disk. There are two common mount types you can use here: ramfs, which will continue to eat memory until your system locks up (bad), and tmpfs, which sets a hard upper limit on mount size, but will swap things out if memory gets low (bad for raspberry pi, you will probably be hard stopping your device if it is low on memory).

We will first solve the usual corruption culprit and then move on to making sure we are covered when our programs decide to blow up.

The following two lines should be added to /etc/fstab:

```
1 none        /var/run        tmpfs   size=1M,noati
2 none        /var/log        tmpfs   size=1M,noati
```

UPDATE (unverified): I have been told that /var/run is now a symlink to a tmpfs filesystem, anyways, so you may not need to add /var/run anymore, and adding it may cause issues.

There's more, however. By default, linux also records when a file was last accessed. That means that every time you read a file, the SD card is written to. That is no good! Luckily, you can specify the "noatime" option to disable this filesystem feature. I use this flag generously.

Also, for good measure, i set /boot to read-only. There's really no need to regularly update this, and you can come back here and change it to "defaults" and reboot when you need to do something.

After this, /etc/fstab should look as follows:

```
1 proc            /proc                proc    defa
2 /dev/mmcblk0p1  /boot                vfat    ro,no
3 /dev/mmcblk0p2  /                    ext4    defau
```

```
4 none            /var/run        tmpfs   size=1M,n
5 none            /var/log        tmpfs   size=1M,n
```

UPDATE (unverified): I have been told that /var/run is now a symlink to a tmpfs filesystem, anyways, so you may not need to add /var/run anymore, and adding it may cause issues.

Go ahead and reboot now to see things come up. Check the Filesystem light on your raspberry pi after it's fully booted. You should see no blinking at all.

Disable swapping

As a note, since i have done the changes above, i have not corrupted an SD card. I'm not saying I've tried very hard, but it is much better, even with power plug pulls, which i tried a few of after doing these changes.

One protection against SD card corruption is an optional, but potentially "I'm glad i did that" change to disable swapping.

The raspberry pi uses dphys-swapfile to control swapping. It dynamically creates a swap partition based on the available RAM. This tool needs to be used to turn off swap, and then needs to be removed from startup.

Run the following commands to disable swapping forever on your system:

```
1 sudo dphys-swapfile swapoff
2 sudo dphys-swapfile uninstall
3 sudo update-rc.d dphys-swapfile remove
```

After doing this, call free -m in order to see your memory usage:

```
1 pi@raspberrypi ~ $ free -m
2              total       used       free     share
3 Mem:           438         59        378
4 -/+ buffers/cache:         22        416
```

```
5 Swap:          0          0          0
```

If you reboot, and run a free -m again, you should still see swap at 0. Now we don't have to worry about tmpfs filesystems swapping out to hard disk!

Housekeeping!

As you go on and install other tools and frameworks on your raspberry pi (like ROS), you need to be aware of where caches and logfiles are written to. If, for example, you use ROS, the nodes will, by default, log to ~/.ros/log/. For these sorts of things, and for ROS in particular, point these logs to a folder on /dev/shm. This mount is created by default on linux boxes, and is a tmpfs filesystem that is globally writable by default. Mine has 88 MB.
For example, pointing ROS logs to this file could be done by adding the following to your .bashrc:

```
1 export ROS_LOG_DIR=/dev/shm/rosLogs
```

If you feel like creating more mounts, feel free, but I have run into ownership issues that required having another script on startup that had to be run as root to chown directories to their proper owners.

Getting to 100%

The only way to fully protect against SD card corruption is to mount your root filesystem as readonly. For me, this was too much of a usability issue. I am editing files and installing new packages too often for this to be feasible. The steps listed above should, however, cover you 98% of the time. Try not to pull power while you're editing files or installing new packages on your device, and you should be fine. Still make backup images of your SD card every once in a while! This is a best practice no matter what!

Happy Hacking!

Write a Comment

21 COMMENTS

Andy
October 1, 2013

Can it be done some like readonly fs as default mode, but some script lauching enables write mode, then another script back fs to read only? or maybe adding user which have write permissions?

