Creating a bootable microSD card

The Overo COM will boot directly from a properly prepared microSD card. This section will outline how to partition and format a bootable microSD card.

In order to create a bootable microSD compatible with the OMAP3 boot ROM, you set a special geometry using the fdisk "Expert mode".

Important Notes:

- When creating a bootable microSD card for an Overo COM, you MUST use expert mode. This will set the correct cylinders, heads and sectors for the microSD card before you partition it.

- Gumstix recommends the use of a microSDHC card.

This example below will show the steps required to set up a new 2GB microSD card.

First insert your card into your development machine's flash card slot. You may need to use a microSD to SD card adaptor to fit your slot.

On my Ubuntu 8.04 machine, the newly inserted card shows up as /dev/sde 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.

Let's unmount the device's existing file system before we get started with fdisk:

```
$ sudo umount /dev/sde1
```

PARTITIONING THE CARD

Now launch fdisk and create an empty partition table. Note that the argument for fdisk is the entire device (/dev/sde) not just a single partition (i.e. /dev/sde1):

```
# sudo fdisk /dev/sde
Command (m for help): o
Building a new DOS disklabel. 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)
```

Let's first look at the current card information:

```
Command (m for help): p
Disk /dev/sde: 2032 MB, 2032664576 bytes
```

```
64 heads, 63 sectors/track, 984 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes
Disk identifier: 0x00aa8e5c
Device Boot      Start         End      Blocks   Id  System
```

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

Now go into "Expert" mode:

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

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 2032664576 bytes reported above by fdisk divided by 255 heads, 63 sectors and 512 bytes per sector:

2032664576 / 255 / 63 / 512 = 247.12 which we round **down** to 247 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
Warning: setting sector offset for DOS compatiblity
Expert command (m for help): c
Number of cylinders (1-1048576, default 984): 247
```

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 cylinder (1-247, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-247, default 15): +32M
```

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
```

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 cylinder (6-247, default 6): 6
Last cylinder or +size or +sizeM or +sizeK (6-247, default 247): 247
```

To verify our work, lets print the partition info:

```
Command (m for help): p
Disk /dev/sde: 2032 MB, 2032664576 bytes
255 heads, 63 sectors/track, 247 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00aa8e5c
Device Boot      Start         End      Blocks   Id  System
/dev/sde1   *           1           5       40131    c  W95 FAT32 (LBA)
/dev/sde2               6         247     1943865   83  Linux
```

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

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/sde1 -n FAT
mkfs.vfat 2.11 (12 Mar 2005)
```

We format the second partition as an ext3 file system:

```
$ sudo mkfs.ext3 /dev/sde2
mke2fs 1.40.8 (13-Mar-2008)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
121920 inodes, 485966 blocks
24298 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=499122176
15 block groups
32768 blocks per group, 32768 fragments per group
8128 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: ^[done
This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

INSTALLING THE BOOT FILES

There are three files required on the first (FAT) partition to boot your Overo:

1. MLO: the boot-loader loader - this small program is loaded into the OMAP3 processor's static RAM. It does some minimal configuration of system memory and io pins and then loads the second file.
2. u-boot.bin: the boot loader
3. uImage: the linux kernel

You can build these yourself or download pre-built images. It is important that these three files have precisely these names.

Once you have completed building or downloading these files, mount the FAT partition of your microSD card. This example will assume that you have mounted it at /media/card:

```
sudo mount /dev/sde1 /media/card
```

Due to constraints of the mask boot ROM in the OMAP processor, MLO should be written first:

```
$ sudo cp MLO-overo /media/card/MLO
```

Then copy u-boot and the linux kernel to the card:

```
$ sudo cp u-boot-overo.bin /media/card/u-boot.bin
$ sudo cp uImage-overo.bin /media/card/uImage
```

You can now unmount the FAT partition:

```
$ sudo umount /dev/sde1
```

At this point you have a bootable FAT partition.

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 Overo rootfs onto your development machine - be careful!

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

```
$ sudo mount /dev/sde2 /media/card
```

Now untar your desired rootfs:

```
$ cd /media/card
$ sudo tar xvjf /path/to/omap3-console-image-overo.tar.bz2
```

You can now unmount the ext3 partition:

```
$ sudo umount /dev/sde2
```

Top Go back to "Downloading pre-built images" section Continue to "Writing images to onboard nand" section

Tags