Bootstrapping Debian

Vincent Sanders <support@simtec.co.uk>

Daniel Silverstone

Ben Dooks

2007-11-08

Revision History

Revision 1.00	8th November 2007	VRS
	Initial Release.	
Revision 1.10	11th January 2008	VRS
	Minor editorial updates.	

Table of Contents

This document describes how to bootstrap the Debian operating system using the debootstrap tool.

1. Introduction

It is sometimes desirable to install an instance of the Debian Operating System (OS) using a system other than the target. This is generally done for one of three reasons:

- The target system has no local storage and operation over the network is desirable, often referred to as "NFS root"
- The target system has no sensible way of installing the system itself so a disc must be prepared for it by another system.
- The system is emulated and expedience demands the host system perform the bootstrap itself.

The procedure for creating the target system is constructed from eight steps:

1. Acquire the necessary tools.
2. Construct the root filesystem.
3. Bootstrap basic system.
4. Minimal configuration of the basic system.
5. Boot the basic system single user.
6. Complete the bootstrap
7. Final system configuration.
8. Bootloader configuration.

These steps are largely similar for each of the installation targets, some details do however differ, these differences will be specifically described where appropriate.

2. Acquire the necessary tools

The primary assumption is that the bootstrap will be performed on a Unix like system. This implies that there will be a functioning bourne shell (/bin/sh) and basic shell tools such as **ar**, **cat** etc.

It *may* be possible to use these instructions on a windows system using the cygwin tools but no attempt to verify this has been made.

2.1. wget

The **wget** tool is extremely common and will probably already be present on most systems.

To obtain the **wget** tool on a deb based system (E.g. Debian or Ubuntu) install the wget package using:

```
sudo apt-get install wget
```

To obtain the **wget** tool on an rpm based system (E.g. Fedora) install the wget package using:

```
yum install wget
```

2.2. debootstrap

The **debootstrap** command is the primary tool required to create the target system. It obtains all the necessary packages from a Debian package repository and unpacks them in the correct way. The tool is based on standard UNIX shell functionality which makes it portable and straightforward to use.

To obtain the **debootstrap** tool on an deb based system (E.g. Debian or Ubuntu) install the debootstrap package using:

```
sudo apt-get install debootstrap
```

To obtain the **debootstrap** tool on an RPM based system (E.g. Fedora) is more complicated as there is generally no RPM package available. The following steps will manually install the Debian package onto a RPM based system. Firstly create a temporary working directory:

```
$ mkdir work
$ cd work
```

Obtain the debootstrap package from the Debian archive. The archive pool contains the .deb package files to obtain, care should be taken to use the most recent version from the pool. The archive pool is available from numerous mirrors [http://www.debian.org/mirror/list]. To obtain the package from the master repository:

```
$ wget http://ftp.debian.org/debian/pool/main/d/debootstrap/debootstrap_1.0.7_all.deb
```

Once the correct package has been obtained it is installed by unpacking the Debian package with the **ar** tool and using **tar** to extract debootstrap. The **tar** operation will require suitable privileges.

```
$ ar -xf debootstrap_1.0.7_all.deb
$ TARFILE=$(pwd)/data.tar.gz
$ cd /
$ zcat < ${TARFILE} | tar xv
```

This will place the debootstrap tool in /usr/sbin which must be in the users path for the tool to operate correctly.

2.3. NFS

If the system is to be accessed by NFS the server must have the tools installed.

To install the NFS server and tools on a deb based system (E.g. Debian or Ubuntu) install NFS server package using:

```
sudo apt-get install nfs-kernel-server
```

To install the NFS server and tools on an RPM based system (E.g. Fedora) install the NFS utilities package using:

```
yum install nfs-utils
```

3. Construct the root filesystem

This step creates a suitable mounted filesystem ready to bootstrap into. Three targets are discussed here correspond to the three installation methods described earlier.

3.1. NFS root

This method is the simplest of the three and consists of creating a directory and configuring the NFS server to serve it.

This method requires that suitable networking is available and configured. It is common for the serving machine to also be configured to serve as a DHCP server. A full explanation of configuring the networking is not presented here but it is necessary to know the IP addresses of the client machine and for the client machine to be able to obtain these values for NFS to operate correctly.

The first step is to create a directory on the host which will be served using the NFS protocol to the client system. The directory should be created somewhere with at least a gigabyte of space free. The directory will be served using the NFS protocol and must be somewhere suitable. A possible directory might be /export/debian. Inside a users home directory is *not* usually suitable.

```
# mkdir /export/debian
```

Once a suitable directory has been created it must be added to the NFS exports. This is achieved by adding a line to the /etc/exports configuration file. The format of the configuration line is fairly straightforward and simply lists the directory to export, the network address to export it to and options to control the export. The /etc/exports file format and options is detailed in the exports manual page. The IP address or the network range that will be assigned to the client must be known.

