打造更持久的树莓派

作者：evling
原文链接：https://ld246.com/article/1607272046459
来源网站：链滴

导读

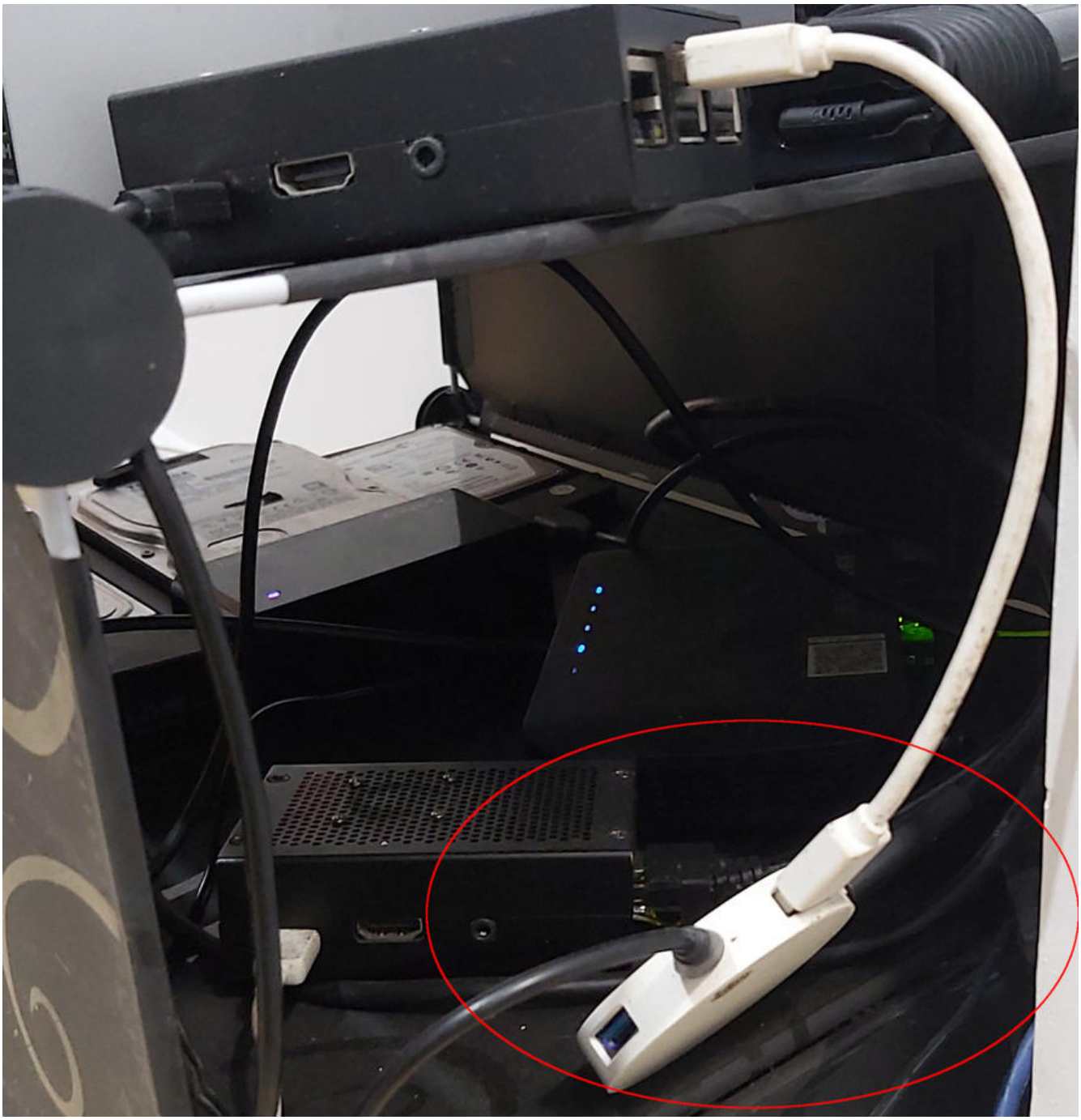
写了那么多期斐讯k3,想必诸位都视觉疲劳了，今天终于轮到树莓派上场了。要想树莓派长久稳定的行，且可扩展能力强，能够把它打造成网盘，那么扩展存储是必然的，同时树莓派最大的毛病就是意断电后文件系统易损坏，需要手动修复，那有没有一劳永逸的法子，答案是肯定的，接着看就是！

