Linux System Loader Description

Michael Löhr

$Id: specs.lyx,v 1.9 2006/02/08 14:49:11 loehr Exp $

Contents

1 Overview

This document describes the installation and configuration of the system loader. Every available option of the system loader configuration file is treated in detail. In addition the usage of the system loader user interface is shown. The appendix provides example configurations.

2 Description

The system loader is an extension of the `kboot` project by Werner Almesberger (kboot homepage `http://kboot.sourceforge.net/`). It runs in a minimal Linux environment that has to be started by a platform specific first stage boot loader. It will read a boot configuration file from a location specified by an URI and display a boot selection menu. Based on the user selection it loads a boot configuration and reinitializes the system via the `kexec` system call. This second stage of the boot process will directly replace the already running minimal Linux system by the system selected in the boot configuration. Because the second stage is controlled dynamically by the system loader configuration file there is no need to rerun the first stage loader when the boot configuration changes.

3 Building

In most cases the end user of the system loader will use prebuilt components and has therefore no need to build the system loader from scratch. If it is necessary to rebuild the system loader components there are three possibilities.

3.1 The `kboot` Build Environment

As an extension to the `kboot` environment the system loader is integrated into the `kboot` build environment and into the Makefile. As a result a `make` in the `kboot` top level will download and compile all required components and finally create a system loader kernel and ramdisk. This ramdisk contains the kboot base environment plus the system loader extensions described in this document. Additional information about this build environment can be found in the original `kboot` documents by Werner Almesberger.

3.2 RPM Package Build

This build option is only available on RPM based Linux distributions. Executing `make redhat-rpm` in the `kboot` top level directory will first execute `make` in the `kboot` build environment and finally create source and binary RPM packages of the system loader. The standard kboot build environment must be working without problems before building RPMs.

3.3 System Loader Standalone Build

It is possible to build the system loader as a standalone application without the whole kboot kernel and ramdisk environment. Calling make in the `kboot/ui` directory will create this version of the system loader. It can be used for debugging purposes or to initiate a fast menu controlled reboot of the system.

4 Installation

In the case of a manual installation it is necessary to copy the system loader kernel and ramdisk to the `/boot` directory. The installation of a system loader rpm package will put the files to `/boot` as well. In both cases the systemloader is not activated automatically.

Instead of the kernel that is provided with the system loader package it is also possible to use a kernel that is already available on the system. The kernel version of such a kernel should be at least 2.6.13 and the kexec system call has to be enabled. It is also important that all modules that are required to access the boot files (e.g. disk drivers) are compiled into this kernel. Kernels and ramdisks are different for 31/32-bit and 64-bit systems and should not be mixed.

To make the system loader usable on the target system it has to be configured as one of the possible boot selections of the first stage loader. On i386 `lilo` or `grub` can be used as first stage loaders, on s390 the system loader can be booted from the reader or it may use `zipl` as the first stage loader. A minimal first stage loader is sufficient because the system loader will handle the boot selection menu and user interaction.

Before the system loader can be used a valid system loader configuration file has to be written. A detailed description of all elements of the system loader configuration file is given in section 6. Example configuration files can be found in sections A.4 and A.2. When the system loader configuration file is available the system loader kernel and initrd can be booted via the first stage boot loader. Sections A.3 and A.1 show example configurations for `grub` and `zipl`.

The system loader will be started if the kernel command line contains an additional parameter `kboot` that specifies the location of the configuration file. Otherwise the `kboot` environment will be started without the system loader.

Example:

```
kboot=dasd://(0.0.5c5e,1)/boot/kboot.conf
```

5 Boot Methods

Depending on the hardware platform and the available disk types the system loader supports several boot methods. On the i386 and s390 platform the system can be booted from a kernel image file. An initial ramdisk and a kernel command line or parmfile can be specified together with this kernel file. On the s390 platform `*.ins` files or the boot map information written by `zipl` can be used in addition.

In any case the location of kernel, ramdisk, parmfile, insfile or boot map will be defined as an URI. Which URI scheme has to be used depends on the system platform and available disk. The `block` URI is typically used on the i386 platform. The `dasd` and `zfcp` URI schemes are specific for the s390 platform.