A suitable alteration to the /etc/exports file for a network with the IP addresses 192.168.1.xx would be:

```
$ echo '/export/debian 192.168.1.0/24(rw,no_root_squash,async)' >>/etc/exports
```

Once altered the new export can be enabled with the **exportfs** command

```
$ exportfs -a
```

The export can be checked with

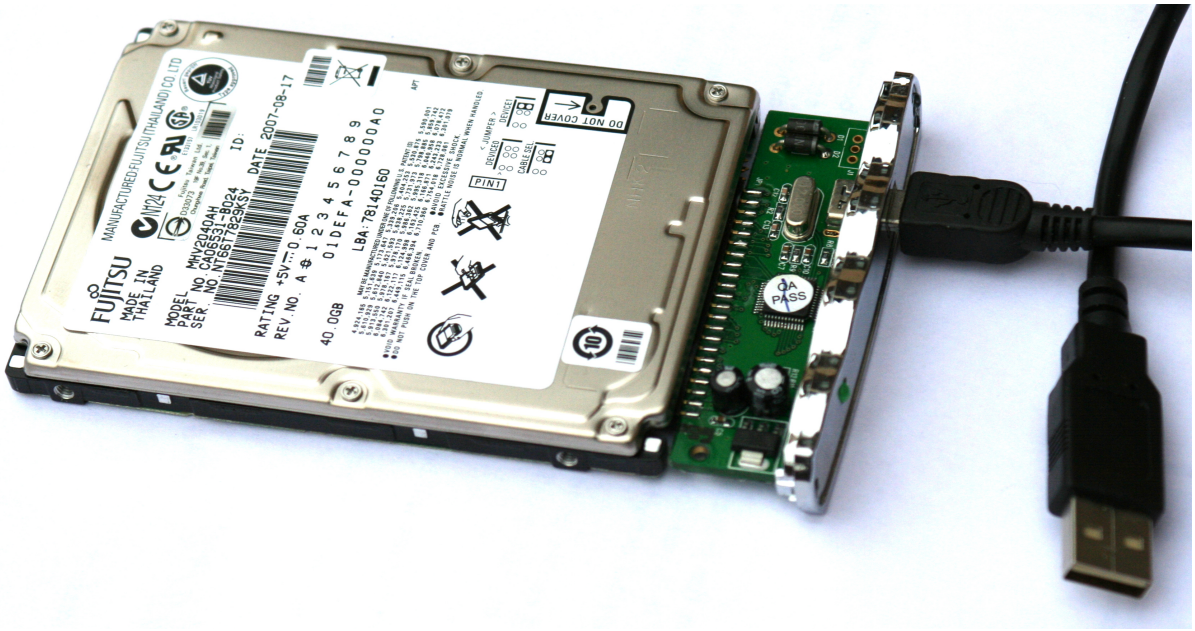
```
$ exportfs
/export/debian    192.168.1.0/255.255.255.0
```

3.2. Disc drive

This method is useful for platforms where a hard drive is practical but there is no easy way to perform a standard OS install using the Debian installer.

Firstly a suitable hard drive of at least ten gigabytes size must be acquired. For most Simtec Electronics platforms this would involve using a 2.5 inch hard drive connected using a standard 44way cable. One straightforward way to connect these hard drives to the host for bootstrapping is to use a USB connection.

Figure 1. USB connected 2.5inch hard drive



Irrespective of how the drive is connected to the host it will have a device node. This can usually be found in the system message log, this can typically be acquired using the **dmesg** command.

Example 1. Dmesg output after connecting a USB attached hard drive

```
usb 5-5: new high speed USB device using ehci_hcd and address 12
usb 5-5: configuration #1 chosen from 1 choice
scsi5 : SCSI emulation for USB Mass Storage devices
usb-storage: device found at 12
usb-storage: waiting for device to settle before scanning
usb-storage: device scan complete
scsi 5:0:0:0: Direct-Access     Generic  USB Disk         9.02 PQ: 0 ANSI: 2
sd 5:0:0:0: [sdd] 78140160 512-byte hardware sectors (40008 MB)
sd 5:0:0:0: [sdd] Write Protect is off
sd 5:0:0:0: [sdd] Mode Sense: 03 00 00 00
sd 5:0:0:0: [sdd] Assuming drive cache: write through
 sdd: unknown partition table
sd 5:0:0:0: [sdd] Attached SCSI disk
sd 5:0:0:0: Attached scsi generic sg3 type 0
```

This shows the output from the dmesg command after connecting a USB attached hard drive. The device name can clearly be seen (sdd) and that the disc has no current partition table.

Warning

Performing these actions will result in the loss of all data on the selected drive. The user *must* ensure they are using the correct device nodes and no liability for direct or indirect data loss from following these instructions will be accepted.

Once a drive is available it must be partitioned. The chosen layout is largely dependant on usage. The simplest layout is one large root partition and a swap partition of a suitable size (128MB to 256MB). If ABLE is being used to start the OS from the drive there are no limits on the partitioning scheme as long as a DOS disc label is used and the root filesystem lies within the first 128GB. A separate boot partition can be used if desired.

Example 2. Partitioning a USB attached hard drive with parted.

The example shown here partitioning of a USB drive into a single large root partition and a small (128MB) swap partition.

```
parted /dev/sdd
GNU Parted 1.7.1
Using /dev/sdd
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel msdos
(parted) unit mb
(parted) print free

Disk /dev/sdd: 40008MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End      Size     Type  File system  Flags
        0.00MB  40008MB  40008MB        Free Space

(parted) mkpart primary ext2 0 39880
(parted) mkpartfs primary linux-swap 39880 40008
(parted) print

Disk /dev/sdd: 40008MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start    End      Size     Type     File system  Flags
```

```
 1      0.00MB   39880MB  39880MB  primary
 2      39880MB  40008MB  128MB    primary  linux-swap

(parted) quit
```

The **parted** command can be driven non interactively and the the above could also be performed with a single command line (assuming the partition sizes are already known)

```
parted /dev/sdd mklabel msdos mkpart primary ext2 0 39880 mkpartfs primary linux-swap 39880
```

Once a partition table has been written the device will be rescanned and the newly created partitions will be available. The partition names can typically be retrieved using **dmesg** command once more.

