Mounting the root filesystem via NFS (nfsroot)
===

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>

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.txt), a ramdisk (see Documentation/initrd.txt) 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.

1.) 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.

2.) 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>

  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.

               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. May be supplied by autoconfiguration,
                 but its absence will not trigger autoconfiguration.

                 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)

                 Default: any

3.) Boot Loader

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

3.1)   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 http://syslinux.zytor.com/

N.B: 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.

3.2) 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/i386/boot/image.iso

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

3.2) 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.

     mknod /dev/boot255 c 0 255

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

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

3.4) 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.

3.5) 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.

3.6) 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/serial-console.txt for more information.

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

4.) 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.