Red-Paper: KIWI Project

OpenSuSE - KIWI Image System System Design

Project, Design and Implementation
by Marcus Schaefer (ms@suse.de)
*

Author: Marcus Schaefer
Datum: July 25, 2006
Revision: 1.2

*SUSE LINUX Products GmbH, Maxfeldstrasse 5, D-90409 Nuremberg

Contents

1 Introduction

The OpenSuSE KIWI Image System provides a complete operating system image solution for Linux supported hardware platforms as well as for virtualisation systems like Xen. The KIWI architecture was designed as a two level system. The first stage, based on a valid **software package source**, creates a so called **physical extend** according to the provided image description. The second stage creates from a required physical extend an operating system image. The result of the second stage is called a **logical extend** or short an image.

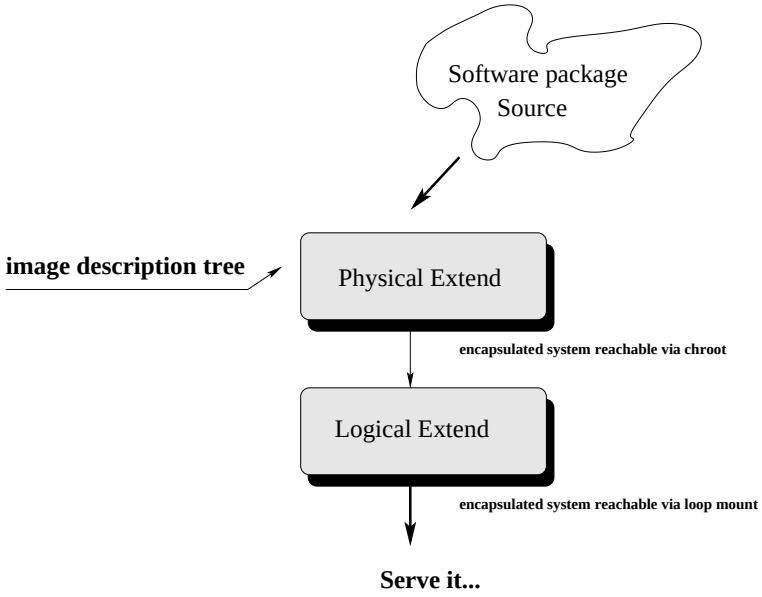

Figure 1.1: Image Serving Architecture

Because this document contains conceptual information about an image system, it is important to understand what an operating system image is all about. A normal installation process is starting from a given installation source and installs single pieces of software until the system is complete. During this process there may be manual user intervention required. However an operating system image represents an already completed *installation* encapsulated as a file and optionally includes the configuration for a specific task. Such an operating system starts working as soon as the image has been brought to a system storage device no matter if this is a volatile or non volatile storage. The process of creating an image takes place without user interaction. This means all requirements of the encapsulated system has to be fulfilled before the image is created. According to this the so called **image description tree** stores all the information needed to create an image.

1.1 What is KIWI good for

The solution introduced in this document covers an implementation for the following two major topics:

- How to create physical and logical extends
- How to serve/activate a logical extend, an image

As already said a physical extend requires a valid software package source. This is the location where KIWI can access the software to build up a system. Software exists as packages and packages exists in different formats. The amount of all packages are organized in a package tree including some meta data, this is called a repository. With the KIWI project I want to be free of choice what kind of package repository should be used. To achieve this goal this project will make use of the **smart** package manager which is very fast and can handle the most important repository structures. Information on smart can be found here:

- http://labix.org/smart

1.1.1 Supported image types

To create a logical extend it is necessary to create a filesystem the operating system data can be stored in. The image type corresponds to the selected filesystem. Supported image types are:

- CPIO
- EXT2 / EXT3
- ReiserFS
- Special Boot image for image installation via network
- Special Xen image based on ReiserFS and Xen kernel

2 Creating Operating System Images

Contents

The creation of operating system images is based on image description trees. An image description contains all the files in a directory that are required to generate an image using the Perl-based image builder **kiwi.pl**. The directory to which the description files are written must contain a VERSION file with a three-part version number of the format:

Major.Minor.Release

- For smaller image modifications that do not add or remove any new packages, only the release number is incremented. The **config.xml** file remains unchanged.
- For image changes that involve the addition or removal of packages the minor number is incremented and the release number is reset.
- For image changes that change the size of the image file the major number is incremented.

2.1 Structure of the Image Description Tree

- **VERSION**
  This file contains the version number of the image description tree, for example, 1.1.2.
- **root**
  Subdirectory that contains special files, directories, and scripts for adapting the image environment **after** the installation of all the image packages. The entire directory is copied into the root of the image tree using *cp -a*.