Example 3. Dmesg output after partitioning a USB attached hard drive

```
sd 5:0:0:0: [sdd] Attached SCSI disk
sd 5:0:0:0: Attached scsi generic sg3 type 0
sd 5:0:0:0: [sdd] 78140160 512-byte hardware sectors (40008 MB)
sd 5:0:0:0: [sdd] Write Protect is off
sd 5:0:0:0: [sdd] Mode Sense: 03 00 00 00
sd 5:0:0:0: [sdd] Assuming drive cache: write through
 sdd: sdd1 sdd2
```

This shows the output from the dmesg command after partitioning a USB attached hard drive. The device names for the new partitions can clearly be seen (sdd1 and sdd2).

Once suitable partitions have been created a filesystem must be placed on them. The filesystem must be readable by the bootloader so a kernel can be retrieved from it. ABLE supports several filesystems but the EXT2 system is best for this application.

```
$ mke2fs -O dir_index,has_journal,sparse_super  /dev/sdd1
mke2fs 1.40.2 (12-Jul-2007)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
4833280 inodes, 9663996 blocks
483199 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
295 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624

Writing inode tables: done
Creating journal (32768 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.
```

The parameters of the created filesystem can be found with the **tune2fs** command. The UUID of the filesystem should be noted as it will be required later in the bootstrap process.

```
$ tune2fs -l /dev/sdd1
tune2fs 1.40.2 (12-Jul-2007)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          9669e340-5254-4cb1-a2d8-fc4ec0d0835c
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal resize_inode dir_index filetype needs_recovery sp
Filesystem flags:         signed directory hash
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              4833280
Block count:              9663996
Reserved block count:     483199
Free blocks:              9467258
Free inodes:              4833269
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1021
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         16384
Inode blocks per group:   512
Filesystem created:       Fri Dec  7 11:38:55 2007
Last mount time:          Fri Dec  7 13:35:27 2007
Last write time:          Fri Dec  7 13:35:27 2007
Mount count:              1
Maximum mount count:      36
Last checked:             Fri Dec  7 11:38:55 2007
Check interval:           15552000 (6 months)
Next check after:         Wed Jun  4 12:38:55 2008
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:           128
Journal inode:            8
Default directory hash:   tea
Directory Hash Seed:      e9836175-d901-4253-a630-f584d39e5fcc
Journal backup:           inode blocks
```

If a swap partition was created that should also be formatted at this point. The UUID of the swap partition should be noted as it will be required later in the bootstrap process.

```
$ mkswap /dev/sdd2
Setting up swapspace version 1, size = 423620 kB
no label, UUID=ebe8f79f-509b-4eb3-9c3a-f0e97b7a8c1c
```

Once the filesystem has been created it should be mounted somewhere convenient on the host system e.g. `/mnt`.

```
$ mount /dev/sdd1 /mnt
```

Wherever the volume is mounted this is where the basic system bootstrap will be performed.

3.3. Virtual disc

The procedure outlined here is fairly straightforward, however the Section A.1, “mkdiscimage” shell script has been provided which will perform these steps and those in sections Section 4, “Bootstrap basic system” and Section 5, “Minimal configuration of the basic system” without user interaction.

Constructing a virtual disc consists of several steps. The general approach is:

- Use the **dd** command to create a file the desired size for the virtual disc image.
- Partition the file with parted.
- Use a loopback device and the device mapper system to create suitable device nodes.
- Create a filesystem on the new device, similar to the approach taken with a physical disc.

The creation of the virtual disc image with the **dd** command is straightforward. The image should be large enough to contain the base system, 512MB is a suitable minimum size.

```
dd if=/dev/zero of=virtdisc bs=1M count=512
```

The virtual disc image must then be partitioned. This is most easily achieved with the **parted** command. The older **fdisk** command could be used but is not covered here. As with a real disc the partition layout is largely arbitrary but should consist of at least a root partition and a swap partition. The Example 2, “Partitioning a USB attached hard drive with parted.” shows how this is achieved for a physical disc.

The partitioning of the image created previously might be performed with:

```
parted --script virtdisc mklabel msdos mkpart primary ext2 0 384 mkpartfs primary linux-swap
```

This will create a 384MB root disc and a 128MB swap partition within the 512MB image.

The next step is to use the loopback mount system to create a block device from the virtual disc file. This is achieved with the **losetup** command. This operation requires superuser privileges.

```
# losetup -f
/dev/loop0
# losetup /dev/loop0 virtdisc
```

Next a device node must be made for the root partition of the virtual disc. We can achieve this using the device mapper system. The device mapper requires us to know the start and length of the partition on the loopback device. This information is available in the partition table but is not easy to calculate by hand. To aid with this the **lspartab** script has been provided (see Section A.2, “lspartab”). When run on the loopback device node created in the previous steps two partitions are listed:

```
# lspartab /dev/loop0
1 32 749920 131
2 749952 250112 130
```

the first partition is the root partition (type 131), the second is the swap (type 130).

