Kernel loading utilizing TFTP protocol and NFS

Author:

Armando Emmanuel Correa Amorelli

Zapopan, Jalisco, June 10, 2023

Table of Contents

Introduction

An alternative way to boot a development board is through a network interface. This will be useful during the development stage of a project as it allows for the flexibility to develop applications or modifications to a Linux kernel more quickly and efficiently.

Objective

Configure a NFS and TFTP server to load a kernel to the beagle bone black

TFTP & NFS

Trivial File Transfer Protocol (TFTP)

Usage:

TFTP is a simple file transfer protocol that allows for transferring files between a client and a server. It is commonly used in scenarios where simplicity and minimal overhead are more important than advanced features.

Functionality:

TFTP operates on top of UDP (User Datagram Protocol) and uses port number 69. It supports basic read and write operations but does not provide authentication or encryption. TFTP is often used in network booting scenarios, where a client downloads a boot file from a TFTP server to initiate the boot process. It is also used for firmware upgrades, configuration file transfers, and other scenarios where small file transfers are required.

Network File System (NFS)

Usage:

NFS is a distributed file system protocol that allows clients to access files and directories on remote servers as if they were local. It provides a transparent and efficient way to share files across a network.

Functionality:

NFS operates over IP networks and allows clients to mount remote file systems over a network. It supports features such as file and directory access, file locking, and file attribute retrieval. NFS uses RPC (Remote Procedure Call) for communication between the client and the server. It provides file sharing among heterogeneous systems, allowing clients to access files on servers running different operating systems.

Setting up the servers

TFTP

We need to install the `tftpd-hpa` package from the apt repositories into our dev machine. After the installation, we need to check the content of the file located at `/etc/default/tftpd-hpa` this is a default configuration file that specify the username, address, options and directory from where the client (BBB) would get the files.

Once we check this, we need to make sure that the directory that points to really exists. Lastly, we need to restart the service in case

NFS

We need to install the `nfs-kernel-server` package from the apt repositories into our dev machine. Then edit `/etc/exports` (as root) and add the following line

```
/home/armando/diplomado/modulo3/nfsroot
192.168.0.100(rw,no_root_squash,no_subtree_check)
```

Lastly you need to restart the NFS server

Setting up the BBB

Utilizing the U-Boot console, we're going to set up the IP address for the server, and the board. Also, we need to specify their MAC addresses and the network interface to be used.

For doing all this we need to save these values to each of the following variables utilizing the setenv command, and lastly the saveenv to make it persistent.

- **ipaddr**
- **serverip**
- **ethprime**
- **usbnet_devaddr**
- **usbnet_hostaddr**

All this is to prepare and be able to utilize the TFTP protocol, for NFS, we need to set the bootargs as shown bellow:
setenv bootargs root=/dev/nfs rw ip=192.168.0.100:::::usb0 console=ttyS0,115200n8

g_ether.dev_addr=f8:dc:7a:00:00:02 g_ether.host_addr=f8:dc:7a:00:00:01

nfsroot=192.168.0.1:/home/armando/diplomado/modulo3/nfsroot,nfsvers=3,tcp

Kernel Loading and booting

To load the kernel, we need first place the needed files at the directory pointed by our TFTP server. Then we run the TFTP command within the U-Boot terminal.
For this we used the following commands

```
tftp 0x81000000 zimage

tftp 0x82000000 am335x-boneblack.dtb
```

To make sure that this worked we can now run the bootz comand to boot with the kernel that we download to the BBB

```
bootz  0x81000000 – 0x82000000
```

Once it boot we should also be able to look through the file system that we shared with NFS.

To make all this automatic and persistent, we can save it as environment command and call it at boot doing the following:

```
setenv tftp_custom “tftp 0x81000000 zimage; tftp 0x82000000 am335x-boneblack.dtb; bootz 0x81000000 - 0x82000000;”

setenv bootcmd “run tftp_custom; run findfdt; run

init_console; run finduuid; run distro_bootcmd”

saveenv
```

Conclusions

TFTP and NFS protocols are something handy because they allow us to make real-time changes to files that can be used by multiple clients connected to our network. Also, we can keep the kernel update more manageable by using the TFTP and loading it at each boot instead of having it persistent on our board.

Bibliography

[1]“The U-Boot Documentation — Das U-Boot unknown version documentation,” *u-boot.readthedocs.io*. https://u-boot.readthedocs.io/en/latest/ (accessed Jun 10, 2023).

[2] B. Mitchell, “Definition of trivial file transfer protocol (TFTP),” Lifewire, https://www.lifewire.com/definition-of-tftp-817576 (accessed Jun. 10, 2023).

[3] JasonGerend, “Network File System Overview,” Microsoft Learn, https://learn.microsoft.com/en-us/windows-server/storage/nfs/nfs-overview (accessed Jun. 10, 2023).

[4] T. Haynes and D. Noveck, “Network File System (NFS) version 4 protocol,” RFC Editor, https://www.rfc-editor.org/rfc/rfc7530.html (accessed Jun. 10, 2023).

[5] k Sollins, “THE TFTP PROTOCOL (REVISION 2),” RFC1350, https://www.rfc-editor.org/rfc/inline-errata/rfc1350.html (accessed Jun. 10, 2023).