设计推导

示例硬件原料

- 树莓派3B+
- USB 无源移动机械硬盘
- 有源 USB HUB

外置存储建议诸位采用有源 usb 移动机械硬盘，容量大且价格适宜，无源的话可以用有源 usb hub 接，就像易雾君这样的



设计畅想

咱们此次是没有准备 TF 卡的，移动文件也需放置于移动设备上，可以考虑分三个区足矣，一个启动区，一个系统分区，一个数据分区。

为了防止失窃造成数据泄漏，还要将数据分区进行加密。

系统分区作为一个基础支持性的分区，最好是不让它变更，固化它，所有需要变更的数据只发生数据区，这倒是可以，系统分区咱给它用上 overlayfs ，数据分区呢主打跑 docker ，在部署阶段设置让统分区能够写入，将 docker 的存储目录改到数据分区，这样就能确保 docker 创建的容器都写在数分区了，基本不与系统分区有何干系，最终部署好以后就可直接固化系统分区了，以后需要增加docke容器呀，就大胆操作吧。

操作系统选型

树莓派 3B+ 本身是基于 ARM 64 位的处理器，果断选用 64 位的操作系统，易雾君用的是由 openfans 出品的 Debian-Pi-Aarch64，尤其好用，项目地址：https://github.com/openfans-community offical/Debian-Pi-Aarch64，支持 2Bv1.2, 3B, 3B+, 3A+, 4B 众多系列。

USB 启动问题

- 树莓派 3B 是无法直接从 usb 存储启动的，可以参考 《完全抛弃TF卡,从 USB 启动树莓派》，进设置，完成设置后，后边就可以不需要 TF 卡了
- 树莓派 3B+ 及以上则支持直接从 USB 存储启动系统

操练时刻

系统镜像准备

易雾君采用了这个固件 2020-06-22-OPENFANS-Debian-Buster-Aarch64-ext4-v2020-2.0-U4-Release.img.xz

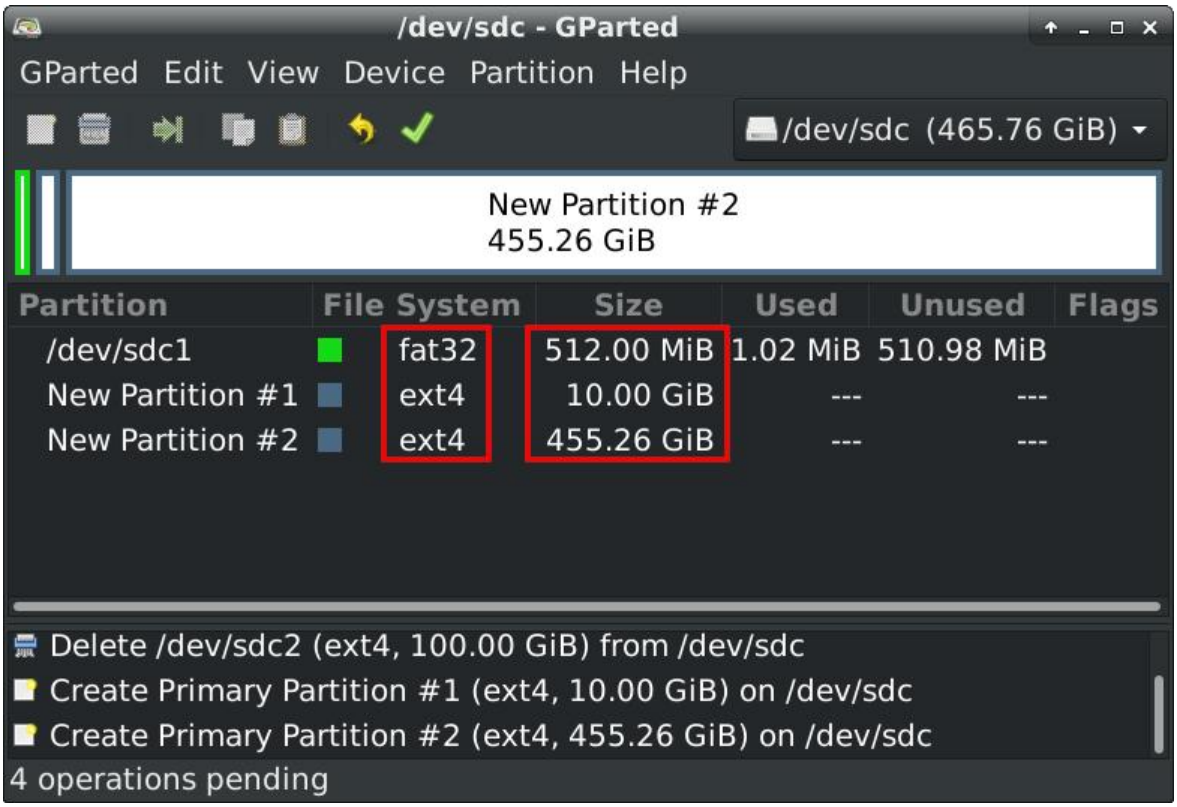
- 资源链接：https://pan.evling.me/s/g9pkSR6eACKjwRP
- 访问密码：在公众号 易雾山庄 回复获取密码 即可
- 可以到上面提供的项目地址那里去下载，可能网速会比较慢而已

