snickerdoodle

GETTING STARTED GUIDE

SEPTEMBER 9, 2016

How to Read this Document

This document makes extensive use of links, references and notices in the page margins to detail additional information that can be useful while following the guide.

WARNING A warning notice indicates a potential hazard. If care is not taken to adhere to the safety precautions, damage may be done to snickerdoodle.

Warnings and cautions will be clearly visible in either the body of the text or in the margin and must be paid close attention while following the guided steps.

CAUTION A caution indication denotes a process that requires special attention. If the caution is not exercised and the process not adhered to, failure may result and/or potential damage to snickerdoodle.

Warning, caution and informational notices, such as this one, may also be found in the margin.

Keywords

Keywords and important terms are shown in *italicized* type. Additional important information can be found in the margins of text with superscript notation[1].

Navigation of menus and directories are shown using ***bold italicized*** type. Any hierarchical navigation is shown using an arrow to denote a ***Parent*** → ***child*** relationship.

`Teletype` text is used to highlight inputs, variables and system files within the host environment.

[1] Margin notes, such as this one, reference the body content and highlight technical details or references for further information.

Introduction

This guide is intended to get you started running Linux (Ubuntu port provided by Linaro[2] and developed by krtkl) on snickerdoodle. This guide uses programs and utilities on a Linux host computer (or virtual machine) to install a complete Linux system that snickerdoodle will boot from a microSD card. A Linux (krtkl recommends the latest LTS release from Ubuntu[3]) system with the ability to mount a microSD card (UHS Speed Class 3 recommended) is required to install the Linux system.

[2] http://www.linaro.org

[3] http://www.ubuntu.com/download/desktop/

Figure 1: SD Card UHS Speed Class Marking (adapted from https://www.sdcard.org/consumers/speed/speed_class/)

Formatting SD Card

Locating SD Card on System

The `mount` command can be used to locate the SD card device, once it has been connected to the host computer. In the example below, an SD card has been connected on `/dev/sdb1` and mounted at `/media/user/UNTITLED`.

```
$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
...
/dev/sdb1 on /media/user/UNTITLED type vfat
    (rw,nosuid,nodev,uid=1000,gid=1000,shortname=mixed,dmask=0077,
    utf8=1,showexec,flush,uhelper=udisks2)
```

Partitioning the SD Card (`fdisk`)

Before partitioning the SD card, any partitions that have been mounted on the system must be unmounted. This can be done using the `umount` command.

```
$ umount /dev/sdb1
```

Once the SD card has been located, we can partition it for the Linux system (*BOOT* partition) and root filesystem (*ROOTFS* partition) using `fdisk`[4]. `fdisk` must be run with root permissions (`sudo`) using the disk parent as the argument (do not use the parition number in the argument).

[4] Additional information on `fdisk` can be found at: http://tldp.org/HOWTO/Partition/fdisk_partitioning.html

CAUTION Do NOT include any partition number (in the example case '1') when running `fdisk` on the SD card. '`/dev/sdb`' NOT '`/dev/sdb1`'

```
$ fdisk /dev/sdb
```

From within the `fdisk` interface, we can view the parition table at any time using the 'p' command. In this example, an 8GB SD card with a single FAT32 partition is being used and will be re-partitioned for snickerdoodle.

```
Command (m for help): p

Disk /dev/sdb: 7969 MB, 7969177600 bytes
255 heads, 63 sectors/track, 968 cylinders, total 15564800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            8192    15564799     7778304    b  W95 FAT32
```

BOOT Partition

First, a partition must be allocated for the Linux system binaries and files. This includes `BOOT.bin` (FSBL, bitstream, U-Boot), `uEnv.txt`, `devicetree.dtb`, and the Linux kernel `uImage`. The partition size for these files is recommended to be 128MB in size which translates to an additional 262144, 512 byte sectors.

```
Command (m for help): d
Selected partition 1

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-15564799, default 2048): <RETURN>
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-15564799, default
    15564799): +262144
```

The default partition type for `fdisk` is **Linux** (type ID 83). The *BOOT* partition

needs to be formatted as **FAT32** (type ID 'C'). To do this, the 't' command is used:

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

ROOTFS Partition

Second, a partition for the root filesystem must be created. This partition will be formatted as a **Linux** type using type ID 83. This is the default partition type.

