Lab 1: Introduction to Gumstix and Code Optimization - Verdex Setup

18–349 Embedded Real-Time Systems

September 15, 2012

Creating a bootable microSD card.

What you will need?

1. A development machine with some version of linux (I had Ubuntu 12.04 installed on my machine) . You should have root access on this machine (or at least the ability to `mount/unmount` devices and run `fdisk`).
2. A microSD card adapter. Your hardware should contain an adapter that converts microSD card into an SD card which you can use on your computer. If your computer/laptop does not have an SD card reader, please contact the course staff to get a USB adapter for the microSD card.

The example covered in this document will show the steps for setting up a brand new 2GB microSD card.

First insert your card into your development machine's flash card slot.

On my Ubuntu 12.04 machine, the newly inserted card shows up as `/dev/mmcblk0` (with any partitions showing up as `/dev/mmcblk0p1`, `/dev/mmcblk0p2`, etc.) and that is the device name that will be used through this example. You should substitute the proper device name for your machine. You can use '`mount`' or '`df`' to see where the card mounts on your machine.

Step 1

Unmount the device's existing file system (all possible partitions) before we get started with `fdisk`

```
$ sudo umount /dev/mmcblk0*
```

Partitioning the card

Step 2

Now launch `fdisk` and create an empty partition table. Note that the argument for `fdisk` is the entire device (`/dev/mmcblk0`) not just a single partition (`i.e. /dev/mmcblk0p1`):

```
$ sudo fdisk /dev/mmcblk0
Command (m for help): o
Building a new DOS disklabel with disk identifier 0x29745232.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
```

```
Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)
```

Step 3

First look at the current card information:

```
Command (m for help): p

Disk /dev/mmcblk0: 1977 MB, 1977614336 bytes
4 heads, 16 sectors/track, 60352 cylinders, total 3862528 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: 0x29745232

  Device Boot      Start         End      Blocks   Id  System
```

Note the card size in bytes. We will need it later in the process.

Step 4

Now go into ”Expert” mode:

```
Command (m for help): x
```

Step 5

Next we will set the geometry to 255 heads, 63 sectors and a calculated value for the number of cylinders required for the particular microSD card.

To calculate the number of cylinders, we take the 1977614336 (replace this number by the number reported by your `fdisk` command) bytes reported above by `fdisk` divided by 255 heads, 63 sectors and 512 bytes per sector:

1977614336 / 255 / 63 / 512 = 240.43 which we round down to 240 cylinders.

Step 6

Set the number of heads, sectors and cylinders

```
Expert command (m for help): h
Number of heads (1-256, default 4): 255

Expert command (m for help): s
Number of sectors (1-63, default 62): 63

Expert command (m for help): c
Number of cylinders (1-1048576, default 984): 247
```

Step 7

Return to `fdisk's` main mode and create a new partition `32 MB FAT` partition:

```
Expert command (m for help): r
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-3862527, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-3862527, default 3862527): +32M
```

Step 8

Change the partition type to `FAT32`:

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

And mark it bootable:

```
Command (m for help): a
Partition number (1-4): 1
```

Step 9

Next we create an `ext3` partition for the rootfs.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First sector (67584-3862527, default 67584): 67584
Last sector, +sectors or +size{K,M,G} (67584-3862527, default 3862527): 3862527
```

Step 10

To verify our work, lets print the partition info

```
Command (m for help): p

Disk /dev/mmcblk0: 1977 MB, 1977614336 bytes
255 heads, 63 sectors/track, 240 cylinders, total 3862528 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: 0x29745232

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1   *        2048       67583       32768    c  W95 FAT32 (LBA)
/dev/mmcblk0p2           67584     3862527     1897472   83 Linux
```

Step 11

Up to this point no changes have been made to the card itself, so our final step is to write the new partition table to the card and then exit.

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

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x partitions, please
see the fdisk manual page for additional information.

Syncing disks.
```

Formatting the new partitions

Step 12

We format the first partition as a FAT file system (the -n parameter gives it a label of FAT, you can change or omit this if you like)

```
$ sudo mkfs.vfat -F 32 /dev/mmcblk01 -n FAT
mkfs.vfat 2.11 (29 Oct 2011)
WARNING: Not enough clusters for a 32 bit FAT!
```

Step 13

We format the second partition as an `ext3` file system

```
$ sudo mkfs.ext3 /dev/mmcblk02
mke2fs 1.42 (29-Nov-2011)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
118800 inodes, 474368 blocks
23718 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=486539264
15 block groups
32768 blocks per group, 32768 fragments per group
7920 inodes per group
Superblock backups stored on blocks:
```

```
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Installing the boot files

There are 2 files required on the first (FAT) partition to boot your Verdex-Pro

1. gumstix-factory.script
2. uimage: the linux kernel

Step 14

Mount the FAT partition of your microSD card. To do this, you must create an empty directory to mount it to (it can be removed afterward, if you like). I used the `/media/card` directory by first issuing the command:

```
$ sudo mkdir /media/card
```

This example will assume that you have mounted it at `/media/card`

```
$ sudo mount /dev/mmcblk01 /media/card
$ sudo cp gumstix-factory.script /media/card/
$ sudo cp uimage /media/card/
$ sync; sync; sync;
```

The `sync` command will flush any of the buffered writes to the microSD card (the command may take a few seconds to execute).

Step 15

You can now unmount the FAT partition:

```
$ sudo umount /dev/mmcblk01
```

At this point you have a bootable FAT partition.

Step 16

The final step is to untar your desired rootfs onto the `ext3` partition that you created above.

Note that this step can be dangerous. You do not want to untar your verdex-pro rootfs onto your development machine by mistake. Be careful!

This example will assume that you have mounted it at `/media/card`

```
$ sudo mount /dev/mmcblk02 /media/card
```

Now untar your desired rootfs:

```
$ sudo tar -xvf /path/to/rootfs.tar.gz -C /media/card
$ sync; sync; sync;
```

The sync command will flush any of the buffered writes to the microSD card (the command may take a minute to execute).

Step 17

You can now unmount the ext3 partition:

```
$ sudo umount /dev/mmcblk02
```

Remove the card from the adapter and insert it in the verdex-pro's card slot. Initiate a serial connection with the board. Apply power to the hardware board. If you see a message saying that the system is booting from the microSD card, Congratulations!! If not, you probably made an error somewhere. Try once again and if not successful, contact your TA or me priya@cs.cmu.edu. Once the system boots up of the microSD card, login using "root" as userid and "gumstix" as password. Write a simple C program and see if gcc works.
References:

1. Gumstix developer's site.
   http://gumstix.org/create-a-bootable-microsd-card/69-create-a-bootable-microsd-card-old.html