The device mapper requires a “table_file” to construct the device. A “table_file” is a sequence of values specific to the mapping being performed. A linear mapping is required and the entries of the appropriate “table_file” are:

The Logical start sector
Total number of sectors
The mapping type “linear”
The destination device e.g. `/dev/loop0`
The start sector on the destination device
For the root partition a linear mapping of the loop device is required. The mapped device in this case should always start at logical sector 0. From this a **dmsetup** command can be constructed.

For the previously listed partition table of the virtual disc a linear mapping from sector 32 for 749920 sectors, where a sector is 512 bytes is indicated. The command would be:

```
# dmsetup create virtdisk1 --table "0 749920 linear /dev/loop0 32"
```

If this is successful it will create a new device node `/dev/mapper/virtdisk1` that can be used as if it were a real partition on a real disc.

If the device mapping has been successful the newly created device can be used to create a filesystem upon the same as with a physical disc. The steps to create a filesystem and get its UUID can be found in the previous section.

Once the filesystem has been created it should be mounted somewhere convenient on the host system e.g. `/mnt`.

```
$ mount /dev/mapper/virtdisk1 /mnt
```

Wherever the volume is mounted this is where the basic system bootstrap will be performed.

4. Bootstrap basic system

The bootstrap phase is simple but requires additional pieces of information, the suite to install and the mirror to obtain packages from.

The Debian suite indicates which OS version to install, Debian releases are code named e.g. sarge, etch, lenny, sid. The current stable release is etch but the Debian website [http://www.debian.org/] will always indicate the latest release.

The package archive is available from numerous mirrors [http://www.debian.org/mirror/list] and the closest one available should be used.

The bootstrap command when executed will obtain all the packages from the archive and unpack them as appropriate. This step may take some time depending on speed of access to the archive mirror and the media being installed upon.

The **debootstrap** command requires several parameters. The first is `--arch arm` which specifies that the bootstrap target is an ARM system. The next is the `--foreign` because the host architecture is not the same as the target. Next is the `suite` to install followed by the `root filesystem` as created in the first stage and finally the package archive source.

Example 4. Bootstrapping the ARM etch suite onto a USB attached hard drive.

```
debootstrap --arch arm --foreign etch /mnt http://mirror/debian
I: Retrieving Release
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Checking component main on http://mirror/debian...
I: Retrieving adduser
I: Validating adduser
I: Retrieving apt
I: Validating apt

...

I: Extracting tar...
I: Extracting tzdata...
I: Extracting util-linux...
I: Extracting zlib1g...
```

5. Minimal configuration of the basic system

The system as bootstrapped is not complete, it lacks a kernel and some initial configuration. This section covers the necessary steps to get the system to a bootable state, the examples assume the targets root filesystem as created in the previous step is mounted on `/mnt` which should be adjusted as appropriate.

A pre-compiled kernel is available for most Simtec Electronics systems which can run Debian. These can generally be found on the products resources section of the Simtec Electronics website. The procedure to install is simply to download the pre-built compressed tar from the website ,unpack it in the root of the bootstrapped system (not the host system), and

remove the tarball.

Example 5. Obtaining and installing a pre-built kernel.

This example obtains the pre-built 2.6.23 Linux kernel which is suitable for all boards which use the Samsung 24xx series SOC. The root filesystem is assumed to be mounted on `/mnt`.

```
$ cd /mnt
$ wget http://www.simtec.co.uk/products/SWLINUX/files/s3c24xx-linux-2.6.23-simtec1.tar.bz2
--13:29:58--  http://www.simtec.co.uk/products/SWLINUX/files/s3c24xx-linux-2.6.23-simtec1.ta
          => `s3c24xx-linux-2.6.23-simtec1.tar.bz2'
Resolving www.simtec.co.uk... 217.147.94.109
Connecting to www.simtec.co.uk|217.147.94.109|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48,086,175 (46M) [application/x-bzip2]

100%[====================================>] 48,086,175   216.28K/s    ETA 00:00

13:35:14 (148.88 KB/s) - `s3c24xx-linux-2.6.23-simtec1.tar.bz2' saved [48086175/48086175]

$ tar -jxf s3c24xx-linux-2.6.23-simtec1.tar.bz2
$ rm s3c24xx-linux-2.6.23-simtec1.tar.bz2
```

The bootstrapped system requires its host name to be set properly otherwise the host system name will be used which may cause confusion. Here we show it being set to “etch”.

```
echo "etch" > /mnt/etc/hostname
```

The basic bootstrap omits some essential device nodes specifically the console and the S3C24xx serial ports.

```
$ mknod /mnt/dev/console c 5 1
$ mknod /mnt/dev/ttySAC0 c 204 64
$ mknod /mnt/dev/ttySAC1 c 204 65
$ mknod /mnt/dev/ttySAC2 c 204 66
```

The filesystem table cannot be constructed by the bootstrap tool so must be created. This is not straightforward as the table must correctly represent the root device from the boot media. The table generally begins with a comment line explaining the line format and an entry for the process filesystem.

```
echo "# <file system> <mount point>   <type>  <options>       <dump>  <pass>" > /mnt/etc/fst
echo "proc /proc proc defaults 0 0" >> /mnt/etc/fstab
```

5.1. NFS root

For systems where the root filesystem will be mounted using NFS over the network an entry must be made in the filesystem table for the root mount, a suitable entry would be:

```
echo "192.168.1.1:/export/debian / nfs defaults 0 1" >> /mnt/etc/fstab
```

The root filesystem is now ready to be booted.