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2): <RETURN>
Using default value 2
First sector (264193-15564799, default 264193): <RETURN>
Using default value 264193
Last sector, +sectors or +size{K,M,G} (264193-15564799, default
    15564799): <RETURN>
Using default value 15564799
```

Before writing the parition table, you should verify the partition layout by printing it with the 'p' command. In this example, an 8GB SD card has been partitioned with a 128MB FAT32 *BOOT* partition and the rest allocated for a Linux *ROOTFS* partition.

> *Always check the partition table before attempting to write it to the disk, using the 'p' command.*

```
Command (m for help): p

Disk /dev/sdb: 7969 MB, 7969177600 bytes
255 heads, 63 sectors/track, 968 cylinders, total 15564800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
```

```
/dev/sdb1            2048      264192      131072+   c  W95 FAT32
    (LBA)
/dev/sdb2          264193    15564799     7650303+  83  Linux
```

Once the partition table has been verified, the 'w' command can be used to write the table to the disk:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Formatting Partitions (mkfs/mke2fs)

With a partitioned SD card, the partitions need to be formatted with the necessary filesystem type. For the *BOOT* partition, the filesystem type is **VFAT**. Formatting the *BOOT* partition can be done using the `mkfs.vfat`[5]. To format a FAT32 filesystem on `/dev/sdb1` with a 'BOOT' disk label, the following command can be used:

[5] Additional information on `mkfs/mke2fs` and it's front end tools can be found at: http://www.tldp.org/HOWTO/Partition/formatting.html

```
$ mkfs.vfat -n BOOT /dev/sdb1
```

The format for the *ROOTFS* partition can be done with `mke2fs` which will format a Linux partition with an ext2/ext3/ext4 filesystem. To format an ext4 filesystem on `/dev/sdb2` with a block size of 1k (1024) and a 'ROOTFS' disk label, the following command can be used:

```
$ mke2fs -b 1024 -t ext4 -L ROOTFS /dev/sdb2
```

CAUTION When formatting disk partitions, make sure the disk partitions are NOT mounted.

If the formatting is successful, the following output with be written to the console (writing superblocks and filesystem accounting information can take some time depending on the size and speed of the SD card):

```
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=ROOTFS
OS type: Linux
Block size=4096 (log=0)
Fragment size=4096 (log=0)
Stride=0 blocks, Stripe width=0 blocks
478208 inodes, 7650300 blocks
382515 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=74973184
934 block groups
8192 blocks per group, 8192 fragments per group
512 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409,
        663553,
    1024001, 1990657, 2809857, 5120001, 5971969

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

After the partitions have been properly formatted, the SD card must be ejected and re-connected before moving the Linux boot components and root filesystem contents to the disk.

```
$ eject /dev/sdb
```

Sources

BOOT Partition Components

The latest *BOOT* partition Linux components for Snickerdoodle and Snickerdoodle Black can be downloaded using `git`.

```
$ git clone
    https://github.com/krtkl/snickerdoodle-linux-prebuilt.git
```

Alternatively, the sources can be downloaded directly from a web browser from the krtkl GitHub page[6] as a `.ZIP` file.

