Mounting the root filesystem via NFS (nfsroot)

Authors: Written 1996 by Gero Kuhlmann <gero@gkminix.han.de>
Updated 1997 by Martin Mares <mj@atrey.karlin.mff.cuni.cz>
Updated 2006 by Nico Schottelius <nico-kernel-nfsroot@schottelius.org>
Updated 2006 by Horms <horms@verge.net.au>
Updated 2018 by Chris Novakovic <chris@chrisn.me.uk>

In order to use a diskless system, such as an X-terminal or printer server for example, it is necessary for the root filesystem to be present on a non-disk device. This may be an initramfs (see Documentation/filesystems/ramfs-rootfs-initramfs.rst), a ramdisk (see Documentation/admin-guide/initrd.rst) or a filesystem mounted via NFS. The following text describes on how to use NFS for the root filesystem. For the rest of this text 'client' means the diskless system, and 'server' means the NFS server.

Enabling nfsroot capabilities

In order to use nfsroot, NFS client support needs to be selected as built-in during configuration. Once this has been selected, the nfsroot option will become available, which should also be selected.

In the networking options, kernel level autoconfiguration can be selected, along with the types of autoconfiguration to support. Selecting all of DHCP, BOOTP and RARP is safe.

Kernel command line

When the kernel has been loaded by a boot loader (see below) it needs to be told what root fs device to use. And in the case of nfsroot, where to find both the server and the name of the directory on the server to mount as root. This can be established using the following kernel command line parameters:

root=/dev/nfs

This is necessary to enable the pseudo-NFS-device. Note that it's not a real device but just a synonym to tell the kernel to use NFS instead of a real device.

nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>]

If the *nfsroot' parameter is NOT given on the command line, the default* "`"/tftpboot/%s"`" will be used.

<server-ip> Specifies the IP address of the NFS server.

The default address is determined by the ip parameter (see below). This parameter allows the use of different servers for IP autoconfiguration and NFS.

<root-dir> Name of the directory on the server to mount as root.

If there is a "%s" token in the string, it will be replaced by the ASCII-representation of the client's IP address.

<nfs-options> Standard NFS options. All options are separated by commas.

The following defaults are used:

```
port            = as given by server portmap daemon
rsize           = 4096
wsize           = 4096
timeo           = 7
retrans         = 3
acregmin        = 3
acregmax        = 60
acdirmin        = 30
acdirmax        = 60
flags           = hard, nointr, noposix, cto, ac
```

ip=<client-ip>:<server-ip>:<gw-ip>:<netmask>:<hostname>:<device>:<autoconf>:<dns0-ip>:<dns1-ip>:<ntp0-ip>

This parameter tells the kernel how to configure IP addresses of devices and also how to set up the IP routing table. It was originally called nfsaddrs, but now the boot-time IP configuration works independently of NFS, so it was renamed to ip and the old name remained as an alias for compatibility reasons.

If this parameter is missing from the kernel command line, all fields are assumed to be empty, and the defaults mentioned below apply. In general this means that the kernel tries to configure everything using autoconfiguration.

The <autoconf> parameter can appear alone as the value to the ip parameter (without all the ':' characters before). If the value is "ip=off" or "ip=none", no autoconfiguration will take place, otherwise autoconfiguration will take place. The most common way to use this is "ip=dhcp".

<client-ip> IP address of the client.

Default: Determined using autoconfiguration.

<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.

Only required for NFS root. That is autoconfiguration will not be triggered if it is missing and NFS root is not in operation.

Value is exported to /proc/net/pnp with the prefix "bootserver " (see below).

Default: Determined using autoconfiguration. The address of the autoconfiguration server is used.

<gw-ip> IP address of a gateway if the server is on a different subnet.

Default: Determined using autoconfiguration.

<netmask> Netmask for local network interface.

If unspecified the netmask is derived from the client IP address assuming classful addressing.

Default: Determined using autoconfiguration.

<hostname> Name of the client.

If a '.' character is present, anything before the first '.' is used as the client's hostname, and anything after it is used as its NIS domain name. May be supplied by autoconfiguration, but its absence will not trigger autoconfiguration. If specified and DHCP is used, the user-provided hostname (and NIS domain name, if present) will be carried in the DHCP request; this may cause a DNS record to be created or updated for the client.

Default: Client IP address is used in ASCII notation.

<device> Name of network device to use.

Default: If the host only has one device, it is used. Otherwise the device is determined using autoconfiguration. This is done by sending autoconfiguration requests out of all devices, and using the device that received the first reply.

<autoconf> Method to use for autoconfiguration.

In the case of options which specify multiple autoconfiguration protocols, requests are sent using all protocols, and the first one to reply is used.

Only autoconfiguration protocols that have been compiled into the kernel will be used, regardless of the value of this option:

```
off or none: don't use autoconfiguration
              (do static IP assignment instead)
on or any:   use any protocol available in the kernel
             (default)
dhcp:        use DHCP
bootp:       use BOOTP
rarp:        use RARP
both:        use both BOOTP and RARP but not DHCP
             (old option kept for backwards compatibility)
```

if dhcp is used, the client identifier can be used by following format "ip=dhcp,client-id-type,client-id-value"

Default: any

<dns0-ip> IP address of primary nameserver.

Value is exported to /proc/net/pnp with the prefix "nameserver " (see below).

Default: None if not using autoconfiguration; determined automatically if using autoconfiguration.

<dns1-ip> IP address of secondary nameserver.

See <dns0-ip>.

<ntp0-ip> IP address of a Network Time Protocol (NTP) server.

Value is exported to /proc/net/ipconfig/ntp_servers, but is otherwise unused (see below).