Reply to Andy

Adrian
November 3, 2013

Thank you verry much for posting this.
I have a lot of trouble with my Raspberry, trying to make it not ruin my card once a few days, so i came across this post; it will surely prove useful.

However, there's something i discovered while playing with my Rasp, trying to see how to make it stable. Maybe it will prove useful to others. It seems that Arch Linux is far more stable than Raspbian, regarding SD card corruption. Without your settings made, i pulled the power plug several times with no problems. I started to read directory tree "dir -Rlh /" and pulled the plug while still listing, still no problems. I started to write a file

"cat /dev/random > /root/file" and then pulled the plug and still no problems found. After restart, Arch Linux would still give me SSH and let me login.

So i would say that this distribution of Linux is much better, at least from this point of vue, than Debian.

Cheers

Reply to Adrian

jeanrochblais

December 2, 2013

my solution, mount /root on a hard drive and mount SD read only... only used for boot from nowon !

Reply to jeanrochblais

A L

January 7, 2015

Or move filesystem to USB 3.0 stick drive and boot from SD card. Have been somewhat every weekday last 3 months powering off (connected to my laptop docking station) 2-3 times a day "violently" but no problems so far! Now I have started to like that installation as it has proven its stability. I think I'll use Raspberry Pi only from USB stick filesystem even it takes one precious USB port.

Reply to A

ds00424

January 2, 2014

Thanks for the info.

As suggested, i moved /var/log to a 1M ramdrive. However, I found that the /var/log gets used up in about a week (on my system) due to various messages. Auth.log was the largest due to a 5 min cron using sudo to reports some stats to a central server.

Is there any recommended maintenance of /var/log directory to stop it from getting to 100% used?

I guess I could have a monitor script run every hour to check the usage and zero a few files (auth.log, syslog, messages) when nearing the limit. But that seems problematic.

Thanks.

Reply to ds00424

admin

February 1, 2014

I handle this sort of problem by either configuring it, if possible, or by throwing a cron job together that truncates the file after a period of time. "echo > /var/log/auth.log" would work.

In your case, the "logrotate" tool is probably going to be the right solution. "man logrotate" for more info.

Reply to admin

ZucOs

January 28, 2014

Thankx for the tips. Just one problem. With this line:

none /var/run tmpfs size=1M,noatime 00

My wireless keyboard and mouse stop working. If I comment that line in fstab they will work again. Do you have any clue how to solve this?

Reply to ZucOs

John

March 14, 2014

Same issue here.
none /var/run tmpfs size=1M,native 00
This line deactivates my wireless mouse and keyboard.
Any solution/idea how to solve this?

Reply to John

lsx49745

March 13, 2014

This article is very, very bad advice.
Disabling swapping: Just don't do it. The device is swapping for a reason.
Disabling swapping makes the device slow as hell.
It is already slow as hell, but after disabling swapping it is nearly impossible to work with it.

9 out 10 SSH attempts result in 'Connection timed out' or 'timed out while waiting to read' just because it is completely stuck.

Why would you store log files in memory any way?
The device is already short in memory.

Just buy a decent SD card and you will likely never experience problems.
I use the cheapest SD card I could find and I pull the power cord regularly and it has never caused any problem for at least 3 months now.

B.T.W.: most corruption problems arise from overclocking.
Don't do that either.

Reply to lsx49745

admin

March 22, 2014

Hello,

1: Swapping only occurs of you run out of RAM.
2: By Disabling it, I am saying "don't even try to save me if I'm out of ram". If you're swapping, you're already going to be "slow as hell".
3: Your failed SSH attempts are likely linked to this complete exhaustion of device memory. When linux runs out of memory, it is usually configured to start killing processes, so I would suspect that your new SSH connection is being killed over and over again. check /var/log/errors.log to see if you can see that after you successfully log in.

4: Our definitions of "Short on memory" may be different. By default, your device already has an 80 MB in-memory mount in /dev/shm, by the way. It won't actually "lock in" that memory until you actually use it. If i tie up all my write-heavy operations using 10M of memory, that is a price i'm willing to pay, as it improves my os ruggedness, and, if you care, the lifetime of the sd card.