[6] https://github.com/krtkl/snickerdoodle-linux-prebuilt

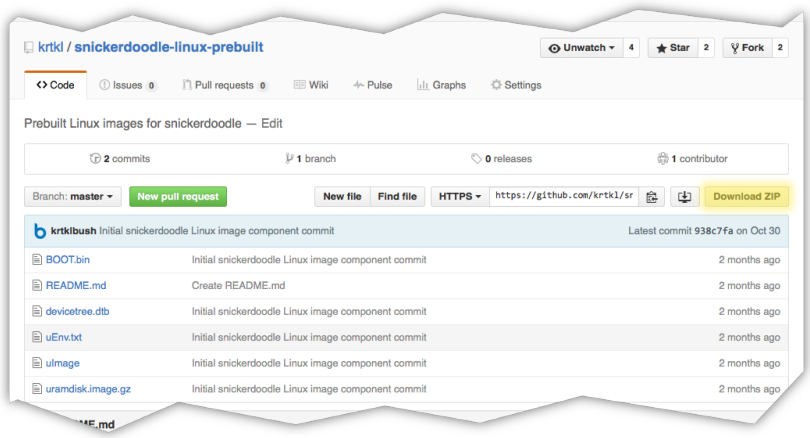

Figure 2: Download prebuilt Linux components from GitHub

Copy Files to SD Card

After downloading the SD card components
The *BOOT* components can be installed onto the SD card using the `cp` command, starting with `BOOT.bin`. In this example, the files are copied to the *BOOT* partition that has been mounted at `/media/user/BOOT`. As stated above, the `mount` command can be used to locate the mount point of the SD card partitions.

```
$ cp BOOT.bin /media/user/BOOT
$ cp uEnv.txt /media/user/BOOT
$ cp devicetree.dtb /media/user/BOOT
$ cp uImage /media/user/BOOT
```

After copying the *BOOT* components, the `sync` command should be used to make sure the system buffers have been flushed and the process of writing the files to the SD card is complete.

```
$ sync
```

ROOTFS Sources

The root filesystem can be extracted directly into the *ROOTFS* partition using the '-C' argument when extracting the archive contents. An Ubuntu 14.04 filesystem can be downloaded from `http://krtkl.com/downloads/`. In this

example, the *ROOTFS* partition is mounted at `/media/user/ROOTFS`, which should be checked before attempting to extract the root filesystem. The root filesystem contains a lot of large packages (ROS, python, etc.) and may take several minutes to complete the process of writing to the SD card.

```
$ tar -C /media/user/ROOTFS -xvzf snickerdoodle-ubuntu-14.04.tar.gz
```

After extracting the root filesystem to the SD card, use the `sync` command to flush the system buffers and ensure the write process is complete. Additionally, making sure to unmount the SD card partitions before ejecting will make certain that any lingering write processes have completed before the SD card is removed.

```
$ sync
$ umount /dev/sdb1
$ umount /dev/sdb2
$ eject /dev/sdb
```

Now that the microSD card has been partitioned, formatted and populated with the Linux files, it is ready to be installed onto snickerdoodle and booted. Install the microSD card into the card cage (J6) and connect power on either the micro USB connector (J1) or the power pins on J2. The familiar Ubuntu boot console should start and you will have access to a terminal to begin configuration and development of snickerdoodle.

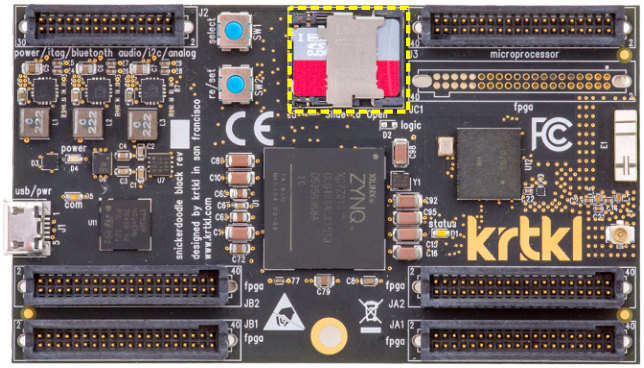

Figure 3: snickerdoodle SD Card Connector with Installed microSD Card

```
Welcome to Ubuntu 14.04 (GNU/Linux 3.14.0-xilinx-dirty armv7l)

 * Documentation:  http://www.ubuntu.com
root@snickerdoodle:~/workspace# python hello_world.py
Congratulations! You've run your first python script on snickerdoodle!
```