硬盘分区

这里咱就不采用直接将镜像 dd 到硬盘，而是先按照如下设定分好区。

- 硬盘总大小：500G
- 启动分区：512MB fat32
- 系统分区：10GB ext4
- 数据分区：剩下的所有

易雾君使用 gparted 进行了分区及格式化，诸位也可使用其他工具，如 fdisk 等。第三个分区可以暂不用格式化，因为咱装好系统之后打算对它进行加密处理。

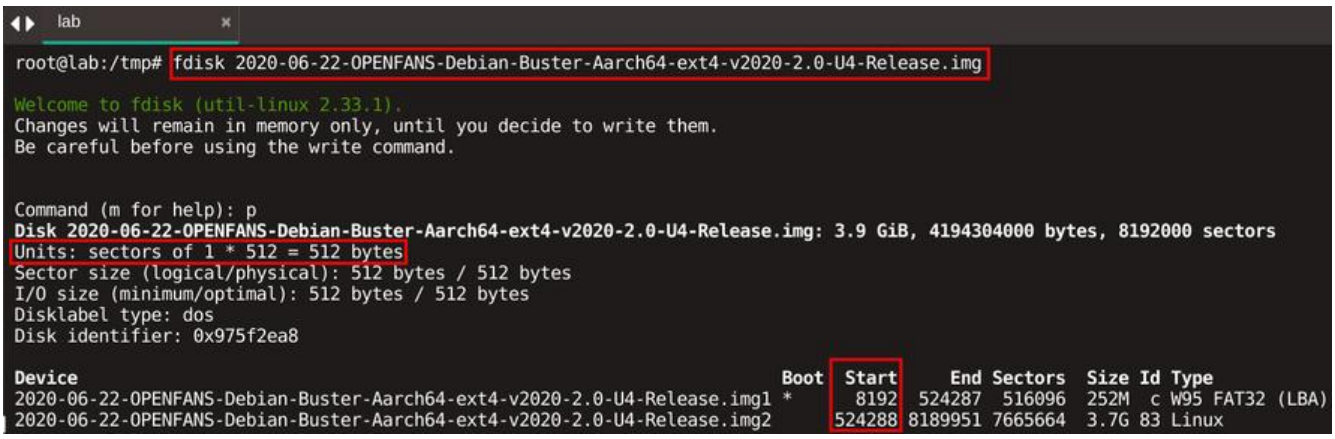


固件刷写

解压 xz 压缩形式的镜像文件

```
xz -d 2020-06-22-OPENFANS-Debian-Buster-Aarch64-ext4-v2020-2.0-U4-Release.img.xz
```

转换起始偏移



- 8192 * 512 = 4194304
- 524288 * 512 = 268435456

挂载 boot 分区并将文件同步到移动硬盘的 boot 分区。

```
mkdir /tmp/boot
mount -o loop,offset=4194304 ./2020-06-22-OPENFANS-Debian-Buster-Aarch64-ext4-v2020
2.0-U4-Release.img /tmp/boot
rsync -Pa /tmp/boot/ /media/root/28CD-91D9/
```

```
sync
```

