NAME

diskless - booting a system over the network

DESCRIPTION

The ability to boot a machine over the network is useful for *diskless* or *dataless* machines, or as a temporary measure while repairing or re-installing file systems on a local disk. This file provides a general description of the interactions between a client and its server when a client is booting over the network.

OPERATION

When booting a system over the network, there are three phases of interaction between client and server:

1. The stage-1 bootstrap, typically PXE built into your Ethernet card, loads a second-stage boot program.

2. The second-stage boot program, typically pxeboot(8), loads modules and the kernel, and boots the kernel.

3. The kernel NFS mounts the root directory and continues from there.

Each of these phases are described in further detail below.

First, the stage-1 bootstrap loads the stage-2 boot program over the network. The stage-1 bootstrap typically uses BOOTP or DHCP to obtain the filename to load, then uses TFTP to load the file. This file is typically called *pxeboot*, and should be copied from */boot/pxeboot* into the TFTP directory on the server, which is typically */tftpdir*.

The stage-2 boot program then loads additional modules and the kernel. These files may not exist on the DHCP or BOOTP server. You can use the **next-server** option available in DHCP configurations to specify the server holding the second stage boot files and kernel. The stage-2 program uses NFS or TFTP to obtain these files. By default, NFS is used. If you are using pxeboot(8), you can install a version that uses TFTP by setting LOADER_TFTP_SUPPORT=YES in your make.conf(5), then recompiling and reinstalling pxeboot(8) via the command listed below. It is often necessary to use TFTP here so you can place a custom kernel in */tftpdir/*. If you use NFS and do not have a custom root file system for the **diskless** client, the stage-2 boot will load your server's kernel as the kernel for the **diskless** machine, which may not be what you want to have happen.

```
cd /usr/src/stand
make clean; make; make install
cp /boot/pxeboot /tftpdir/
```

In phase 3, the kernel acquires IP networking configuration in one of two ways, and then proceeds to mount the root file system and start operation. If the phase 2 loader supports passing network configuration to the kernel using the kernel environment, then the kernel will configure the network interface using that information. Otherwise, it must use DHCP or BOOTP to acquire configuration information. The boot scripts recognize a **diskless** startup and perform the actions found in */etc/rc.d/resolv*, */etc/rc.d/tmp*, */etc/rc.d/var*, and */etc/rc.initdiskless*.

CONFIGURATION

In order to run a **diskless** client, you need the following:

- An NFS server which exports a root and */usr* partitions with appropriate permissions. The **diskless** scripts work with read-only partitions, as long as root is exported with **-maproot**=0 so that some system files can be accessed. As an example, */etc/exports* can contain the following lines:

  ```
  <ROOT> -ro -maproot=0 -alldirs <list of diskless clients>
  /usr -ro -alldirs <list of diskless clients>
  ```

  where <ROOT> is the mount point on the server of the root partition. The script */usr/share/examples/diskless/clone_root* can be used to create a shared read-only root partition, but in many cases you may decide to export (again as read-only) the root directory used by the server itself.

- A BOOTP or DHCP server. bootpd(8) can be enabled by uncommenting the "bootps" line in */etc/inetd.conf*. A sample */etc/bootptab* can be the following:

  ```
   .default:\
     hn:ht=1:vm=rfc1048:\
     :sm=255.255.255.0:\
     :sa=<SERVER>:\
     :gw=<GATEWAY>:\
     :rp="<SERVER>:<ROOT>":

  <CLIENT>:ha=0123456789ab:tc=.default
  ```

  where <SERVER>, <GATEWAY> and <ROOT> have the obvious meanings.

- A properly initialized root partition. The script */usr/share/examples/diskless/clone_root* can help in creating it, using the server’s root partition as a reference. If you are just starting out, you should simply use the server’s own root directory, */*, and not try to clone it.

  You often do not want to use the same *rc.conf* or *rc.local* files for the **diskless** boot as you do on the server. The **diskless** boot scripts provide a mechanism through which you can override various files in */etc* (as well as other subdirectories of root).

One difference that you should pay particular attention to is the value of *local_startup* in */etc/defaults/rc.conf*. A typical value for a **diskless** boot is *mountcritremote*, however your needs may be different.

