Introduction

This howto will show you how to boot your Mini2440 over the network using NFS. I'm using Ubuntu 9.04 as usual.

I ran in to no problems at all getting this to work. This has been one of the more simpler things I've done with the mini. The NFS Server described here has no security and is wide open. You should make sure this runs on your internal network only and that the shares are not a security risk to your own system and that you only export to IPs that need it.

Installing the NFS Server

On your host machine, you'll need to install the NFS Server. Use the following command in Ununtu:

Code:

```
$ apt-get install nfs-kernel-server portmap nfs-common
```

Next, you will need to create and export your NFS directories so that U-Boot and Linux can find them. I always export my kernel separate from my filesystem and there's really no real good reason for it, I just like doing it that way.

Create the directories like so:

Code:

```
$ mkdir /export/
$ mkdir /export/fs
$ mkdir /export/kernel
```

Now you'll need to export those by editing the file /etc/exports:

Code:

```
/export
192.168.1.0/24(rw,fsid=0,insecure,no_subtree_check,async)
/export/fs
```

```
192.168.1.0/24(rw,fsid=0,insecure,no_subtree_check,async,no
_root_squash)
/export/kernel
192.168.1.0/24(rw,fsid=0,insecure,no_subtree_check,async)
```

The above means that you're exporting to anyone on 192.168.1 subnet. You must add the no_root_squash to the filesystem export otherwise you'll get errors during boot.

Now restart your NFS Server:

Code:

```
$ /etc/init.d/nfs-kernel-server restart
```

Setting U-Boot Environment Variables

First you'll need to set the bootcmd variable so that it loads the kernel in to RAM from the remote host and then executes it. Replace x with your NFS server IP.

Code:

```
MINI2440# setenv bootcmd nfs 0x31000000
192.168.1.x:/export/kernel/uImage \; bootm
```

Second, you'll need to set the bootargs to tell the linux kernel where to find your rootfs. Replace x with your nfs server IP and y with the IP you want to be assigned to the MINI2440.

Code:

```
MINI2440# setenv bootargs console=ttySAC0,115200
root=/dev/nfs nfsroot=192.168.1.x:/export/fs rw
ip=192.168.1.y mini2440=1tb
```

Finally, save your variables.

Code:

```
MINI2440# saveenv
```

If you want to obtain an IP via dhcp, you can set ip=dhcp.

Copying your filesystem

Now that you have the NFS Server set up, and your MINI2440 is configured to boot, you need to copy your filesystem and kernel over to the export directories. And that is pretty simple. If you're using OpenEmbedded, as you should be, just tell OE in the local.conf to create tar.gz files of the root filesystem. (See the local.conf for information), and then when you build the filesystem, it'll create a nice tarball for you that you can untar in to the fs directory in your exports. OE will even create a nice uImage you can put in your kernel directory.

Final Thoughts

There are probably better ways to do the Mini2440 variables. And you can read all about that in the U-Boot manual, but my way works and that's all that matters to me. But feel free to send better ideas to me about how to do any of this.

I personally don't think NFS is very fast, but it saves a lot of wear and tear on the SD card and on the NAND. It's a great way to debug a filesystem and its a lot nicer if you're trying out new filesystems all the time PLUS you have write and read access to that filesystem while the other machine is using it.

As usual, I hope this has helped you in some way. Please share with me your stories of failure or success and if you want to contribute tips and advice, please feel free to do so.

Thanks,
Bill

UPdate: If your kernel doesn't have CONFIG_NFS_PNP enabled, you'll get an error during boot where it's looking for the port of the NFS server. If thats so, you can use these parameters on the

kernel command line to manually specify your IP and netmask etc...

Code:

```
nfsaddrs=<client-ip>:<server-ip>:<gw-
ip>:<netmask>:<hostname>:<device>:<autoconf>

  If this parameter is missing on the kernel command line,
all fields are
  assumed to be empty, and the below mentioned defaults
apply. In general
  this means that the kernel tries to configure everything
using both
  RARP and BOOTP (depending on what has been enabled during
kernel confi-
  guration, and if both what protocol answer got in first).

  <client-ip>   IP address of the client. If empty, the
address will either
      be determined by RARP or BOOTP. What protocol is used
de-
      pends on what has been enabled during kernel
configuration
      and on the <autoconf> parameter. If this parameter is
not
      empty, neither RARP nor BOOTP will be used.

  <server-ip>   IP address of the NFS server. If RARP is
used to determine
      the client address and this parameter is NOT empty
only
      replies from the specified server are accepted. To
use
      different RARP and NFS server, specify your RARP
server
      here (or leave it blank), and specify your NFS server
in
      the nfsroot parameter (see above). If this entry is
blank
      the address of the server is used which answered the
RARP
      or BOOTP request.

  <gw-ip>   IP address of a gateway if the server in on a
```

```
different
      subnet. If this entry is empty no gateway is used and
the
      server is assumed to be on the local network, unless
a
      value has been received by BOOTP.

  <netmask>   Netmask for local network interface. If this
is empty,
      the netmask is derived from the client IP address,
un-
      less a value has been received by BOOTP.

  <hostname>   Name of the client. If empty, the client IP
address is
      used in ASCII-notation, or the value received by
BOOTP.

  <device>   Name of network device to use. If this is
empty, all
      devices are used for RARP requests, and the first one
      found for BOOTP. For NFS the device is used on which
      either RARP or BOOTP replies have been received. If
      you only have one device you can safely leave this
blank.

  <autoconf>   Method to use for autoconfiguration. If this
is either
      'rarp' or 'bootp' the specified protocol is being
used.
      If the value is 'both' or empty, both protocols are
used
      so far as they have been enabled during kernel
configura-
      tion. 'none' means no autoconfiguration. In this case
you
      have to specify all necessary values in the fields
before.

  The <autoconf> parameter can appear alone as the value to
the nfsaddrs
  parameter (without all the ':' characters before) in
which case auto-
```

```
  configuration is used. However, the 'none' value is not
available in
  that case.
```