All boot methods and URI schemes will be described in detail in the following chapter.

6 Elements of the Config File

The config file for the system loader consists of some global definitions which are followed by one or more boot entries. External files are always referenced by an URI. A detailed description of supported URIs will be given in section 6.3.

6.1 Global Definitions

Preceding the boot entries the system loader configuration file contains a number of definitions that are valid for the whole configuration file.

6.1.1 Comments

Comments are allowed in the global definitions part of the configuration file and in the boot entries. A comment starts with # and ends at the end of the line. Tabs or spaces in front of the # are possible but comments at the end of a line that contains a definition are not allowed.

Example:

```
# this is a comment
        # this is also a comment

<definition> # this is not allowed!
```

6.1.2 default

The `default` statement references the boot entry that will be selected automatically after the timeout. Valid references are the label of a boot entry or a number. Implicit numbering of the boot entries starts with 0. If the default statement is not used, entry number 0 will be used as the default.

Example:

```
default linux2
default 1
```

6.1.3 timeout

The `timeout` statement specifies the time in seconds until the default boot entry will be started automatically. If the statement is not used or the timeout is set to 0 no timeout will occur.

Example:

```
timeout 15
```

6.1.4 password

The `password` statement defines the password that has to be entered if a locked boot entry (see section 6.2.8) has been selected.

Example:

```
password topsecret
```

6.1.5 userinterface

The `userinterface` statement specifies which userinterface module should be started to display the boot selection menu. For every userinterface line a userinterface process will be started. This allows to start several instances of the same userinterface module listening on different input devices. Additional options may be specified depending on the user interface module. Currently `linemode` is the only available user interface module. It allows to specify the device on which the boot selection menu will be displayed.

Syntax:

```
userinterface MODULE_NAME [MODULE_OPTIONS]
```

Example:

```
userinterface linemode /dev/tty1
userinterface linemode /dev/tty
```

6.1.6 include

The include statement can be used anywhere in the system loader configuration file. The content of the file file refereced by the URI will be included at the position of the include statement. Nested includes are allowed but limited to twenty levels.

Syntax:

```
include URI
```

Example:

```
include dasd://(0.0.5c5e,1)/boot/extra_menu.conf
```

6.2 Boot Entry Definitions

The global definitions are followed by a section of boot entries which describe the available boot configurations. A boot entry starts with the line

```
boot_entry {
```

and ends with a closing bracket

```
}
```

Every boot entry must contain exactly one statement that triggers a boot action. These boot action statements are kernel, insfile, bootmap, halt and reboot. Other statements can be used optionally. Some of them will specify labels or variables to improve the handling of the configuration file, some will influence the behaviour of the boot menu and some will pass additional information to the booted system. All possible statements inside the boot entry are described in the following sections.

6.2.1 title

The title statement defines a text that will be displayed in the boot selection menu and should give some meaningful description of the boot configuration. This statement is mandatory for all boot entries.

Example:

```
title Debian GNU/Linux, latest kernel
```

6.2.2 label

The `label` statement allows to assign a symbolic label to a boot entry. This label can be used in the default statement to reference the default boot entry (see section 6.1.2). If no label definition is present the boot entry can still be referenced numerically by its position in the configuration file.

Example:

```
label linux3
```

6.2.3 root

The `root` statement defines an URI path prefix and will be prepended to all URIs[1] specified in the same boot entry. Typically it is used to specify a common path for kernel, initrd and parmfile. Note that the path prefix and the rest of the URI will be concatenated as they are specified. There is no automatic insertion of a `'/'` character and no syntax checking.

Example:

```
root dasd://(0.0.5c5e,1)/boot/
```

6.2.4 kernel

The `kernel` statement specifies the kernel file that will be booted if this boot entry is selected. All supported URI formats are allowed to specify the location of the kernel file. If a root statement is given in the same boot entry, it will be prepended to the specified kernel path.