- **config**
  Optional Subdirectory that contains Bash scripts that are called after the installation of all the image packages, primarily in order to remove the parts of a package that are not needed for the operating system. The name of the Bash script must resemble the package name listed in the config.xml

- **config.sh**
  Optional configuration script while creating the physical extend. This script is called at the end of the installation but **before** the package scripts have run. It is designed to configure the image system, such as the activation or deactivation of certain services (insserv). The call is not made until after the switch to the image has been made with **chroot**.

- **images.sh**
  Optional configuration script while creating the logical extend. This script is called at the beginning of the image creation process. It is designed to clean-up the image system. Affected are all the programs and files only needed while the physical extend exists.

- **config.xml**
  Configuration file that indicates the image type, base name, options, and which packages make up the image. The structure of the file corresponds to the format

```
<image name="Name">
    <preferences>
        <type>Type</type>
        <size unit="Unit">Size</size>
        <compressed>Yes/No</compressed>
    </preferences>
    <drivers type="Type">
        <file name="Filename"/>
    </drivers>
    <repository type="Type">
        <source path="URL"/>
    </repository>
    <packages type="Type">
        <package name="Packagename"/>
    </packages>
</image>
```

The config.xml file contains four major parts embedded into a XML im-

age tag. The image tag contains the attrubute **name** to indicate the base name of the image. It is automatically expanded using the version number and the date. The version number is extracted from the directory in which the description files for this image are located.

1. The **preferences** tag contains information needed to create the logical extend. For this tag the following subtags are defined:
   - **size**
     Image size as a number whereas the attribute **unit** defines the scale unit **M** or **G** standing for Megabyte or Gigabyte. Note: *kiwi* supports the feature extending the image size automatically to a calculated size, if the specified config size value is to small. If the config size value plus the additional size needed to build the image is more than 100MB, *kiwi* will abort with an error message. Furthermore *kiwi* will not reduce the image size automatically by design, because it must be possible to configure additional space, for example, if custom scripts are run.
   - **type**
     The image type of the logical extend: **ext2**, **ext3**, **cpio**, **reiserfs**. Different formats are possible, if necessary.
   - **timezone**
     The time zone. The possible time zones are located in the directory */usr/share/zoneinfo*. For the image itself, only one time zone each is required. For this reason, the relative path to the time zone to use in the image is indicated after the **timezone** key, for example, **Europe/Berlin**. kiwi uses this information to extract the corresponding time zone from the timezone package and to store it as */etc/localtime* in the image.
   - **keytable**
     Contains the name of the console keymap to use. The name corresponds to a map file stored below the path */usr/share/kbd/keymaps*. Furthermore, the variable *KEYTABLE* within the file */etc/sysconfig/keyboard* will be set according to the keyboard mapping.
2. The optional **drivers** tag contains driver file names. The names are interpreted as general driver name and captured if they are contained in the kernel tree. The attribute **type** specifies one of the following driver types:
   - **netdrivers**
     Every file is indicated relative to the directory */lib/modules/<Version>/kernel/drivers/net*
   - **drivers**
     Every file is indicated relative to the directory */lib/modules/<Version>/kernel*

3. The **repository** tag defines the source path and type used by smart. The attribute **type** setup the the smart type of the repository, for example, **type="yast2"** and the subtag **source** contains the attribute **path** to setup the the smart location of the repository, for example, **source="/image/CDs/full-i386"**. The path specifaction can be done as:
   - local path starting with /
   - **html:**// or **ftp:**// Network-Location
   - opensuse://Project-Name

   Multiple repository tags are allowed. For information on how to setup a smart source refer to *http://labix.org/smart*
4. The **packages** tag contains a list of package names whereas the attribute **type** specifies under which cirumstances this package set needs to be used. There are three different types of package sets:
   - **image**
     used to finish the image installation. All packages which makes up the image are listed there.
   - **boot**
     used to start buildig the image. Basic components like libc or the smart package manager are listed here.
   - **xen**
     used when the image needs support for Xen based virtualisation. Option **–virtual xen**

In addition all values entered as **image** tag of the config.xml file are stored in a file called **.profile**. The file is created before the execution of an image script like *config.sh, images.sh* or a *package script* and can then be sourced. The parameters of the config file are then available as variables in the script and can be processed appropriately. The script itself is called within the image environment, which means it is not possible to damage the host system with your script even if you are using absolute paths. Such a script should look like the following template:

```
#!/bin/sh
echo -n "Image [<name>]..."
test -f /.profile && . /.profile

... script code

echo done
```

The parameter **name** should be the name of the image to which this script belongs.

3 Activating an image

Contents

After a logical extend (an image) has been created from a physical extend there are in principal three possibilities to activate the image:

1. In case of a **local accessable system harddisk**, the image can be installed by dumping (dd) the image file on a previously created partition on this disk. To activate the system a boot manager like grub or lilo can be used

2. In case of a **network enabled system** (netboot client) the image can be installed via a special boot image. The boot image which serves as initial ramdisk (initrd) and the appropriate kernel are downloaded from a network service. The linux kernel automatically calls a program named **linuxrc** which takes over all tasks needed to download and install the system image. The installation can be done persistently on disk or temporary into the RAM of the machine.