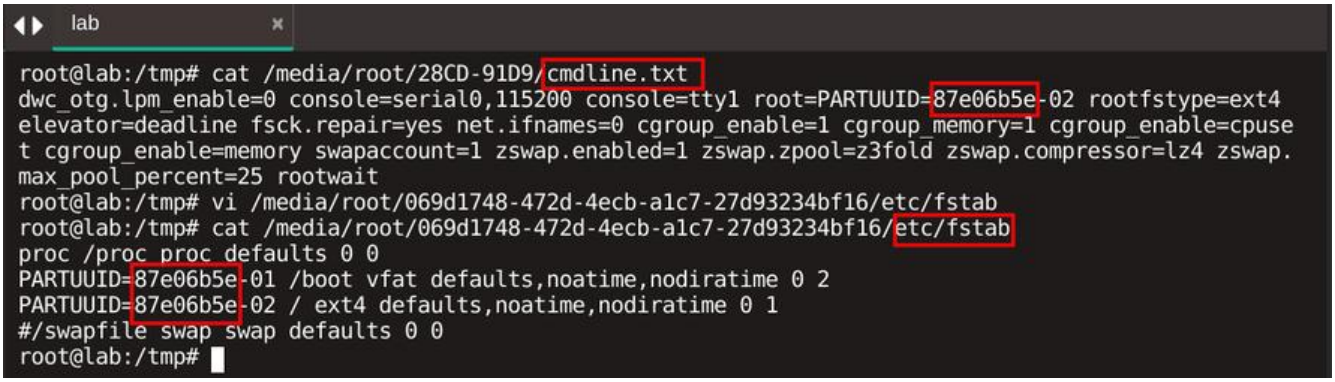
挂载 system 分区并将文件同步到移动硬盘的 system 分区。

```
umount /tmp/boot
mkdir /tmp/system
mount -o loop,offset=268435456 ./2020-06-22-OPENFANS-Debian-Buster-Aarch64-ext4-v200-2.0-U4-Release.img /tmp/system
sync
umount /tmp/system
```

更新新环境的分区 id， 先查看移动硬盘的分区 id 值

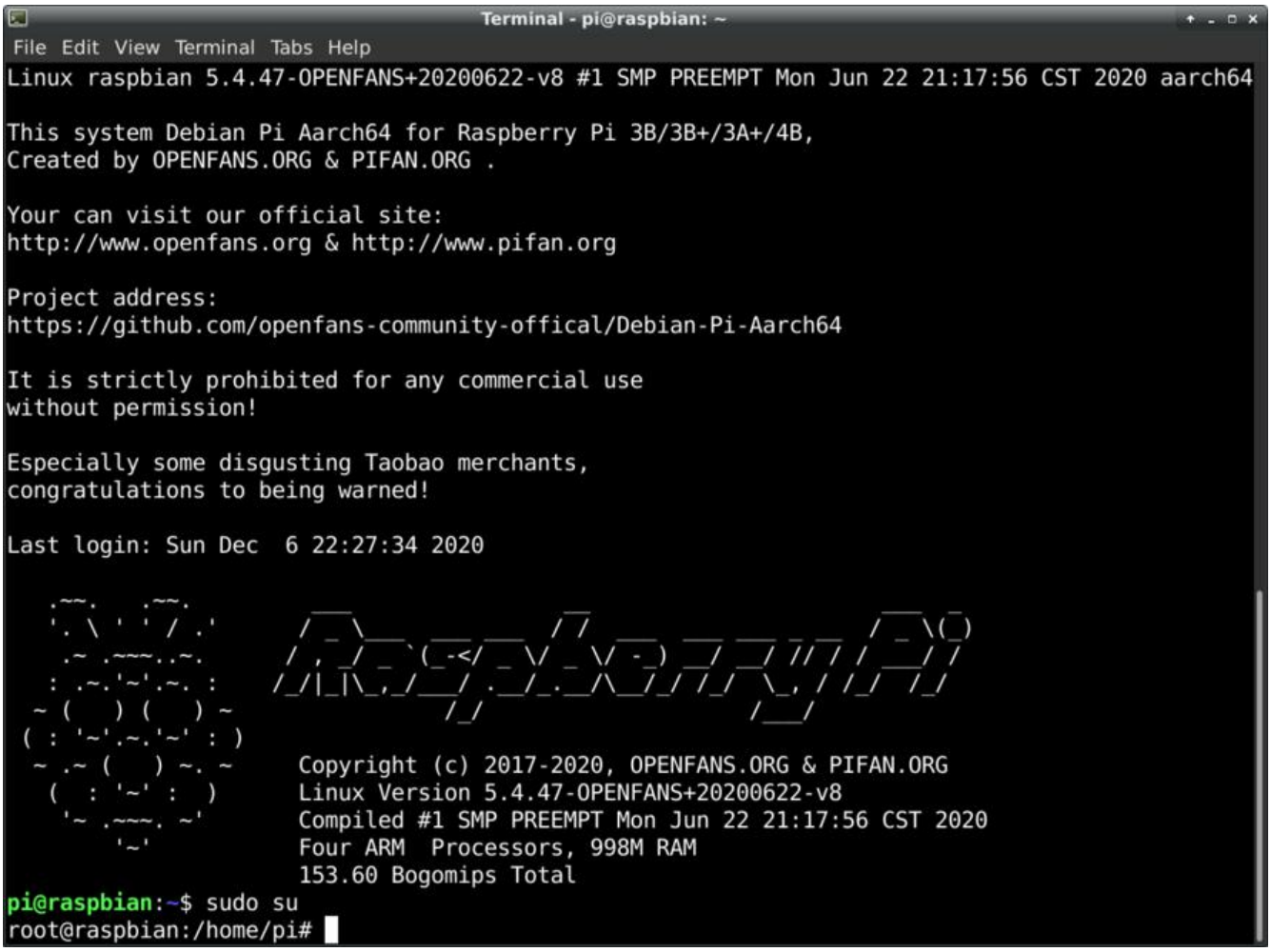
```
root@lab:/tmp# blkid
/dev/sdc1: UUID="28CD-91D9" TYPE="vfat" PARTUUID="87e06b5e-01"
/dev/sdc2: UUID="069d1748-472d-4ecb-a1c7-27d93234bf16" TYPE="ext4" PARTUUID="87e6b5e-02"
/dev/sdc3: UUID="0a4a8d70-6c7a-4efa-b941-1b7d28ac2fad" TYPE="ext4" PARTUUID="87e0b5e-03"
```

