Bootable SD card

From linux-sunxi.org

Contents

Introduction

This page describes how to create a bootable SD card. Depending on how the SD card is connected, the location to write data to can be different. Throughout this document ${card} refers to the SD card and ${p} to the partition if any. If the SD card is connected via a USB adapter, linux will know it for example as /dev/sdb (with /dev/sda being a boot drive). Please notice that this device can be different based on numerous factors, so when not sure, check the last few lines of dmesg after pluging in the device. If connected via a SD slot on a device, linux will know it as /dev/mmcblk0 (or mmcblk1, mmcblk2 depending on which mmc slot is used).

Data is either stored raw on the SD card or in a partition. If ${p} is used then the appropiate partition should be used. Also this differs for USB adapters or mmc controllers. When using an USB adapter, ${p} will be 1, 2, 3 etc so the resulting device is /dev/sdb1. Using an mmc controller, this would be p1, p2, p3 etc so the resulting device is /dev/mmcblk0p1.

To summarize, ${card} and ${card}${p}1 mean /dev/sdb and /dev/sdb1 on a USB connected SD card, and /dev/mmcblk0, /dev/mmcblk0p1 on an mmc controller connected device.

If the SD card is connected in another way, the device nodes can change to be even different, take this into account.

SD Card Layout

start	size	usage
0	8KB	Unused, available for partition table etc.
8	24KB	Initial SPL loader
32	512KB	u-boot
544	128KB	environment
672	352KB	reserved
1024	-	Free for partitions

Identify the card

First identify the device of the card and export it as `${card}`.

If the SD card is connected via USB and is sdX (replace X for a correct letter)

```
1 export card=/dev/sdX
2 export p=""
```

If the SD card is connected via mmc and is mmcblk0

```
1 export card=/dev/mmcblk0
2 export p=p
```

Cleaning

To be on safe side erase the first part of your SD Card (also clears the partition table).

```
dd if=/dev/zero of=${card} bs=1M count=1
```

If you wish to keep the partition table, run:

```
dd if=/dev/zero of=${card} bs=1k count=1023 seek=1
```

Bootloader

```
dd if=spl/sunxi-spl.bin of=${card} bs=1024 seek=8
dd if=u-boot.img of=${card} bs=1024 seek=40
```

As an alternative, you can use the combined u-boot-sunxi-with-spl.bin:

```
dd if=u-boot-sunxi-with-spl.bin of=${card} bs=1024 seek=8
```

To update the bootloader from the u-boot prompt itself:

```
mw.b 0x48000000 0x00 0x100000                # Zero buffer
tftp 0x48000000 u-boot-sunxi-with-spl.bin    # Or use load to re
mmc erase 0x10 0x400                         # Erase the MMC reg
mmc write 0x48000000 0x10 0x400              # Write updated u-b
```

If using u-boot v2013.07 or earlier then the offsets, and therefore procedure, slightly different

```
dd if=spl/sunxi-spl.bin of=${card} bs=1024 seek=8
dd if=u-boot.bin of=${card} bs=1024 seek=32
```

Partitioning

With recent u-boot it's fine to use ext2/ext3 as boot partition, and other filesystems in the root partition too.

With separate boot partition

Partition the card with a 16MB boot partition starting at 1MB, and the rest as root partition

```
sfdisk -R ${card}
cat <<EOT | sfdisk --in-order -L -uM ${card}
1,16,c
,,L
EOT
```

You should now be able to create the actual filesystems:

```
mkfs.vfat ${card}${p}1
```

```
2 mkfs.ext4 ${card}${p}2
```

```
1 cardroot=${card}${p}2
```

Boot Partition

```
1 mount ${card}${p}1 /mnt/
2 cp linux-sunxi/arch/arm/boot/uImage /mnt/
3 cp sunxi-boards/sys_config/a10/script.bin /mnt/
4 umount /mnt/
```

Single partition

(experimental)

```
1 sfdisk -R ${card}
2 cat <<EOT | sfdisk -L --in-order -uM ${card}
3 1,,L
4 EOT
```

```
1 mkfs.ext4 ${card}${p}1
```

```
1 cardroot=${card}${p}1
```

Boot Partition

```
1 mount ${card}${p}1 /mnt/
2 mkdir /mnt/boot
3 cp linux-sunxi/arch/arm/boot/uImage /mnt/boot
4 cp sunxi-boards/sys_config/a10/script.bin /mnt/boot
5 umount /mnt/
```

GPT (experimental)

There is 8kb space for partition data. MBR uses only the first sector and allows for 4 partitions. If you are concerned about the 4 partition limitation you can try different partitioning scheme. While GPT standard mandates that GPT should have at least 128 entries gdisk can resize a GPT partition to 56 entries which fit into the 7kb that follow the protective MBR header and GPT header. Linux understands such GPT but some tools refuse it since it does not adhere to the standard. YMMV

Boot Script

Manual_build_howto#boot.cmd

Rootfs

Here this depends on what distribution you want to install. Which partition layout you use does not matter much since the root device is passed to the kernel as argument. You might need tweaks to fstab or other files if your layout does not match what the rootfs expects. As of this writing most available images use two partitions with separate /boot.