The scripts provide four overriding directories situated in */conf/base*, */conf/default*, */conf/<broadcast-ip>*, and */conf/<machine-ip>*. You should always create */conf/base/etc*, which will entirely replace the server's */etc* on the **diskless** machine. You can clone the server's */etc* here or you can create a special file which tells the **diskless** boot scripts to remount the server's */etc* onto */conf/base/etc*. You do this by creating the file */conf/base/etc/diskless_remount* containing the mount point to use as a basis of the **diskless** machine's */etc*. For example, the file might contain:

```
10.0.0.1:/etc
```

Alternatively, if the server contains several independent roots, the file might contain:

```
10.0.0.1:/usr/diskless/4.7-RELEASE/etc
```

This would work, but if you copied */usr/diskless/4.7-RELEASE* to */usr/diskless/4.8-RELEASE* and upgraded the installation, you would need to modify the *diskless_remount* files to reflect that move. To avoid that, paths in *diskless_remount* files beginning with */* have the actual path of the client's root prepended to them so the file could instead contain:

```
/etc
```

The **diskless** scripts create memory file systems to hold the overridden directories. Only a 5MB partition is created by default, which may not be sufficient for your purposes. To override this, you can create the file */conf/base/etc/md_size* containing the size, in 512 byte sectors, of the memory disk to create for that directory.

You then typically provide file-by-file overrides in the */conf/default/etc* directory. At a minimum, you must provide overrides for */etc/fstab*, */etc/rc.conf*, and */etc/rc.local* via */conf/default/etc/fstab*, */conf/default/etc/rc.conf*, and */conf/default/etc/rc.local*.

Overrides are hierarchical. You can supply network-specific defaults in the */conf/<BROADCASTIP>/etc* directory, where *<BROADCASTIP>* represents the broadcast IP address of the **diskless** system as given to it via BOOTP. The *diskless_remount* and *md_size* features work in any of these directories. The configuration feature works on directories other then

/etc, you simply create the directory you wish to replace or override in */conf/{base,default,<broadcast>,<ip>}/** and work it in the same way that you work */etc*.

Since you normally clone the server's */etc* using the */conf/base/etc/diskless_remount*, you might wish to remove unneeded files from the memory file system. For example, if the server has a firewall but you do not, you might wish to remove */etc/ipfw.conf*. You can do this by creating a */conf/base/<DIRECTORY>.remove* file. For example, */conf/base/etc.remove*, which contains a list of relative paths that the boot scripts should remove from the memory file systems.

As a minimum, you normally need to have the following in */conf/default/etc/fstab*

```
<SERVER>:<ROOT> /     nfs    ro 0 0
<SERVER>:/usr   /usr  nfs    ro 0 0
```

You also need to create a customized version of */conf/default/etc/rc.conf* which should contain the startup options for the **diskless** client, and */conf/default/etc/rc.local* which could be empty but prevents the server's own */etc/rc.local* from leaking onto the **diskless** system.

In *rc.conf*, most likely you will not need to set *hostname* and *ifconfig_** because these will be already set by the startup code. Finally, it might be convenient to use a **case** statement using ‘hostname‘ as the switch variable to do machine-specific configuration in case a number of **diskless** clients share the same configuration files.

- The kernel for the **diskless** clients, which will be loaded using NFS or TFTP, must include support for the NFS client:

  ```
  options NFSCL
  options NFS_ROOT
  ```

  If you are using a boot mechanism that does not pass network configuration to the kernel using the kernel environment, you will also need to include the following options:

  ```
  options BOOTP
  options BOOTP_NFSROOT
  options BOOTP_COMPAT
  ```

  Note: the PXE environment does not require these options.

  The **diskless** booting environment relies on memory-backed file systems to support temporary local storage in the event that the root file system is mounted read-only; as such, it is necessary to add the following to the device section of the kernel configuration:

device md

If you use the firewall, remember to default to "open", or your kernel will not be able to send/receive the BOOTP packets.

SECURITY ISSUES

Be warned that using unencrypted NFS to mount root and user partitions may expose information such as encryption keys.

SEE ALSO

ethers(5), exports(5), make.conf(5), bootpd(8), mountd(8), nfsd(8), pxeboot(8), reboot(8), tftpd(8)

ports/net/etherboot

HISTORY

The **diskless** environment first appeared in FreeBSD 2.2.5.

BUGS

This manpage is probably incomplete.

FreeBSD sometimes requires to write onto the root partition, so the startup scripts mount MFS file systems on some locations (e.g. */etc* and */var*), while trying to preserve the original content. The process might not handle all cases.