得到 87e06b5e， 更新 boot 分区下文件 /media/root/28CD-91D9/cmdline.txt 及系统分区下文件 media/root/069d1748-472d-4ecb-a1c7-27d93234bf16/etc/fstab 对应的值。



正式环境初始化配置

这时拔出 usb 移动硬盘插到树莓派上去，加电稍等几分钟即可进入系统。

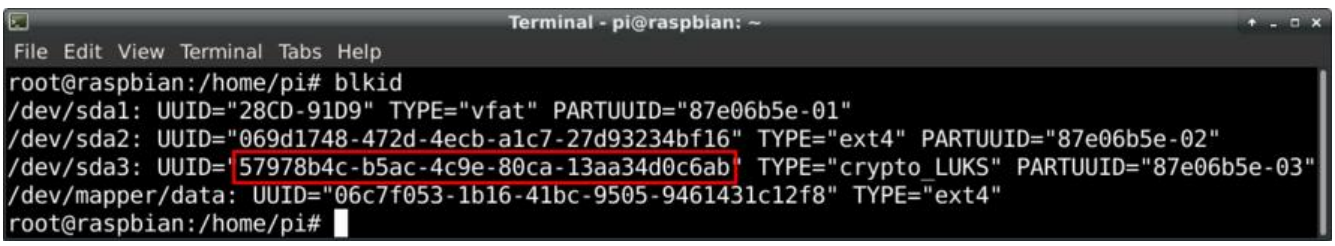


加密数据分区

```
apt update && apt install -y cryptsetup
cryptsetup luksFormat /dev/sda3
# 按照提示输入大写的 YES， 紧接着输入加密磁盘的口令
dd if=/dev/urandom of=/root/enc.key bs=1 count=4096
cryptsetup luksAddKey /dev/sda3 /root/enc.key
# 输入你设定的磁盘口令进行授权
cryptsetup luksOpen /dev/sda3 data -d /root/enc.key
mkfs.ext4 /dev/mapper/data
mkdir /data
mount /dev/mapper/data /data
```