Example:

```
kernel vmlinuz
```

6.2.5 initrd

The `initrd` statement specifies the initial ramdisk that will be used if this boot entry is selected. All supported URI formats are allowed to specify the location of the ramdisk file. If a root statement is given in the same boot entry, it will be prepended to the specified ramdisk path.

Example:

```
initrd initrd.img
```

6.2.6 cmdline

The cmdline statement specifies the kernel commandline that is used to start the kernel if this boot entry is selected.

Example:

```
cmdline ro root=/dev/ram0 ramdisk_size=100000
```

[1] The root definition will not be applied to `bootmap` URIs because a path definition makes no sense for a structure that is not part of any filesystem.

6.2.7 parmfile

As an alternative to the cmdline statement the `parmfile` statement can be used to specify the file that will be used as the kernel command line if this boot entry is selected. All supported URI formats are allowed to specify the location of the parmfile. If a root statement is given in the same boot entry, it will be prepended to the specified parmfile path. If parmfile and cmdline are specified at the same time they will be concatenated as `parmfile + ' ' + cmdline` .

Example:

```
parmfile parmfile.txt
```

6.2.8 lock

The `lock` command allows to have password protected boot entries. If a locked boot entry is selected the user has to enter the password specified in the `password` statement (see section 6.1.4) to execute this boot selection entry.

Example:

```
lock
```

6.2.9 pause

The `pause` statement displays a message and waits for user input before the boot entry will be started. This can be used to ask the user to prepare the system for booting (e.g. by inserting a boot CD to the CD drive).

Example:

```
pause Please insert your boot floppy!
```

6.2.10 insfile

The `insfile` statement can be used as an alternative method to specify a boot configuration. An `*.ins` file contains the definitions of a kernel, initial ramdisk and parmfile, therefore it makes no sense to specify these components . The `insfile` statement is only available on the s390 platform and is provided with several Linux distributions for this platform.

Example:

```
insfile dasd://(0.0.5c5e,1)/usr/local/insfile_test/redhat/generic.ins
```

6.2.11 bootmap

The `bootmap` command can be used as an alternative method to specify a boot configuration. It is only available on the s390 platform and boots the system using the boot information from the boot map of the specified disk. This boot information is written by the tool `zipl`. Only bootmaps in the format created by `zipl` version 1.2 or newer are supported. The bootmap command uses a modified URI format that is described in section 6.3.5 and 6.3.6.

Example:

```
bootmap dasd://(0.0.5e2a,0)
```

6.2.12 halt

The `halt` statement can be used instead of a real boot selection. It will halt the Linux environment of the system loader.

6.2.13 reboot

The `reboot` statement can be used instead of a real boot selection. This statement will reboot the system. If a first stage bootloader is installed the system will be restarted via this bootloader.

6.3 URI Definitions

URIs provide a generic mechanism to describe the location of files. They are used by the system loader to locate boot and configuration files. The following sections describe the supported URI schemes in detail.

6.3.1 block

The `block` URI can be used to reference a file on any block device containing a supported filesystem. This is the typical method to access the boot files on the i386 platform and on other non s390 platforms. If no filesystem type is specified, the filesystem type will be autodetected.

Syntax:

```
block://(<device node>[,<filesystem type>])/<path to file>
```

Example:

```
block://(/dev/hda5,ext3)/boot/vmlinuz
```

6.3.2 dasd

The `dasd` URI can be used to reference files on zSeries ESCON/FICON attached storage. This type of URI is only available on the s390 platform. If no filesystem type is specified, the filesystem type will be autodetected.

Syntax:

```
dasd://(<bus id>,[<partition>[,<filesystem type>]])/<path to file>
```

Example:

```
dasd://(0.0.5c5e,1)/usr/local/insfile_test/SuSE/suse.ins
```

6.3.3 zfcp

The zfcp URI can be used to reference files on s390 (zSeries) FCP attached storage. This type of URI is only available on the s390 platform. If no filesystem type is specified, the filesystem type will be autodetected.

