jamie.lentin.co.uk

D-Link DNS-320 & DNS-325 NAS: Keeping D-Link Firmware

It's not necessary to remove the original firmware to boot Linux, so long as you have serial access you can tell u-boot to load a different kernel temporarily.

Debian kernels will support the DNS-320 and DNS-325 from jessie onwards.

Building a root filesystem ↑

debootstrap is a utility that allows you to create a Debian installation from another already-installed system. My good friend has a guide on building one using debootstrap, so will only go into brief detail here. All of his tips are useful, so read that first!

Start debootstrap with packages you want to install:

```
host:# aptitude install debootstrap binfmt-support qemu-
host:# echo -n "linux-image-kirkwood,u-boot-tools" > inc
host:# echo -n ",netbase,ifupdown" >> includepackages
host:# echo -n ",localepurge,dialog,procps,busybox-syslc
host:# echo -n ",ne,nano,dropbear" >> includepackages
host:# debootstrap --verbose --foreign --arch=armel --va
       --include=$(cat includepackages) \
       jessie rootfs http://ftp.uk.debian.org/debian
```

Use qemu-arm-static to chroot and finish the job:

```
host:# cp /usr/bin/qemu-arm-static rootfs/usr/bin/ && ch
        && rm rootfs/usr/bin/qemu-arm-static
I have no name!@host:/# ./debootstrap/debootstrap --seco
```

Make busybox do everything that doesn't have a full-fat equivalent:

```
for f in $(busybox --list-full | grep -v readlink); do
```

Do some minimal configuration:

```
I have no name!@host:/# cat <<EOF > etc/fstab
/dev/root    /       auto    noatime                 0 0
tmpfs        /tmp    tmpfs   nodev,nosuid,size=32M   0 0
```

contact me

```
EOF
I have no name!@host:/# echo 'nas' > etc/hostname
I have no name!@host:/# passwd
I have no name!@host:/# cat <<EOF > etc/network/interfac
# Edit to match your network
auto eth0
iface eth0 inet static
    address 192.168.1.9
    netmask 255.255.255.0
    gateway 192.168.1.1
EOF
I have no name!@host:/# echo "nameserver 192.168.1.1" >
```

Next, set up APT sources:

```
I have no name!@host:/# echo 'APT { Install-Recommends '
I have no name!@host:/# cat <<EOF > etc/apt/sources.list
deb http://ftp.uk.debian.org/debian/ jessie main contrib
deb http://security.debian.org/ jessie/updates main cont
EOF
```

Configure the locales you want and purge the rest:

```
I have no name!@host:/# dpkg-reconfigure locales localep
# Choose "no" to working via. dpkg for now
I have no name!@host:/# localepurge -v
```

Build u-boot images whenever a new kernel is installed:

```
# All these modules will be stored in the initramfs and
# You may want some filesystems too, e.g. ext4
I have no name!@host:/# cat <<'EOF' >> etc/initramfs-too
# Thermal management
gpio-fan
kirkwood_thermal
# SATA
ehci_orion
sata_mv
# Ethernet
mv643xx_eth
marvell
mvmdio
ipv6
# Power / USB buttons
evdev
gpio_keys
# USB disks
sd_mod
usb_storage
EOF
I have no name!@host:/# cat <<'EOF' > /etc/kernel/postin
#!/bin/sh -e
# passing the kernel version is required
version="$1"
[ -z "${version}" ] && exit 0

# NB: change depending on your NAS model
cat /boot/vmlinuz-${version} /usr/lib/linux-image-${vers
    > /tmp/appended_dtb
```

```
/usr/bin/mkimage -A arm -O linux -T kernel -C none -n u
                 -a 0x00008000 -e 0x00008000 \
                 -d /tmp/appended_dtb /boot/uImage-${ve
ln -sf /boot/uImage-${version} /boot/uImage

/usr/bin/mkimage -A arm -O linux -T ramdisk -C gzip -n u
                 -a 0x00e00000 -e 0x00e00000 \
                 -d /boot/initrd.img-${version} /boot/u
ln -sf /boot/uInitrd-${version} /boot/uInitrd
EOF
I have no name!@host:/# chmod a+x /etc/kernel/postinst.d
# Recreate the kernel package to create the images to bo
I have no name!@host:/# dpkg-reconfigure $(dpkg --get-se
```

Tidy up and exit the chroot:

```
I have no name!@host:/# apt-get clean
I have no name!@host:/# rm -- tmp/* var/tmp/* var/lib/ap
I have no name!@host:/# exit
```

At this stage, you should have a directory full of all the normal files that make up a Debian system, and kernel / initrd images in /boot. Next you will need to get it to the NAS and boot from it.

Booting the NAS ↑

First, connect to the serial port and turn on the NAS. At the `Hit any key to stop autoboot` prompt press space then 1. You should then be sitting at a "Marvell>>" u-boot prompt. Before doing anything, run:

```
Marvel>> printenv
```

And make a note of the contents. This will be a useful reference if you want to restore any u-boot parameters.

Booting Via USB

U-boot (the bootloader) supports ext2-formatted USB keys. However it is **very** picky about what you use. Use a proper USB key, not a USB card reader. If u-boot seems unable to find the key, try another.