5.2. Disc Drive

For systems where the root filesystem is mounted on an attached hard drive an entry must be made in the filesystem table for the root mount. Using the partitioning outlined earlier the line would be:

```
$ echo "UUID=9669e340-5254-4cb1-a2d8-fc4ec0d0835c / ext3 defaults,errors=remount-ro 0"
$ echo "UUID=ebe8f79f-509b-4eb3-9c3a-f0e97b7a8c1c none swap sw 0 0" >> /mnt/etc/fstab
```

The root filesystem is now ready to be booted by the target system. The disc must be unmounted and attached to the target system as appropriate.

```
$ umount /mnt
```

5.3. Virtual Disc

A virtual disc is very similar to the physical disc, an entry must be made in the filesystem table for the root mount. Using the partitioning outlined earlier the line would be:

```
echo "/dev/sda1 / ext3 defaults,errors=remount-ro 0" >> /mnt/etc/fstab
echo "/dev/sda2 none swap sw 0 0" >> /mnt/etc/fstab
```

The root filesystem is now ready to be booted by the virtual machine. The filesystem should be unmounted, the device mapper device removed and the loopback mount undone.

```
# umount /mnt
# dmsetup remove /dev/mapper/virtdisk1
# losetup -d /dev/loop0
```

6. Boot system single user

The system must be booted from the bootloader as appropriate for the medium into a command line ready for the next step. Whichever method is used the system should boot the Linux kernel and present a single user root shell prompt.

6.1. NFS root

The NFS boot method means the kernel image must be retrieved from the network. Using the ABLE bootloader this is a simple operation using a commandline similar to this:

```
(tftpboot)vmlinuz-2.6.23-simtec1 init=/bin/sh root=/dev/nfs nfsroot=192.168.1.1:/export
```

The vmlinuz kernel image should be copied from the root filesystem boot directory into the TFTP server directory on the host.

The IP address of the NFS server should be substituted in the command line as well as the correct export directory. The rsize and wsize options are *important* and omitting them will result in a series of error messages and no shell running. The `ip=dhcp` will cause the kernel to obtain its IP parameters from a DHCP server, obviously this can be altered to use a fixed ip address if required. The `Documentation/nfsroot.txt` file in the kernel sources has more details.

The console argument sets the serial port as the console. The parameter sets the baud rate to 115200 which is the ABLE default.

6.2. Disc drive

The hard drive should be physically attached to the development platform and power applied. Once ABLE has started the

drive should be detected and be available. There should be a section of the ABLE boot message showing the drive detection and the partition with the root filesystem.

```
hda: ST94813A: ATA PIO mode 4
hda: Diagnosing disc drive: ok
(hda) 37GB
(hd0) on ((hda1):ext2)
(hd1) on (hda2)
```

From this the appropriate device to use to access the root filesystem can be deduced. The kernel can then be booted with a command such as:

```
(hd0)/boot/vmlinuz-2.6.23-simtec1 init=/bin/sh root=/dev/sda1 console=ttySAC0,115200
```

6.3. Virtual disc

The system can be started with the **qemu** emulator. Generally no additional bootloader is required (although ABLE can be run within qemu if desired). The kernel to boot must however be provided this can either be retrived from the Simtec Electronics website or copied from the boot directory in the virtual disc.

A suitable command line to start the virtual system in single user mode would be:

```
qemu-system-arm -M simtecbast -kernel vmlinuz-2.6.23-simtec1 -append "init=/bin/sh root=/dev
```

7. Complete the bootstrap

From the shell prompt the root filesystem must be remounted read-write.

```
sh-3.1# mount -n -o rw,remount /
```

Next the process filesystem must be mounted.

```
sh-3.1# mount /proc
```

An appropriate path must be set for the next bootstrap stage.

```
sh-3.1# export PATH=/sbin:/usr/sbin:/bin:/usr/bin
```

The second stage bootstrap must be run, this process may take a long time and depends on both processor and filesystem performance.

```
sh-3.1# /debootstrap/debootstrap --second-stage
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Installing core packages...
I: Unpacking required packages...
I: Unpacking base-files...
I: Unpacking base-passwd...
I: Unpacking bash...
I: Unpacking bsdutils...
```

```
...

I: Configuring tasksel-data...
I: Configuring sysklogd...
I: Configuring tasksel...
I: Base system installed successfully.
```

If the system is to access a hard disc (real or virtual) the device nodes for sda must be created. This is achieved using the **MAKEDEV** command found in the dev directory.

```
sh-3.1# cd /dev
sh-3.1# ./MAKEDEV sda
```

If a login on the console serial port is required (which is recommended) some additional entries must be made to the systems inittab. The inittab is updated to give a login process on the console serial port. The serial port device must also be marked as a secure terminal so the superuser (root) can log in.

```
sh-3.1# echo "T0:2345:respawn:/sbin/getty -L ttySAC0 115200 vt100" >>/etc/inittab
sh-3.1# echo ttySAC0 >> /etc/securetty
```

The APT tool package sources which were derived from the host may not be appropriate. The `etc/apt/sources.list` file should be edited to contain suitable values.