Syntax:

```
zfcp://(<bus id>,<WWPN>,<LUN>,[<partition>[,<filesystem type>]])/<path to file>
```

Example:

```
zfcp://(0.0.54e0,0x5005076303000104,0x4011400500000000,1)/boot/initrd.img
```

6.3.4 `file`

The `file` URI can be used to reference files on an already mounted filesystem. This type of URI is available on all platforms. In an unmodified system loader boot environment it can only be used to reference files on the initial ramdisk. If the system loader is running as a program on an already booted Linux system it can be used instead of the other URIs.

Example:

```
file:///boot/vmlinuz
```

6.3.5 `dasd` URI for the bootmap command

Because the bootmap information is not stored inside a file system the bootmap command (see section 6.2.11) uses a reduced form of the dasd URI.

Syntax:

```
dasd://(<bus id>[,<program number>])
```

Example:

```
dasd://(0.0.5e89,1)
```

6.3.6 `zfcp` URI for the bootmap command

Because the bootmap information is not stored inside a file system the bootmap command (see section 6.2.11) uses a reduced form of the zfcp URI.

Syntax:

```
zfcp://(<bus id>,<WWPN>,<LUN>[,<program number>])
```

Example:

```
zfcp://(0.0.04ae,0x500507630e01fca2,0x4010404500000000,2)
```

7 Using the Command Line Interface

The command line interface is the basic user interface of the system loader. It is designed to run on all platforms and without any special requirements regarding the capabilities of the user interface device. Therefore it will be usable on line mode interfaces like the 3270 terminal on the s390 platform or via a serial line on typical open systems platforms.

The command line interface displays the title information from all boot entries found in the system loader configuration file. The default entry is marked by an arrow symbol `->`, locked entries are displayed in brackets `[]`.

```
kboot user interface is starting.
Configuration file source: file:///boot/boot_menu.config

Welcome to kboot!

The following boot options are available:

-> 1 Debian GNU/Linux, latest kernel
   2 Debian GNU/Linux, rescue kernel
  [3 Debian GNU/Linux, rescue kernel (locked)]
   4 kboot environment
   5 System Reboot
   6 System Halt
   7 INSFILE Boot

Please enter your selection:
```

The selection of a specific entry is done by entering its number. If the menu is not completely visible it can be redisplayed by entering an empty input. If a timeout occurs while one or more user interfaces are waiting for input, all user interfaces will be terminated and the default entry will be executed. If a timeout is defined it will be displayed by the user interface.

8 Using Online Documentation

Man pages are available for the `kboot` executable and for the system loader configuration file `kboot.conf`. These man pages are meant as a short overview and refer to this specification and to the system loader design document for a detailed description of this software.

9 Extending the System Loader

The system loader is designed to be extensible. The extension can be done without changing the code of the existing components. User provided loader modules can provide support for additional URI schemes (e.g. ftp, http). Additional user interface modules can provide more comfortable user interfaces (e.g. graphical or web-based). Interface definitions are given in the design documents.

10 Dependencies on Other Software

The system loader is designed as an extension to the kboot environment by Werner Almesberger. Therefore in most cases it will be used within this environment and depends on all components that make up the kboot environment. It is possible to use the system loader within any other Linux runtime environment. In this case the main dependencies are the availablility of the kexec system call in the kernel and the kexec tool. In addition the current implementation depends on several basic Linux tools and programs like `udev`, `sh`, `awk`, `mount`, `umount`, `mkdir`, `rmdir`, `usleep`, etc. . The `/bin` , `/sbin`, `/usr/bin` and `/usr/sbin` directories of the `kboot` ramdisk provide a still small subset of these programs and can be used as a guideline.

A Examples

A.1 `zipl.conf`

The following example shows a configuration file for the zipl boot loader on the s390 platform. It selects the kboot based system loader as the default boot option and offers an additional kernel that can be booted directly.

```
# This is an expample zipl.conf file
[defaultboot]
defaultmenu = menu
[linux-2.6]
target     = "/boot"
image      = "/boot/image-2.6.14-15.x.20051116-s390xdefault"
parmfile   = "/boot/parmfile"
[kboot]
target     = "/boot"
image      = "/boot/image.kboot"
ramdisk    = "/boot/initrd.kboot"
parameters = "kboot=dasd://(0.0.5c5e,1)/boot/kboot.conf"
[dump]
target     = "/boot"
dumpto     = "/dev/dasd?1"
:menu
target     = "/boot"
1          = "linux-2.6"
2          = "kboot"
default    = 2
prompt     = 1
timeout    = 15
```