Using rootfs tarball

```
1 mount ${card}${p}2 /mnt/
2 tar -C /mnt/ -xjpf my-chosen-rootfs.tar.bz2
3 umount /mnt
```

Linaro rootfs

Linaro offers a set of different root filesystems (https://wiki.linaro.org/Platform/DevPlatform/Rootfs) . A retention policy of 30 days applies to Linaro rootfs on snapshot servers. New snapshots can be generated on request. Latest snapshots can be made from sources such as Ubuntu Build Service (https://git.linaro.org/gitweb?p=ci/ubuntu-build-service.git)

In any case, you can get the actual rootfs tarballs here (https://snapshots.linaro.org/quantal/images/) . ALIP is a minimal LXDE based desktop environment which might me useful to most allwinner users.

Using debootstrap - Debian/Ubuntu based distributions

`debootstrap` is an alternative to rootfs tarball as described in the previous section

```
# feel free to change distro to raring/saucy/.. as appropriate for later Ubuntu version or wheezy for De
distro=precise
mount ${card}${p}2 /mnt/
# you can add --variant=buildd to install a compiler in your chroot at debootstrap time but using apt is
debootstrap --arch=armhf --foreign $distro /mnt/
cp /usr/bin/qemu-arm-static /mnt/usr/bin/
chroot /mnt
/debootstrap/debootstrap --second-stage
```

```
# for Ubuntu
cat <<EOT > etc/apt/sources.list
deb http://ports.ubuntu.com/ $distro main universe
deb-src http://ports.ubuntu.com/ $distro main universe
deb http://ports.ubuntu.com/ $distro-security main universe
deb-src http://ports.ubuntu.com/ $distro-security main universe
```

```
deb http://ports.ubuntu.com/ $distro-updates main universe
deb-src http://ports.ubuntu.com/ $distro-updates main universe
EOT
```

```
# for Debian
cat <<EOT > etc/apt/sources.list
deb http://http.debian.net/debian $distro main contrib non-free
deb-src http://http.debian.net/debian $distro main contrib non-free
deb http://http.debian.net/debian $distro-updates main contrib non-free
deb-src http://http.debian.net/debian $distro-updates main contrib non-free
deb http://security.debian.org/debian-security $distro/updates main contrib non-free
deb-src http://security.debian.org/debian-security $distro/updates main contrib non-free
EOT
```

```
cat <<EOT >> etc/fstab
none    /tmp    tmpfs   defaults,noatime,mode=1777 0 0
# if you have a separate boot partition
${card}${p}1    /boot   vfat defaults 0 0
EOT
exit
cp /etc/resolv.conf /mnt/etc
chroot /mnt
export LANG=C
apt-get update
# set up 'apt
cat <<END > /etc/apt/apt.conf.d/71-no-recommends
APT::Install-Recommends "0";
APT::Install-Suggests "0";
END
# set up locales (Debian) - dpkg scripts tend to complain otherwise
apt-get install locales
dpkg-reconfigure locales
# Choose en_US.UTF-8 for both prompts, or whatever you want.
export LANG=en_US.UTF-8
# install your favourite packages here
apt-get install nvi ; apt-get remove nano
# set root password - otherwise you won't be able to log in
passwd
exit
# cleanup
rm /mnt/usr/bin/qemu-arm-static /mnt/etc/resolv.conf
# ensure hostname has not carried over from your build host
echo something > /mnt/etc/hostname
# enable serial console (Debian/sysvinit way)
echo T0:2345:respawn:/sbin/getty -L ttyS0 115200 vt100 >> etc/inittab
# enable serial console (Ubuntu/upstart way - TBD)
umount /mnt
```

Kernel modules

When you have copied rootfs to your card you might want to copy the kernel modules as well.

Troubleshooting

- **check partitioning** - if you did the partitioning yourself read back the layout with sfdisk in sectors. sfdisk and gparted sometimes apply weird rounding when using megabytes.

```
sfdisk -uS -d /dev/sdd
# partition table of /dev/sdd
unit: sectors

/dev/sdd1 : start=     2048, size= 16150528, Id=83
/dev/sdd2 : start=        0, size=        0, Id= 0
/dev/sdd3 : start=        0, size=        0, Id= 0
/dev/sdd4 : start=        0, size=        0, Id= 0
```

- **re-check that you have written the image correctly** Check image checksum if provided. Re-read writing instructions carefully. Try another writing method if available - dd/phoenixsuit/win32-diskimage. Especially writing on Windows tends to cause trouble. If your board is new you can try an image for similar board with the same CPU. Use console cable if you have one to check the boot messages.

- **power off the board completely before booting** If you are using a console cable the board may not power off completely. There is a possiblility that self-powered USB peripherials or USB hubs may cause sililar issue. The red power light would get dimmer when the board is off but does not turn off completely. In this case the mmc controller may not get reset properly and the board boots from nand. Power off the board, disconnect all peripherials, and disconnect the serial console cable. Try booting again. You can re-connect your peripherials before booting. This issue does not seem to happen when the kernel powers down the mmc controller properly but is common when the kernel crashes.

- **check for bad micro-SD card contact** This is common issue on boards that use micro-SD socket. Try removing and re-inserting the card, cleaning the contacs on the card and dusting off the SD card socket. Some people report that inserting the card together with a piece of paper improves contact and allows booting cards which are too loose in the socket.

See also

U-Boot

- U-Boot

External

- Additional info on sunxi's flavor of U-Boot (https://github.com/linux-sunxi/u-boot-sunxi/wiki)

Categories: Software | Tutorial