Example 6. APT sources list for “stable” suite.

```
deb http://mirror/debian/ stable main
deb-src http://mirror/debian/ stable main

deb http://security.debian.org/ stable/updates main
deb-src http://security.debian.org/ stable/updates main
```

Once the base system is installed the system should be started by executing the init process.

```
sh-3.1# exec init
INIT: version 2.86 booting
Activating swap...done.
Setting the system clock..
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access method.
Cleaning up ifupdown....
Loading kernel modules...done.
Loading device-mapper support.
Checking file systems...fsck 1.40-WIP (14-Nov-2006)
done.
Setting kernel variables...done.
Mounting local filesystems...done.
Activating swapfile swap...done.
Setting up networking....
Configuring network interfaces...done.
INIT: Entering runlevel: 2
Starting system log daemon: syslogd.
Starting kernel log daemon: klogd.
* Not starting internet superserver: no services enabled.
Starting periodic command scheduler: crond.

Debian GNU/Linux 4.0 etch ttySAC0
```

```
etch login:
```

8. Final configuration

Log in as the superuser, there is no password set at this point.

```
Debian GNU/Linux 4.0 etch ttySAC0

etch login: root
Last login: Sun Dec 16 00:55:42 2007 on ttySAC0
Linux etch 2.6.23-simtec1 #1 Mon Nov 26 13:32:19 GMT 2007 armv4tl

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
etch:~#
```

The superuser password should immediately be set.

```
etch:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Next the APT package archive should be updated using the **apt-get update** command.

```
etch:~# apt-get update
Get:1 http://mirror stable Release.gpg [378B]
Get:2 http://mirror stable Release [58.2kB]
Get:3 http://security.debian.org stable/updates Release.gpg [189B]
Get:4 http://security.debian.org stable/updates Release [22.5kB]
Get:5 http://mirror stable/main Packages [5454kB]
Get:6 http://security.debian.org stable/updates/main Packages [225kB]
Get:7 http://security.debian.org stable/updates/main Sources [31.0kB]
Get:8 http://mirror stable/main Sources [1653kB]
Fetched 7444kB in 2m25s (51.2kB/s)
Reading package lists... Done
```

Generally the udev device management daemon should be installed, if the serial console option is being used the relevant line in the `inittab` file *must* be altered.

```
etch:~# apt-get install udev
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  libvolume-id0
The following NEW packages will be installed:
  libvolume-id0 udev
0 upgraded, 2 newly installed, 0 to remove and 10 not upgraded.
Need to get 329kB of archives.
After unpacking 1069kB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://mirror stable/main libvolume-id0 0.105-4 [65.3kB]
Get:2 http://mirror stable/main udev 0.105-4 [264kB]
Fetched 329kB in 2s (113kB/s)
Preconfiguring packages ...
```

```
Selecting previously deselected package libvolume-id0.
(Reading database ... 7086 files and directories currently installed.)
Unpacking libvolume-id0 (from .../libvolume-id0_0.105-4_arm.deb) ...
Selecting previously deselected package udev.
Unpacking udev (from .../archives/udev_0.105-4_arm.deb) ...
Setting up libvolume-id0 (0.105-4) ...

Setting up udev (0.105-4) ...
sed: can't read /etc/udev/rules.d/z25_persistent-net.rules: No such file or directory
Populating the new /dev filesystem temporarily mounted on /tmp/udev.hFUjs7/...
Restarting system log daemon: syslogd

etch:~# sed 's/\(T0.*\)ttySAC0\(.*\)/\1s3c2410_serial0\2/' </etc/inittab >/etc/inittab2
etch:~# mv /etc/inittab2 /etc/inittab
etch:~# echo s3c2410_serial0 >>/etc/securetty
```

If remote access to the system is desired the ssh daemon should be installed.

```
etch:~# apt-get install ssh
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  libedit2 libkrb53 openssh-client openssh-server
Suggested packages:
  krb5-doc krb5-user ssh-askpass xbase-clients rssh molly-guard
The following NEW packages will be installed:
  libedit2 libkrb53 openssh-client openssh-server ssh
0 upgraded, 5 newly installed, 0 to remove and 10 not upgraded.
Need to get 1269kB of archives.
After unpacking 3215kB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://mirror stable/main libedit2 2.9.cvs.20050518-2.2 [56.2kB]
Get:2 http://security.debian.org stable/updates/main libkrb53 1.4.4-7etch4 [390kB]
Get:3 http://mirror stable/main openssh-client 1:4.3p2-9 [605kB]
Get:4 http://mirror stable/main openssh-server 1:4.3p2-9 [217kB]
Get:5 http://mirror stable/main ssh 1:4.3p2-9 [1054B]
Fetched 1269kB in 16s (76.2kB/s)
Preconfiguring packages ...
Selecting previously deselected package libkrb53.
(Reading database ... 7201 files and directories currently installed.)
Unpacking libkrb53 (from .../libkrb53_1.4.4-7etch4_arm.deb) ...
Selecting previously deselected package libedit2.
Unpacking libedit2 (from .../libedit2_2.9.cvs.20050518-2.2_arm.deb) ...
Selecting previously deselected package openssh-client.
Unpacking openssh-client (from .../openssh-client_1%3a4.3p2-9_arm.deb) ...
Selecting previously deselected package openssh-server.
Unpacking openssh-server (from .../openssh-server_1%3a4.3p2-9_arm.deb) ...
Selecting previously deselected package ssh.
Unpacking ssh (from .../ssh_1%3a4.3p2-9_all.deb) ...
Setting up libkrb53 (1.4.4-7etch4) ...

Setting up libedit2 (2.9.cvs.20050518-2.2) ...