5: I'm sure you can get lucky or that certain sd cards won't have this issue so much, but mine did. If you would rather not buy a new SD card, this is an option. I am not overclocking my pi, either.

Reply to admin

AzHofi

March 23, 2014

Linux not only swapping if runs out of ram, linux uses swap in thousand other cases, check the memory management algorithms in the kernel.

Reply to AzHofi

Nicholas Wertzberger

May 5, 2014

"Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space." – CentOS. You will need to provide proof by contradiction in this case.

Reply to Nicholas

JC

February 3, 2015

You have no idea what you're talking about.
Ask yourself: why is it that sysadmins and even extremely limited embedded devices like older phones ran linux WITH swap?

I'm astounded that so many people are following your advice here. Guess what, raspberrypi.org forums are not great source of information.

FYI: I have ran 2 raspberrypi servers since September 2011 WITH swap, WITH logging, torrenting, streaming, backup schedules, and have had NO corruption. I did invest on my sd-cards and do not overclock.

Reply to JC

Nicholas Wertzberger

March 16, 2015

You're entitled to your opinion. These changes let you run with less write-safe sd cards.

Swapping is helpful if you are strapped for memory, but it's not particularly a good situation once you're in there. Why is it a bad idea to disable swap or the things that are writing if your goal is to minimize writes out to your sd card? It only makes sense that, if you are trying to minimize the chance of having a catastrophic corrupting power failure, minimize the situations where it occurs.

If you can always guarantee a safe shutdown, you don't need these steps, but if you're plugging in and unplugging your raspberry pi all the time without properly shutting it down, these steps make that a LOT less likely to cause filesystem failure (which makes swap kind of useless, eh?)

Reply to Nicholas

K4c

May 17, 2014

In the past i was using voyager linux. This system was working as os for APs, default in ro mode. There were to scripts rw and ro to remount filesystem. Maybe its worth to investigate.

Reply to K4c

turiakki

October 8, 2014

What about this? SD-cards are relatively cheap and get cheaper as time passes. So, would not one solution be to set up a card for regular use making a backup image maybe like once a week. If the card file system ever gets corrupted, then just format and reinstall from backup image, until one day the card stops working entirely (AFAIK they have a quite limited write/read amount), then get a new one.

Some claim, however, and this most likely is true, that other USB storage devices are much quicker than micro sd. So, there are good reasons for only booting from card and then having everything else on USB. But then you loose one USB port permanently, as you cannot even get the OS running without it.

Reply to turiakki

Andy
January 12, 2015

For the sake of completeness, you may want to add how to go around apache2's dependency on a existing log file. It will not start if it has it's logging directory taken away and it's one of the main packages of the raspberry pi.

Reply to Andy

Nicholas Wertzberger
January 19, 2015

oh man... so there totally is two ways to handle this.... One of which is "if you don't want to re-configure apache, you could make an init.d script that generates those folder on boot time BEFORE apache starts up..." just sayin...

Reply to Nicholas

Andy
January 12, 2015

I was running raspbian as a Infra-red remote control device and the line in /etc/fstab that contain /dev/run would not allow me to execute any actions through the IR emitter on the circuit I built. I am using the package "lirc", I tested this between reboots being the removal/addition of this line the only thing to happen.

Reply to Andy

Nicholas Wertzberger
January 19, 2015

Hey andy, are you discussing /var/run?

Reply to Nicholas

Daniel

January 25, 2015

Thanks for this article.

But maybe this needs to get an update. /var/run now (raspbian of 2014) is a symlink inside /run which is mounted on a tmpfs anyway. And with adding that /var/run line into fstab, udevd wasn't able to properly run (besides of other daemons or whatever which may not have told me)

Reply to Daniel

Write a Comment

READERS WHO SHARED THIS PROBLEMI CON IL RASPBERRY PI · IT CAVE | EXPERIENCIA MONTANDO UNA WEB SOBRE UNA RASPBERRY PI · THANK YOU!

Independent Publisher empowered by WordPress