3. In case of a **virtual target system** like Xen, the image can be be installed by copying and loop mounting the image file on the target system. To activate the virtual system the loop mounted location needs to be configured according to the needs of the virtualisation system.

3.1 The KIWI netboot image

The KIWI netboot image can be used to install an operating system image to a network client. To establish communication with the client a boot server infrastructure including the following services is required:

- DHCP server to give the client an IP address
- TFTP server to allow file transfer from/to the client

3.1.1 The Boot Process of a netboot System

To understand how to use the operating system image with the netboot image, a short description of the netboot client is useful. The following diagram shows the simplified boot process of a netboot client.

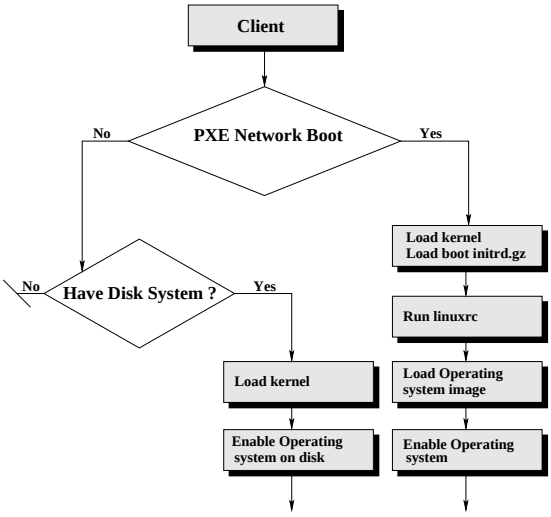

Figure 3.1: The Boot Process of a netboot client

The most important point is to understand the cooperation between the initial boot image **initrd.gz** and the intrinsic operating system image. If the system is able to boot via a network, it will load the kernel and the compressed boot image from the network. The *brain* of the boot image is the **linuxrc** script, which does all the stuff controlled by an image configuration file also obtained from the network. The major task is to download and activate the operating system image. The boot image is exchanged for the operating system image to be activated. The following overview describes the steps that take place when the netboot client is booted:

- Via PXE network boot or boot manager (GRUB), the client boots the initrd (initrd.gz) that it receives from the TFTP server. If no PXE boot is possible, the client tries to boot from the hard disk, if accessible.
- Running **linuxrc** starts the process described below.

1. The required file systems to receive system data are mounted. Example: **proc** file system.

2. Network support is activated. A default list of modules is used and are tried one after the other. The module is loaded using *modprobe*. Any dependencies to other modules are cleared at that time.

3. The network interface is set up via DHCP. After the interface has been established, the DHCP variables are exported into the file */var/lib/dhcpcd/dhcpcd-eth0.info* and the contents of DOMAIN and DNS are used to generate a */etc/resolv.conf*.

4. The TFTP server address is acquired. During this step, a check is first made to determine whether the host name **tftp.$DOMAIN** can be resolved. If not, the DHCP server is used as the TFTP server. For more infomation about the TFTP servers structure, refer to Section 3.1.2

5. The configuration file is loaded from the server directory */var/lib/tftpboot/KIWI* via TFTP. At this point, the client expects the file:

   config.<MAC Address>

   If this file is not available and cannot be loaded, it means this is a new client that can be immediately registered. A new client is registered by uploading a control file to the TFTP servers upload directory: */var/lib/tftpboot/upload*. After the upload, the client branches off into a loop in which the following steps are taken:

   - the DHCP lease file is renewed (*dhcpcd -n*).
   - a new attempt is made to load the file config.<MAC address> from the TFTP server.
   - if the file does not exist, there is a 60-second wait period before a new run begins.

   If the configuration file does load, it contains data on image, configuration, synchronization, or partition parameters. For more infomation about the file format of the configuration file, refer to Section 3.1.3.

6. The PART: line in the configuration is analyzed. If there is a PART line in the configuration file, the following analysis takes place:

   - A check is made using the image version to see whether any local system needs to be updated. If not, local boot process continues immediately. No image download occurs.
   - No system or update required: client hard disk is partitioned.

7. Indicated images are downloaded with multicast TFTP.

8. Checksums checked. Repeat download if necessary.

9. The CONF: line is evaluated. All the indicated files are loaded from the TFTP server and stored in a */config/* path.
10. Terminate all the user-land processes based on the boot image (dhcpcd -k).
11. operating system image is mounted.
12. The configuration files stored in */config/...* are copied into the mounted operating system image.
13. The system switches to the mounted operating system image. The root file system is converted to the operating system image via **pivot_root**. All the required configuration files are now present, because they had been stored in the operating system image or have been downloaded via TFTP.
14. The boot image is unmounted using an **exec umount** call.
15. At termination of linuxrc or the exec call, the kernel initiates the **init** process that starts processing the boot scripts as specified **/etc/inittab**, for example, to configure the network interface.