Setting up openssh-client (4.3p2-9) ...

Setting up openssh-server (4.3p2-9) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
NET: Registered protocol family 10
Restarting OpenBSD Secure Shell server: sshd.

Setting up ssh (4.3p2-9) ...
```

It is inadvisable to use the superuser for all activities on a system and a normal user should be added.

```
etch:~# adduser simtec
```

```
Adding user `simtec' ...
Adding new group `simtec' (1000) ...
Adding new user `simtec' (1000) with group `simtec' ...
Creating home directory `/home/simtec' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for simtec
Enter the new value, or press ENTER for the default
        Full Name []: Simtec User
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [y/N] y
```

9. Bootloader configuration

The system can now be configured to automatically boot, the configuration is dependant on the boot media.

9.1. NFS root

The most straightforward method for NFS based systems is to create an ABLE shell script on the TFTP server. This allows the boot parameters to be altered by editing the script on the server.

The script will contain a boot line similar to that constructed in the single user Section 6.1, “NFS root”.

Example 7. NFS root boot script

This script will typically be placed on the TFTP boot server alongside the kernel image. A suitable filename might be `nfs-etch.sh`, however this is largely up to the requirements of the specific configuration.

```
#!/bin/sh
(tftpboot)vmlinuz-2.6.23-simtec1 root=/dev/nfs nfsroot=192.168.1.1:/export/debian,rsize=1024
```

Once the script has been created ABLE can be configured to run it at startup by setting the *`boot.cmd`* non volatile variable to “(tftpboot)nfs-etch.sh” and then saving the variables with **nvsave**.

```
>boot.cmd="(tftpboot)nfs-etch.sh"
>nvsave
```

9.2. Disc drive

The bootloader configuration to boot from a disc drive is best achieved by setting the ABLE bootloader to run a script from the boot directory directly from the drive, this allows for the boot parameters to be changed from the running Debian system simply by editing the file.

The script will contain a boot line similar to that constructed in the single user Section 6.2, “Disc drive”.

Example 8. Disc drive boot script

This script should be placed alongside the kernel image in the boot directory and called `able.sh`. The bootloader ex-

ecutes the script directly and any commands to be run in ABLE can be placed here.

```
#!/bin/sh
(hd0)/boot/vmlinuz-2.6.23-simtec1 root=/dev/sda1 console=ttySAC0,115200
```

Once the script has been created ABLE can be configured to run it at startup by setting the *boot.cmd* non volatile variable to “(hd0)/boot/able.sh” and then saving the variables with **nvsave**.

```
>boot.cmd="(hd0)/boot/able.sh"
>nvsave
```

9.3. Virtual disc

Generally the emulated machine aproach removes the need for a seperate bootloader as the kernel image can be readily retrived by the host system.

A suitable command line to start the virtual system in multi user mode would be:

```
qemu-system-arm -M simtecbast -kernel vmlinuz-2.6.23-simtec1 -append "root=/dev/sda1 co
```

A. Support Scripts

A.1. mkdiscimage

This script creates a disc image, partitions it, creates a filesystem and performs a basic bootstrap on the filesystem.

The location of the Debian mirror to use should be edited before use.

The latest available version of this script should always be used.

The script can be downloaded [http://www.simtec.co.uk/products/SWDEBIAN/files/mkdiscimage] from the Simtec Electronics website.