开机自动挂载加密的数据分区

获取数据分区的 uuid 值 57978b4c-b5ac-4c9e-80ca-13aa34d0c6ab



在 /etc/crypttab 增加如下一行

```
data    UUID=57978b4c-b5ac-4c9e-80ca-13aa34d0c6ab      /root/enc.key
```

在 /etc/fstab 增加如下一行

```
/dev/mapper/data /data ext4 defaults 0 0
```

配置交换分区，大小设定为 8 GB

```
fallocate -l 8G /data/swapfile
chmod 600 /data/swapfile
mkswap /data/swapfile
swapon /data/swapfile
```

将 /etc/fstab 下交换分区文件的路径更新为新路径 /data/swapfile，交换分区那行应形如

```
/data/swapfile swap swap defaults 0 0
```

docker 配置

安装 docker-compose

```
apt update && apt install docker-compose
```

修改 docker 的存储目录到 /data/docker ，在系统服务 /lib/systemd/system/docker.service 增加个启动参数 --graph=/data/docker

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --graph=/data/docker --containerd=/run/containerd/con
ainerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old locati
n
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to
```

```
make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not support it.
# Only systemd 226 and above support this option.
TasksMax=infinity

# set delegate yes so that systemd does not reset the cgroups of docker containers
Delegate=yes

# kill only the docker process, not all processes in the cgroup
KillMode=process

[Install]
WantedBy=multi-user.target
```

重启 docker 服务

```
systemctl restart docker
```

固化脚本

根据前面的需求，咱只希望对系统分区，即 /dev/sda2 进行固化，新增脚本文件 /sbin/overlayRoot.h，注意变量 rootDev 需指定为你实际的分区。

```
#!/bin/sh
#  Read-only Root-FS for Raspian using overlayfs
#  Version 1.0
#
#  Created 2017 by Pascal Suter @ DALCO AG, Switzerland
#  to work on Raspian as custom init script
#  (raspbian does not use an initramfs on boot)
#
#  Modified 2017-Apr-21 by Tony McBeardsley
#
#  This program is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, either version 3 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see
```

```
#    <http://www.gnu.org/licenses/>.
#
#
#  Tested with Raspbian mini, 2017-01-11
#
#  This script will mount the root filesystem read-only and overlay it with a temporary tempfs
#  which is read-write mounted. This is done using the overlayFS which is part of the linux ker
el
#  since version 3.18.
#  when this script is in use, all changes made to anywhere in the root filesystem mount will b
 lost
#  upon reboot of the system. The SD card will only be accessed as read-only drive, which sign
ficantly
#  helps to prolong its life and prevent filesystem coruption in environments where the syste
 is usually
#  not shut down properly
#
#  Install:
#  copy this script to /sbin/overlayRoot.sh and add "init=/sbin/overlayRoot.sh" to the cmdline.
xt
#  file in the raspbian image's boot partition.
#  I strongly recommend to disable swapping before using this. it will work with swap but that
just does
#  not make sens as the swap file will be stored in the tempfs which again resides in the ram.
#  run these commands on the booted raspberry pi BEFORE you set the init=/sbin/overlayRoo
.sh boot option:
#  sudo dphys-swapfile swapoff
#  sudo dphys-swapfile uninstall
#  sudo update-rc.d dphys-swapfile remove
#
#  To install software, run upgrades and do other changes to the raspberry setup, simply rem
ve the init=
#  entry from the cmdline.txt file and reboot, make the changes, add the init= entry and rebo
t once more.

fail(){
        echo -e "$1"
        /bin/bash
}


# Load overlay module
modprobe overlay
if [ $? -ne 0 ]; then
    fail "ERROR: missing overlay kernel module"
fi


# Mount /proc
mount -t proc proc /proc
if [ $? -ne 0 ]; then
    fail "ERROR: could not mount proc"
fi
```

```
# Create a writable fs on /mnt to then create our mountpoints
mount -t tmpfs inittemp /mnt
if [ $? -ne 0 ]; then
    fail "ERROR: could not create a temporary filesystem to mount the base filesystems for over
ayfs"
fi


# Mount a tmpfs under /mnt/rw
mkdir /mnt/rw
mount -t tmpfs root-rw /mnt/rw
if [ $? -ne 0 ]; then
    fail "ERROR: could not create tempfs for upper filesystem"
fi