A.2 `kboot.conf` for s390

The following example configuration for the system loader uses most of the options that are available for the system loader on the s390 platform.

```
#
# My first kboot config file
#

default linux1
timeout 20

userinterface linemode /dev/console

#include dasd://(0.0.5e98,1)/boot/kboot/kboot_incl.conf

boot_entry {
        title latest kernel from DASD
        label linux1

        kernel dasd://(0.0.5c5e,1)/boot/image
        cmdline dasd=5c5e-5c5f root=/dev/dasda1 ro noinitrd selinux=0
}

boot_entry {
        title kboot (New Prototype)
        label kboot

        kernel dasd://(0.0.5c5e,1)/boot/image.kboot
        initrd dasd://(0.0.5c5e,1)/boot/initrd.kboot
        cmdline kboot=dasd://(0.0.5c5e,1)/boot/kboot.conf
        pause Please insert your boot floppy!
}

boot_entry {
        title kboot (development version)
```

```
        label kboot2

        root dasd://(0.0.5c5e,1)/boot/
        kernel image
        initrd kboot-root-glibc.cpio.gz
        cmdline kboot=dasd://(0.0.5c5e,1)/boot/kboot.conf
}

boot_entry {
        title kboot (Werner Almesberger Version)
        label kboot_wa

        root dasd://(0.0.5c5e,1)/boot/
        kernel vmlinuz.kboot
        initrd initrd.kboot
        cmdline ro root=/dev/ram0 ramdisk_size=100000
}

boot_entry {
        title Linux 2.6.13-14.x from FCP disk
        label linux2

        root zfcp://(0.0.54e0,0x5005076303000104,0x4011400500000000,1)/boot/

        kernel image-2.6.13-14.x.20050907-s390xdefault
        # cmdline dasd=5e2a,5e29 root=/dev/dasda1 ro noinitrd selinux=0
        parmfile parmfile
}

boot_entry {
        title System Reboot
        label reboot

        reboot
}

boot_entry {
        title System Halt

        halt
}

boot_entry {
        title IBM insfile

        insfile dasd://(0.0.5c5e,1)/usr/local/insfile_test/IBM/IBM-Linux.ins
}

boot_entry {
        title SuSE insfile

        insfile dasd://(0.0.5c5e,1)/usr/local/insfile_test/SuSE/suse.ins
}

boot_entry {
        title redhat insfile

        insfile dasd://(0.0.5c5e,1)/usr/local/insfile_test/redhat/generic.ins
}

boot_entry {
        title Boot Map Boot

        bootmap dasd://(0.0.5e2a,0)
}
```