Default: None if not using autoconfiguration; determined automatically if using autoconfiguration.

After configuration (whether manual or automatic) is complete, two files are created in the following format; lines are omitted if their respective value is empty following configuration:

- /proc/net/pnp:

  #PROTO: <DHCP|BOOTP|RARP|MANUAL> (depending on configuration method) domain <dns-domain> (if autoconfigured, the DNS domain) nameserver <dns0-ip> (primary name server IP) nameserver <dns1-ip> (secondary name server IP) nameserver <dns2-ip> (tertiary name server IP) bootserver <server-ip> (NFS server IP)

- /proc/net/ipconfig/ntp_servers:

<ntp0-ip> (NTP server IP) <ntp1-ip> (NTP server IP) <ntp2-ip> (NTP server IP)

<dns-domain> and <dns2-ip> (in /proc/net/pnp) and <ntp1-ip> and <ntp2-ip> (in /proc/net/ipconfig/ntp_servers) are requested during autoconfiguration; they cannot be specified as part of the "ip=" kernel command line parameter.

Because the "domain" and "nameserver" options are recognised by DNS resolvers, /etc/resolv.conf is often linked to /proc/net/pnp on systems that use an NFS root filesystem.

Note that the kernel will not synchronise the system time with any NTP servers it discovers; this is the responsibility of a user space process (e.g. an initrd/initramfs script that passes the IP addresses listed in /proc/net/ipconfig/ntp_servers to an NTP client before mounting the real root filesystem if it is on NFS).

nfsrootdebug

This parameter enables debugging messages to appear in the kernel log at boot time so that administrators can verify that the correct NFS mount options, server address, and root path are passed to the NFS client.

rdinit=<executable file>

To specify which file contains the program that starts system initialization, administrators can use this command line parameter. The default value of this parameter is "/init". If the specified file exists and the kernel can execute it, root filesystem related kernel command line parameters, including 'nfsroot=', are ignored.

A description of the process of mounting the root file system can be found in Documentation/driver-api/early-userspace/early_userspace_support.rst

Boot Loader

To get the kernel into memory different approaches can be used. They depend on various facilities being available:

- Booting from a floppy using syslinux

  When building kernels, an easy way to create a boot floppy that uses syslinux is to use the zdisk or bzdisk make targets which use zimage and bzimage images respectively. Both targets accept the FDARGS parameter which can be used to set the kernel command line.

  e.g:

  ```
  make bzdisk FDARGS="root=/dev/nfs"
  ```

  Note that the user running this command will need to have access to the floppy drive device, /dev/fd0

  For more information on syslinux, including how to create bootdisks for prebuilt kernels, see https://syslinux.zytor.com/

  > **Note**
  >
  > Previously it was possible to write a kernel directly to a floppy using dd, configure the boot device using rdev, and boot using the resulting floppy. Linux no longer supports this method of booting.

- Booting from a cdrom using isolinux

  When building kernels, an easy way to create a bootable cdrom that uses isolinux is to use the isoimage target which uses a bzimage image. Like zdisk and bzdisk, this target accepts the FDARGS parameter which can be used to set the kernel command line.

  e.g:

  ```
  make isoimage FDARGS="root=/dev/nfs"
  ```

  The resulting iso image will be arch/<ARCH>/boot/image.iso This can be written to a cdrom using a variety of tools including cdrecord.

  e.g:

  ```
  cdrecord dev=ATAPI:1,0,0 arch/x86/boot/image.iso
  ```

  For more information on isolinux, including how to create bootdisks for prebuilt kernels, see https://syslinux.zytor.com/

- Using LILO

  When using LILO all the necessary command line parameters may be specified using the 'append=' directive in the LILO configuration file.

  However, to use the 'root=' directive you also need to create a dummy root device, which may be removed after

LILO is run.

e.g:

```
mknod /dev/boot255 c 0 255
```

For information on configuring LILO, please refer to its documentation.

- Using GRUB

  When using GRUB, kernel parameter are simply appended after the kernel specification: kernel <kernel> <parameters>

- Using loadlin

  loadlin may be used to boot Linux from a DOS command prompt without requiring a local hard disk to mount as root. This has not been thoroughly tested by the authors of this document, but in general it should be possible configure the kernel command line similarly to the configuration of LILO.

  Please refer to the loadlin documentation for further information.

- Using a boot ROM

  This is probably the most elegant way of booting a diskless client. With a boot ROM the kernel is loaded using the TFTP protocol. The authors of this document are not aware of any no commercial boot ROMs that support booting Linux over the network. However, there are two free implementations of a boot ROM, netboot-nfs and etherboot, both of which are available on sunsite.unc.edu, and both of which contain everything you need to boot a diskless Linux client.

- Using pxelinux

  Pxelinux may be used to boot linux using the PXE boot loader which is present on many modern network cards.

  When using pxelinux, the kernel image is specified using "kernel <relative-path-below /tftpboot>". The nfsroot parameters are passed to the kernel by adding them to the "append" line. It is common to use serial console in conjunction with pxeliunx, see Documentation/admin-guide/serial-console.rst for more information.

  For more information on isolinux, including how to create bootdisks for prebuilt kernels, see https://syslinux.zytor.com/

Credits

The nfsroot code in the kernel and the RARP support have been written by Gero Kuhlmann <gero@gkminix.han.de>.

The rest of the IP layer autoconfiguration code has been written by Martin Mares <mj@atrey.karlin.mff.cuni.cz>.

In order to write the initial version of nfsroot I would like to thank Jens-Uwe Mager <jum@anubis.han.de> for his help.