Format the stick with an ext2 filesystem. Then copy the root filesystem you generated onto it:

```
host:~# cp -arv rootfs/* /mnt/usb_stick/
```

Of course you can also just have a small `/boot` partition that is

ext2 formatted and the rest something else, but I will leave that as an exercise for the reader.

Now go back to your serial console where your NAS is waiting. Insert the USB stick and run the following commands:

```
Marvell>> setenv ethaddr 00:50:43:xx:xx:xx [Any MAC addı
Marvell>> setenv bootargs console=ttyS0,115200 root=/dev
Marvell>> usb reset ; ext2load usb 0:1 0xa00000 /boot/uI
Marvell>> bootm 0xa00000 0xf00000
```

This does the following...

1. Set the MAC address. D-link ships the NASes with corrupt u-boot configuration, so it assigns a random MAC address on each reboot until you do `saveenv`. This will confuse Debian. If you're not sure what to choose, make a note of what it uses when booted with the D-link formware and use that.
2. Set kernel command line options, namely a serial console and to use the USB stick as root.
3. Initialise USB, load image off first USB stick into memory.
4. Run loaded image. **NB:** D-link's u-boot will refuse to load the initrd unless it is copied to this memory address.

After this, you should watch your system boot.

Booting Via Network

You can load kernels over the network using TFTP, and use NFS to get the root filesystem. You will need to set up servers for both, see instructions here.

Copy the uImage built earlier into the TFTP server's directory, for example:

```
host:linux-2.6$ cp rootfs/boot/u{Image,Initrd}* /srv/tft
```

And copy the root filesystem into a NFS export, for example:

```
host:~# cp -arv rootfs/* /srv/export/nas_root/
```

Now go back to your serial console where your NAS is waiting. Do something like:

```
Marvell>> setenv ethaddr 00:50:43:xx:xx:xx [Any MAC addı
Marvell>> setenv ipaddr [Our IP address, e.g. 10.150.1.1
Marvell>> setenv serverip [Where TFTP server is, e.g. 10
```

```
Marvell>> setenv bootargs console=ttyS0,115200 root=/dev
  nfsroot=[NFS server, e.g. 10.150.1.10]:/srv/export/nas
Marvell>> tftp 0xa00000 uImage-3.10-0.bpo.2-kirkwood
Marvell>> tftp 0xf00000 uInitrd-3.10-0.bpo.2-kirkwood
Marvell>> bootm 0xa00000 0xf00000
```

This does the following...

1. Set the MAC address.
2. Configure the network so the TFTP server can be found.
3. Set kernel command line options, namely a serial console and where to find the NFS server
4. Copy kernel image into memory via TFTP.
5. Run loaded image. **NB:** D-link's u-boot will refuse to load the initrd unless it is copied to this memory address.

After this, you should watch your system boot.

Boot Debian by default ↑

Your NAS should now be booting. You can make the above happen by default by:

```
Marvell>> [all the setenv commands from above]
Marvell>> setenv bootcmd '[commands needed to boot]'
Marvell>> saveenv
Marvell>> reset
```

And return to D-Link firmware either by running the contents of the original bootcmd manually, or setenv'ing the parameters back again.

Installing kernel to NAND ↑

Whilst we're trying to leave the NAND alone on this page, it may make your life easier to write the kernel image here to save you keeping a TFTP server running, e.g.

NAND chips can be divided into partitions. D-Link by default divide it into 6:

```
Creating 6 MTD partitions on "orion_nand":
0x000000000000-0x000000100000 : "u-boot"
0x000000100000-0x000000600000 : "uImage"
0x000000600000-0x000000b00000 : "ramdisk"
0x000000b00000-0x000007100000 : "image"
0x000007100000-0x000007b00000 : "mini firmware"
0x000007b00000-0x000008000000 : "config"
```

The first is for u-boot, the second for the kernel image. We want to write the uImage built earlier here:

```
nas:~# dd if=/dev/mtd1 of=dlink.uImage bs=1M
nas:~# flash_eraseall /dev/mtd1
Erasing 128 Kibyte @ 500000 -- 100 % complete.
nas:~# nandwrite -p /dev/mtd1 ${path-to-uImage}
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
Writing data to block 2 at offset 0x40000
Writing data to block 3 at offset 0x60000
Writing data to block 4 at offset 0x80000
Writing data to block 5 at offset 0xa0000
Writing data to block 6 at offset 0xc0000
Writing data to block 7 at offset 0xe0000
Writing data to block 8 at offset 0x100000
Writing data to block 9 at offset 0x120000
Writing data to block 10 at offset 0x140000
Writing data to block 11 at offset 0x160000
Writing data to block 12 at offset 0x180000
Writing data to block 13 at offset 0x1a0000
Writing data to block 14 at offset 0x1c0000
```

Finally, set up u-boot environment to permanently boot our kernel:

```
Marvell>> [all the setenv commands from above]
Marvell>> setenv bootcmd 'nand read.e 0x1000000 0x00100(
Marvell>> saveenv
Marvell>> reset
```

Next Steps ↑

- Replace everything, including u-boot
- Userland configuration