```
#!/bin/bash
#
# mkdiscimage
#
# Create a disc image
#
# Usage mkdiscimage <image file> [<size>]
# size is in MB
#
# Version 1.0
#
# Copyright 2007 Simtec Electronics
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of this software and associated documentation
# files (the "Software"), to deal in the Software without
# restriction, including without limitation the rights to use, copy,
# modify, merge, publish, distribute, sublicense, and/or sell copies
# of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

```
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
# MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
# BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
# ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#

DEFAULTSIZE=512
SWAPSIZE=128
DEBIAN_MIRROR="http://mirror/debian"
DEBIAN_SUITE=etch
DEBIAN_ARCH=arm
KERNEL_SRC=http://www.simtec.co.uk/products/SWLINUX/files
KERNEL_IMG=s3c24xx-linux-2.6.23-simtec1.tar.bz2


###################### end of parameters ######################

bootstrap_fs()
{
    MOUNT=$1
    ROOT_UUID=$2

    if [ "x${MOUNT}" = "x" ]; then
        return 1
    fi

    if [ "x${ROOT_UUID}" = "x" ]; then
        return 2
    fi

    OLDDIR=$(pwd)

    cd ${MOUNT}

    debootstrap --arch ${DEBIAN_ARCH} --foreign ${DEBIAN_SUITE} ${MOUNT} ${DEBIAN_MIRROR}

    wget ${KERNEL_SRC}/${KERNEL_IMG}
    tar -jxf ${KERNEL_IMG}
    rm ${KERNEL_IMG}

    echo "${DEBIAN_ARCH}" > ${MOUNT}/etc/hostname

    mknod ${MOUNT}/dev/console c 5 1
    mknod ${MOUNT}/dev/ttySAC0 c 204 64
    mknod ${MOUNT}/dev/ttySAC1 c 204 65
    mknod ${MOUNT}/dev/ttySAC2 c 204 66

    echo "# <file system> <mount point>   <type>  <options>       <dump>  <pass>" > ${MOUNT}
    echo "proc /proc proc defaults 0 0" >> ${MOUNT}/etc/fstab

    echo "UUID=${ROOT_UUID} / ext3 defaults,errors=remount-ro 0" >> ${MOUNT}/etc/fstab

    cd ${OLDDIR}

}

bootstrap_e2fs ()
{
    DMDEVICE=$1

    if [ "x${DMDEVICE}" = "x" ]; then
        return 1
    fi

    # the devicemapper setup worked
    mke2fs -O dir_index,has_journal,sparse_super ${DMDEVICE}

    UUID=$(tune2fs -l ${DMDEVICE} | sed -n 's/^Filesystem UUID:[ \t]*\(.*\)/\1/p')
```

```
    mount ${DMDEVICE} /mnt

    if [ $? -eq 0 ];then
        bootstrap_fs /mnt $UUID
    fi

    umount ${DMDEVICE}
}

####################### main code ######################

FORM=virtdisc

# command line parameters
D=$(dirname $0)

# required tools
TOOLS="${D}/lspartab dd losetup parted dmsetup"

DISCIMAGE=$1
if [ "x${DISCIMAGE}" = "x" ]; then
    echo "usage:"
    echo "mkvirtdisc <image-filename> [<size>]"
    exit 1
fi

# ensure superuser privileges are available
if [ "x${UID}" = "x" -o "${UID}" -ne 0 ]; then
    which sudo >/dev/null
    if [ $? -ne 0 ]; then
        echo "Script requires superuser privileges!"
        exit 1
    else
        sudo $0 $*
        exit $?
    fi
fi

DISCSIZE=$2
if [ "x${DISCSIZE}" = "x" ]; then
    echo "Using default size of ${DEFAULTSIZE}"
    DISCSIZE=${DEFAULTSIZE}
fi


# ensure the required commands are present
which ${TOOLS} >/dev/null
if [ $? -ne 0 ]; then
    echo "Required command is missing"
    echo "${TOOLS}"
    exit 1
fi

# calculated values
ROOTSIZE=$(expr ${DISCSIZE} - ${SWAPSIZE})
READTAB="${D}/lspartab"

# check we can setup a loopback device
LOOP=$(losetup -f)

if [ "x${LOOP}" = "x" ]; then
    echo "Cannot find a loop"
    exit 1
fi

# create image file
dd if=/dev/zero of=${DISCIMAGE} bs=1M count=${DISCSIZE}

# partition it
parted --script ${DISCIMAGE} mklabel msdos mkpart primary ext2 0 ${ROOTSIZE} mkpartfs p
```

```
# create device nodes
if ! losetup "${LOOP}" "${DISCIMAGE}"; then
    losetup -d "${LOOP}"
    echo "Unable to losetup!"
    exit 1
fi

LOOPNODE=$(stat -c '%t:%T' "${LOOP}")

"${READTAB}" "${DISCIMAGE}" | while read partnum start length type; do
    echo "Partition $FORM$partnum starts at $start and is $length sectors"
    echo "0 $length linear $LOOPNODE $start" | dmsetup create ${FORM}${partnum}
    if [ -b /dev/mapper/${FORM}${partnum} ]; then
        #device mapper setup was succesful

        case "${type}" in
            ( "131" ) bootstrap_e2fs /dev/mapper/${FORM}${partnum} ;;
        esac


        dmsetup remove ${FORM}${partnum}
    fi
done


if ! losetup -d "${LOOP}"; then
    echo "losetup -d failed?"
    exit 1
fi

exit 0
```

A.2. lspartab

This script lists the partition table of an MSDOS labeled disc. The output format is partition number, start, length, type. The start and length are given in disc blocks suitable for use with the Linux device mapper.

The latest available version of this script should always be used.

Updates to the script may be downloaded [http://www.simtec.co.uk/products/SWDEBIAN/files/lspartab] from the Simtec Electronics website.

```
#!/usr/bin/perl
#
# lspartab
#
# Read, parse and list an MSDOS labeled disc partition table
#
# Copyright 2007 Simtec Electronics
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of this software and associated documentation
# files (the "Software"), to deal in the Software without
# restriction, including without limitation the rights to use, copy,
# modify, merge, publish, distribute, sublicense, and/or sell copies
# of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
# MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
# BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
# ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
```

```
# SOFTWARE.

$F = $ARGV[0] or die "Usage: lspartab <disc image>";

open PARTTAB, "<", $F;
binmode PARTTAB;

my @partstarts = (446,462,478,494);

$extended = 4;

sub scanparts {
   my $secoffset = shift;
   my $pnum = shift;
   my $isext = shift;
   $buffer = "                ";
   foreach $partstart (@partstarts) {
      seek(PARTTAB, (512 * $secoffset) + $partstart, 0);
      sysread PARTTAB, $buffer, 16;
      $pnum++;
      ($type, $b1, $b2, $b3, $sys, $e1, $e2, $e3, $start, $count) = unpack("CCCCCCCCLL"
      if ($sys == 5 || $sys == 15 || $sys == 0x85) {
         if ($secoffset == 0) {
            scanparts($isext+$start, $extended, $secoffset+$start);
         } else {
            scanparts($isext+$start, $extended, $isext);
         }
      }
      $extended++ if $isext;
      if( $count == 0 ) { $pnum++; next; }
      print $pnum," ", $secoffset+$start, " ", $count, " ", $sys, "\n";
   }
   return $pnum;
}

scanparts(0,0,0);

close PARTTAB;
```