A.3 `menu.lst` for grub

This example shows how the system loader can be booted as default on the i386 platform using grub as the first stage boot loader.

```
# Deutsche Tastaturbelegung
```

```
setkey y z
setkey z y
setkey Y Z
setkey Z Y
setkey equal parenright
setkey parenright parenleft
setkey parenleft asterisk
setkey doublequote at
setkey plus bracketright
setkey minus slash
setkey slash ampersand
setkey ampersand percent
setkey percent caret
setkey underscore question
setkey question underscore
setkey semicolon less
setkey less numbersign
setkey numbersign backslash
setkey colon greater
setkey greater bar
setkey asterisk braceright
# menu.lst - See: grub(8), info grub, update-grub(8)
#            grub-install(8), grub-floppy(8),
#            grub-md5-crypt, /usr/share/doc/grub
#            and /usr/share/doc/grub-doc/.

## default num
# Set the default entry to the entry number NUM. Numbering starts from 0, and
# the entry number 0 is the default if the command is not used.
default 1

## timeout sec
# Set a timeout, in SEC seconds, before automatically booting the default entry
# (normally the first entry defined).
timeout 5

# Pretty colours
color cyan/blue white/blue
gfxmenu (hd0,4)/boot/message

## password ['--md5'] passwd
# If used in the first section of a menu file, disable all interactive editing
# control (menu entry editor and command-line)  and entries protected by the
# command 'lock'
# e.g. password topsecret
#      password --md5 $1$gLhU0/$aW78kHK1QfV3P2b2znUoe/
# password topsecret

#
# examples
#
# title Windows 95/98/NT/2000
# root (hd0,0)
# makeactive
# chainloader +1
#
# title Linux
# root (hd0,1)
# kernel /vmlinuz root=/dev/hda2 ro
#

#
# Put static boot stanzas before and/or after AUTOMAGIC KERNEL LIST

### BEGIN AUTOMAGIC KERNELS LIST
## lines between the AUTOMAGIC KERNELS LIST markers will be modified
## by the debian update-grub script except for the default optons below

## DO NOT UNCOMMENT THEM, Just edit them to your needs

## ## Start Default Options ##
## default kernel options
## default kernel options for automagic boot options
## If you want special options for specifiv kernels use kopt_x_y_z
## where x.y.z is kernel version. Minor versions can be omitted.
## e.g. kopt=root=/dev/hda1 ro
# kopt=root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce vga=0x317

## default grub root device
```

```
## e.g. groot=(hd0,0)
# groot=(hd0,4)

## should update-grub create alternative automagic boot options
## e.g. alternative=true
##      alternative=false
# alternative=false

## should update-grub lock alternative automagic boot options
## e.g. lockalternative=true
##      lockalternative=false
# lockalternative=false

## altoption boot targets option
## multiple altoptions lines are allowed
## e.g. altoptions=(extra menu suffix) extra boot options
##      altoptions=(recovery mode) single
# altoptions=(recovery mode) single

## controls how many kernels should be put into the menu.lst
## only counts the first occurence of a kernel, not the
## alternative kernel options
## e.g. howmany=all
##      howmany=7
# howmany=all

## ## End Default Options ##

title Debian GNU/Linux, kernel
root (hd0,4)
kernel /boot/vmlinuz root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce lapic vga=normal
initrd /boot/initrd.img
boot

title kboot system loader
root (hd0,4)
kernel /boot/image.kboot ro root=/dev/ram0 ramdisk_size=100000 lang=de apm=power-off nomce lapic vga=normal
initrd /boot/initrd.kboot
boot

### END DEBIAN AUTOMAGIC KERNELS LIST
title Windows 2K/XP/2003 (hda1)
chainloader (hd0,0)+1
```

A.4 `kboot.conf` for i386

The following example show an example of a system loader configuration for the i386 platform. Two userinterface instances are started to listen on two different devices simultaneously.

```
#
# This is a sample config file for kilo
#

default linux1
timeout 0
password secret

userinterface linemode /dev/tty1
userinterface linemode /dev/tty

boot_entry {
        title Debian GNU/Linux, latest kernel
        label linux1

        root block://(/dev/hda5,ext3)/boot/
        kernel vmlinuz
        initrd initrd.img
        cmdline root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce lapic vga=normal
}

boot_entry {
        title Debian GNU/Linux, rescue kernel
```

```
        label linux2

        root block://(/dev/hda5,ext3)/boot/
        kernel vmlinuz-2.6.11-kanotix-7
        initrd initrd.img-2.6.11-kanotix-7
        cmdline root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce vga=0x317
}

boot_entry {
        title Debian GNU/Linux, rescue kernel (locked)
        label linux3

        lock
        root block://(/dev/hda5,ext3)/boot/
        kernel vmlinuz-2.6.11-kanotix-7
        initrd initrd.img-2.6.11-kanotix-7
        cmdline root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce vga=0x317
}

boot_entry {
        title kboot environment
        label kboot

        root block://(/dev/hda5,ext3)/boot/
        kernel vmlinuz
        initrd kboot-root-glibc.cpio.gz
        cmdline root=/dev/hda5 ro ramdisk_size=100000 lang=de apm=power-off nomce lapic vga=normal

}

boot_entry {
        title System Reboot
        label reboot

        reboot
}

boot_entry {
        title System Halt

        halt
}
```