# Identify root fs device, PARTUUID, mount options and fs type

#rootDev=`blkid -o list | awk '$3 == "/" {print $1}'`
# Changed here(point to / ) in case the cmd above doesn't work # By ChenYang 20171122
rootDev=/dev/sda2
rootPARTUUID=`awk '$2 == "/" {print $1}' /etc/fstab`
rootMountOpt=`awk '$2 == "/" {print $4}' /etc/fstab`
rootFsType=`awk '$2 == "/" {print $3}' /etc/fstab`


# Mount original root filesystem readonly under /mnt/lower
mkdir /mnt/lower
mount -t ${rootFsType} -o ${rootMountOpt},ro ${rootDev} /mnt/lower
if [ $? -ne 0 ]; then
    fail "ERROR: could not ro-mount original root partition"
fi


# Mount the overlay filesystem
mkdir /mnt/rw/upper
mkdir /mnt/rw/work
mkdir /mnt/newroot
mount -t overlay -o lowerdir=/mnt/lower,upperdir=/mnt/rw/upper,workdir=/mnt/rw/work o
erlayfs-root /mnt/newroot
if [ $? -ne 0 ]; then
    fail "ERROR: could not mount overlayFS"
fi


# Create mountpoints inside the new root filesystem-overlay
mkdir /mnt/newroot/ro
mkdir /mnt/newroot/rw

# Remove root mount from fstab (this is already a non-permanent modification)
grep -v "$rootPARTUUID" /mnt/lower/etc/fstab > /mnt/newroot/etc/fstab
echo "#the original root mount has been removed by overlayRoot.sh" >> /mnt/newroot/etc/f
```

```
tab
echo "#this is only a temporary modification, the original fstab" >> /mnt/newroot/etc/fstab
echo "#stored on the disk can be found in /ro/etc/fstab" >> /mnt/newroot/etc/fstab


# Change to the new overlay root
cd /mnt/newroot
pivot_root . mnt
exec chroot . sh -c "$(cat <<END

        # Move ro and rw mounts to the new root
        mount --move /mnt/mnt/lower/ /ro
        if [ $? -ne 0 ]; then
            echo "ERROR: could not move ro-root into newroot"
            /bin/bash
        fi
        mount --move /mnt/mnt/rw /rw
        if [ $? -ne 0 ]; then
            echo "ERROR: could not move tempfs rw mount into newroot"
            /bin/bash
        fi

        # Unmount unneeded mounts so we can unmout the old readonly root
        umount /mnt/mnt
        umount /mnt/proc
        umount /mnt/dev
        umount /mnt

        # Continue with regular init
        exec /sbin/init
END
)"
```

给予执行权限

```
chmod a+x /sbin/overlayRoot.sh
```

在 /boot/cmdline.txt 文件下增加 init=/sbin/overlayRoot.sh，形如

```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=87e06b5e-02
ootfstype=ext4 elevator=deadline fsck.repair=yes net.ifnames=0 cgroup_enable=1 cgroup_m
mory=1 cgroup_enable=cpuset cgroup_enable=memory swapaccount=1 zswap.enabled=1 z
wap.zpool=z3fold zswap.compressor=lz4 zswap.max_pool_percent=25 rootwait init=/sbin/ov
rlayRoot.sh
```

为了后边有时必须对系统作出更改，咱还是做个切换 可写及 只读模式的命令，如 reboot_rw 重启系后做出改动，随后执行 reboot_ro 恢复到 只读模式，省心。定义这两个函数如下

```
cat << EOF >> ~/.bashrc
function reboot_rw(){
  sed -i 's/ init=\/sbin\/overlayRoot.sh//g' /boot/cmdline.txt
  reboot
}
function reboot_ro() {
  sed -i 's/\($\)/ init=\/sbin\/overlayRoot.sh/g' /boot/cmdline.txt
```

```
  reboot
}
EOF
```

结语

树莓派基础环境弄好了就放心的 24 小时开机吧，能跑个网盘应用、站点、博客应用就很不错了，敬期待下篇《树莓派 docker 跑 kodbox 网盘》

对了对了，更多精彩不要错过，扫码关注我哟！诸位有心的话请前往“易雾山庄”公众号进行多多点，点得越凶，那我也更得越猛。要不要告哈嘛，都是准备好的干货。

参考

- [1]: 使用overlayfs打造一个只读的不怕意外关机的树莓派Raspberry Pi