Raspberry SD card

The Raspberry is notorious for developing SD card IO issues after extensive SD card usage. With some simple adjustments and precautions the lifetime of the SD card can be increased significantly. Some guidelines:

- Use a SD card from a reliable brand, with minimal Class 10. Initially I encountered isues with some cheap 'noname' cards.
- Use a SD card that is over dimensioned (16 of 32 Gb).
- Change the file system type of the root filesysteem (/) to F2FS, which is targeted to flash memory types.
- Use a in memory ramdisk (tmpfs)for /tmp, /var/log in /etc/fstab.
- Mount the root filesystem (/) with options noatime and nodiratime
- Disable swapfile

With these measures I have two Raspberry Pi's running without problems 7×24 for about 2 years. They even survived two power failures.

F2FS File System

F2FS (Flash-Friendly File System) is a flash file system for the Linux kernel initially developed by Samsung. The F2FS file system takes into account the characteristics of NAND flash memory storage devices (such as solid-state disks,eMMC, and SD cards). Benefits are:

- Increase life span of flash storage due to less write required
- Increased I/O speed
- Doesn't suffer file size limitation of 4GB

Migrate EXT4 to F2FS

Our plan is now to replace the EXT4 file system of the root partition, by a F2FS file system, which is SD card friendly and is said to extend the card's durability.

Preparations

Required are: a Raspberry Pi (this example based on Raspbian Stretch). A second computer running Linux. Both should have support for the F2FS file system type available. in order sto install F2FS support run the next command in a terminal:

```
sudo apt-get install f2fs-tools
```

On the Raspberry Pi, type the following command in a Terminal:

```
sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL
```

```
   NAME      FSTYPE  SIZE MOUNTPOINT LABEL
   mmcblk0           60,1G
   ├─mmcblk0p1 vfat     60M /boot    boot
   └─mmcblk0p2 ext4     60G /
```

The micro SD card on the Raspberry Pi is, as usual, named mmcblk0. It consists of two partitions: mmcblk0p1 containing a vfat file system, mounted on /boot and mmcblk0p2 containing an ext4 file system, mounted on / (root).

Since the root partition of the SD card cannot be re-formatted on the running system, the system needs to be shut down, and, because the root partition will be on a f2fs file system when booted again, we need to shutdown the Raspberry Pi and perform the migration on the other computer. Then remove the micro SD card from the Raspbrry Pi and insert it into the second computer's SD card drive. Usually on common Linux desktop systems the card's file systems will be auto-mounted, and with

```
   sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL
   NAME   FSTYPE   SIZE MOUNTPOINT
LABEL
   . . .
   sdb            60,1G
   ├─sdb1 vfat      60M /media/user/boot
boot
   └─sdb2 ext4      60G /media/user/ad6203a1-ec50-4f44-a1c0-e6c3dd4c9202
   . . .
```

Now I prefer to make some more usable mount point names. Mount the boot partition off the Raspberry Pi, (/dev/sdb1 with a 60 MB vfat file system) to the temporary directory /tmp/source_boot. And second: Mount the root partition off the Raspberry Pi, (/dev/sdb2 with a 60 GB ext4 file system) to the temporary directory /tmp/source_root.

```
   sudo umount /dev/sdb1
   sudo umount /dev/sdb2
   sudo mkdir /tmp/source_boot
   sudo mkdir /tmp/source_root
   sudo mount -t vfat /dev/sdb1 /tmp/source_boot
   sudo mount -t ext4 /dev/sdb2 /tmp/source_root
```

Now, the SD card mount is much better readable:

```
   sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL

   sdb            60,1G
   ├─sdb1 vfat      60M /tmp/source_boot
   └─sdb2 ext4      60G /tmp/source_root
```

Backup root partition data

Since the conversion of the partition will destroy all the data. So we need to temporary backup the contents of the root partition. We copy with the -a option is important to ensure that really everything, including e.g. hidden/system files, is copied to the backup folder.

```
   sudo mkdir /tmp/backup_root
   sudo cp -a /tmp/source_root /tmp/backup_root
```

Create the f2fs partition

Now unmount the /dev/sdb2 partition and format it with the the f2fs file system.

```
   sudo umount /dev/sdb2
   sudo mkfs.f2fs /dev/sdb2
```

Restore root partition data

The partition has now a f2fs file system, but is empty, because by formatting all it's content got lost. Thus, we need to Re-mount the partition to it's previous mount point and copy the data that we backed up.

```
   sudo mount -t f2fs /dev/sdb2 /tmp/source_root
```

Now with the f2fs type option. The SD card mount will then look like this:

```
   sdb              60,1G
   ├─sdb1 vfat        60M /media/user/boot
   └─sdb2 f2fs        60G /media/user/root
```

We can now restore the root file system's content from the backup folder:

```
   sudo cp -a /tmp/backup_root/* /tmp/source_root
```

Make it boot

Make the Raspberry Pi SD card bootable with this new partition type. Edit the SD card's /etc/fstab file and adapt the root file system's mount entry to f2fs:

```
   sudo vi /media/user/root_fs/etc/fstab
   . . .
   /dev/mmcblk0p1  /boot  vfat    defaults          0       2
   /dev/mmcblk0p2  /      f2fs    defaults,noatime,discard  0  proc
/proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p2  /               f2fs    defaults,noatime,nodiratime,discard
0       1
tmpfs           /var/log        tmpfs
defaults,noatime,nosuid,mode=0755,size=80m    0 0
```

```
tmpfs           /var/tmp        tmpfs   defaults,noatime,nosuid,size=30m
0 0

     1
   . . .
```

Then edit the cmdline.txt file on the SD card's boot partition and adapt the root file system type parameter to f2fs:

```
sudo vi /media/user/boot_fs/cmdline.txt
----------------------------------------
console=serial0,115200 console=tty1 root=PARTUUID=84c78fde-02
rootfstype=f2fs fsck.repair=yes rootw>
```

That's it. You can now unmount the SD card's file systems, eject the card and insert it into the Raspberry Pi's card drive. The Pi will hopefully boot normally.

Fstab Adjustments

Usually temporary files and log files are written quite intensive, wearing out the SD card. By putting these files in memory (ramdisk), the fsash wear is eliminated. Only downside of this approach is that log files vanish when the Raspberry reboots. Use a in memory ramdisk (tmpfs)for /tmp, /var/log in /etc/fstab. Mount the root filesystem (/) with options noatime and nodiratime, which eliminates the update of the file properties when a file is last accessed.

```
sudo nano /etc/fstab
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p2  /               f2fs    defaults,noatime,nodiratime,discard
0       1
tmpfs           /var/log        tmpfs
defaults,noatime,nosuid,mode=0755,size=80m    0 0
tmpfs           /var/tmp        tmpfs   defaults,noatime,nosuid,size=30m
0 0
```

Disable Swapfile

Debian/Raspbian uses a swapfile instead of a swap partition, which can be configured with the dphys-swapfile utility. Usage: /sbin/dphys-swapfile {setup|swapon|swapoff|uninstall}

```
sudo dphys-swapfile swapoff
```

There is no necessity to uninstall, however if you are not using it, and do want the space, you can safely remove it. Alternate command to remove:

```
sudo dphys-swapfile uninstall
```