iSCSI Boot

Version 0.2

19. February 2008

Author: Andreas Florath
debml@flonatel.org

Table of Contents

1 Introduction

The aim of this project is, to get iSCSI boot for Linux™ working.

1.1 Wording

The key words **must**, **must not**, **required**, **shall**, **shall not**, **should**, **should not**, **recommended**, **may** and **optional** in this document are to be interpreted as described in [RFC 2119].

2 Requirements

The following requirements must be fulfilled:

2.1 Root on iSCSI target

The Linux™ operating system **must** be booted in the way, that it's root directory is located on a iSCSI target.

2.2 Kernel and initrd

The Linux™kernel and possible an initrd **must** be retrieved via a common boot protocol like dhcp / tftp / bootp. Alternative the kernel and initrd **must** be stored locally when using a virtualization environment like Xen.

Note: It is not required that the kernel and initrd are stored on the iSCSI root disk.

2.3 Multipath

During all stages of operation the Linux™ operating system **must** be able to reach all file systems over all configured paths.

Short: Multipath in all stages to all disks is a **must**.

2.4 Parameter passing

It **must** be possible to configure all aspects of booting by

- command line parameters given during boot
- some attribute in a dhcp response.

Note: This allows also to mix the different types. Example: some parameters can be given by command line parameters, other in the dhcp response.

It **must** be possible to specify all needed configuration options in the dhcp response.

Note: Of course the network devices must be specified in the command line.

3 Command Line Parameters

3.1 Current implementation

Currently there are some different command line parameters given to the init script for handling root file pointing – most, but not all of them can also be handled by the kernel itself. These are:

- **root** This is used for a couple of things:
  - When starting with `LABEL=` then the root file system is taken from `/dev/disk/by-label/<label>`
  - When starting with `UUID=` then the root file system is taken from `/dev/disk/by-uuid/<uuid>`
  - If this is set to `/dev/nfs` the root file system is imported with NFS. (This is mostly the same as specifying `boot=nfs`, except that it does not overwrite any `boot=` parameters that were specified before.)
  - In all other cases the given value is taken as the path to the root device.
- **rootflags** Gives additional flags when mounting the root file system.
- **rootfstype** Specifies the file system type of the root file system.
- **rootdelay** Maximum amount of time that is waited to get a ready to use root file system.
- **nfsroot** Gives the name of the NFS server and the path of the disk. Is only used, when `boot=nfs` is specified. The parameter has the form `nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>]`
- **boot** This can hold 'nfs' or 'local'. 'nfs' is for NFS root based boot, 'local' for local disks. Note that also FC attached disks are seen as local here.

The IP configuration is done with the ip parameter. This has the form

```
ip=<client-ip>:<server-ip>:<gw-ip>:<netmask>\
:<hostname>:<device>:<autoconf>
```

3.2 Drawbacks of Current Situation

The current implementation has some drawbacks:

- The `ip` parameter can only handle exactly one device. It is therefore currently not possible to handle more than one network device.
- There are some fields from the `ip` parameter that are unused for normal operation. Note: the `ip` parameter historically was named `nfsaddrs` and the server part was the NFS server.
- The `root` parameter is overloaded and has some strange dependency to the `boot` parameter.

3.3 Proposals

3.3.1 rfs

The parameter `rfs` (for Root File System) is the generic parameter for handling all aspects of the root file system detection and mounting. It is composed from some words that are separated by a colon ':'. The first word gives the type how to mount the root file system. All further options are a semicolon separated list of key=value pairs.

```
rfs=<type>:k1=val1;k2=val2;...
```

All further parameters – name and semantic – depend on this first word and are described in the corresponding sections

3.3.1.1 General options

Some variables can be used for some or all boot methods. These are described in this section.

3.3.1.1.1 delay

In some circumstances the root file system does not appear immediately. To specify a maximum wait time, the delay parameter can be used. The value is the number of seconds to wait for.

Default value: 180

3.3.1.1.2 flags

The given `flags` are used to the `-o` option of the mount command during mounting the root file system.

Default value: <empty>

3.3.1.1.3 fstype

The file system type of the root file system can specified with the fstype parameter.

Default value: <empty>

3.3.1.1.4 path

The path variable defines the full path to the root device.

Example:

```
rfs=local:path=/dev/hda1
```

This can be used with local and iscsi type.

3.3.1.1.5 label

The `label` variable is short for `path=/dev/disk/by-label/<label>`.

Example:

```
rfs=local:label=dnsserver-root
```

is the same as

```
rfs=local:path=/dev/disk/by-label/dnsserver-root
```

This can be used with local and iscsi type.

3.3.1.1.6 uuid

The `uuid` is short for `path=/dev/disk/by-uuid/<uuid>`.

Example:

```
rfs=local:uuid=fccac673-51e4-42d2-855a-c796de4d0d90
```

is the same as

```
rfs=local:path=/dev/disk/by-uuid/fccac673-51e4-42d2-855a-c796de4d0d90
```

This can be used with local and iscsi type.

3.3.1.2 Type: iscsi

When booting from an iSCSI target, some more special parameters are needed.

The parameters `portals`, `localiqn` are mandatory. All other parameters are optional.

The variables `path`, `label` and `uuid` specify the path to the root device. Only one can be specified. If more than one is specified the behaviour is undefined.

3.3.1.2.1 portals

This is a comma separated list of portals that are used to discover iSCSI disks.

Example:

```
portals=192.168.129.1:3260,192.168.130.1:3260
```

3.3.1.2.2 localiqn

The localiqn is the IQN from the client that wants to boot. It is not the IQN of the server. The server IQN is retrieved during the discovery process. This is the reason, why it is not possible to use the procedure described in [RFC 4173]; this mandatory parameter is missing in this document.

Example:

```
localiqn=iqn.1987-05.com.company:01.1b3c44ad847
```

3.3.1.3 Type: local

For specifying local disks, the type local must be used.

The variables `path`, `label` and `uuid` specify the path to the root device. Only one can be specified. If more than one is specified, than the behaviour is undefined.

3.3.1.4 Type: nfs

When using NFS as root, the server and dir variables must be given.

3.3.1.4.1 server

This specifies the IP address of the NFS server.

Example:

```
server=192.168.126.1
```

3.3.1.4.2 dir

This variable holds the directory of the NFS on the server side.

Example:

```
dir=/otherboot/dnsserver
```

3.3.1.5 Examples

3.3.1.5.1 Local with noatime

```
rfs=local:path=/dev/hda1;flags=noatime
```

3.3.1.5.2 iSCSI multipath

```
rfs=iscsi:path=/dev/mapper/dnsserver-root-part1;\
portals=192.168.129.1:3260,192.168.130.1:3260,\
192.168.131.1:3260,192.168.132.1:3260;\
localiqn=iqn.1987-05.com.company:01.e6d74b5134a2
```

3.3.2 ips

ToDo...

Bibliography

RFC 2119: Bradner, S., "Key words for use in RFCs to Indicate Requirement

Levels", 1997,http://www.ietf.org/rfc/rfc2119.txt
RFC 4173: Sarkar, P.; Missimer, D.; Sapuntzakis, C., "Bootstrapping Clients using the Internet Small Computer System Interface (iSCSI) Protocol", 2005,http://www.ietf.org/rfc/rfc4173.txt