3.1.2 TFTP Server Structure

The TFTP server directory structure is divided into the following main areas

- **Image configurations**
  The */var/lib/tftpboot/KIWI/* directory contains the various *config.<MAC Address>* image configuration files.
- **Configuration files**
  The */var/lib/tftpboot/KIWI/<MAC Address>/* directory contains the various system configuration files, such as xorg.conf.
- **Boot files**
  The */var/lib/tftpboot/boot/* directory is where the initrd.gz, the kernel to boot, and the PXE loader pxelinux.0 are kept.
- **PXE configuration file**
  The */var/lib/tftpboot/boot/pxelinux.cfg* directory is where the PXE configuration file is kept.
- **Image files and checksums**
  The */var/lib/tftpboot/image/* directory is where all the image files and their checksums are kept.
- **Upload area**
  The directory */var/lib/tftpboot/upload/* is the directory into which the *hwtype.<MAC Address>* files for registering new netboot clients are uploaded.

3.1.3 The netboot client Configuration File

This section describes the netboot client configuration file:

config.<MAC Address>

The configuration file contains data about image, configuration, synchronization, or partition parameters. The configuration file is loaded from the TFTP server directory */var/lib/tftpboot/KIWI* via TFTP for previously installed netboot clients. New netboot clients are immediately registered and a new configuration file with the corresponding MAC address is created. Below, find an example of a cash register configuration file:

```
IMAGE=/dev/hda2;image/browser;1.1.1;192.168.1.1;4096
CONF=/KIWI/00:30:05:1D:75:D2/ntp.conf;/etc/ntp.conf;192.168.1.1;1024,      \
     /KIWI/00:30:05:1D:75:D2/xorg.xonf;/etc/X11/xorg.xonf;192.168.1.1;1024
PART=200;S;x,300;L;/,500;L;/opt,x;L;/home
DISK=/dev/hda
```

The following format is used:

IMAGE=device;name;version;srvip;bsize;compressed,...,
SYNC=syncfilename;srvip;bsize
CONF=src;dest;srvip;bsize,..., src;dest;srvip;bsize
PART=size;id;Mount,...,size;id;Mount
JOURNAL=ext3
DISK=device

- **IMAGE**
  Specifies which image (name) should be loaded with which version (version) and to which storage device (device) it should be linked, e.g., **/dev/ram1** or **/dev/hda2**. The netboot client partition (device) **hda2** defines the root file system "/" and `hda1` is used for the swap partition. The numbering of the hard disk device should not be confused with the RAM disk device, where `/dev/ram0` is used for the initial RAM disk and can not be used as storage device for the second stage system image. SUSE recommends to use the device `/dev/ram1` for the RAM disk. If the hard drive is used, a corresponding partitioning must be performed.

- **srvip**
  Specifies the server IP address for the TFTP download. Must always be indicated, except in PART.

- **bsize**
  Specifies the block size for the TFTP download. Must always be indicated, except in PART. If the block size is too small according to the maximum number of data packages (32768), **linuxrc** will automatically calculate a new blocksize for the download.

- **compressed**
  Specifies if the image file on the TFTP server is compressed and handles it accordingly. To specify a compressed image download only the keyword "'**compressed**"' needs to be added. If compressed is not specified the standard download workflow is used. **Note:** The download will fail if you specify "'compressed"' and the image isn't compressed. It will also fail if you don't specify "'compressed"' but the image is compressed. The name of the compressed image has to contain the suffix **.gz** and needs to be compressed with the **gzip** tool. Using a compressed image will automatically **deactivate** the multicast download option of atftp.

- **CONF**
  Specifies a comma-separated list of source:target configuration files. The source (src) corresponds to the path on the TFTP server and is loaded via TFTP. The download is made to the file on the netboot client indicated by the target (dest).

- **PART**
  Specifies the partitioning data. The comma-separated list must contain the size (size), the type number (id), and the mount point (Mount).

  - The first element of the list must define the swap partition.
  - The second element of the list must define the **root** partition.
  - The swap partition must not contain a mount point. A lowercase letter **x** must be set instead.
  - If a partition should take all the space left on a disk one can set a lower **x** letter as size specification.

- **DISK**
  Specifies the hard disk. Used only with PART and defines the device via which the hard disk can be addressed, e.g., **/dev/hda**.

- **RELOAD_IMAGE**
  If set to a non-empty string, forces the configured image to be loaded from the server even if the image on the disk is up-to-date. Used mainly for debugging purposes, this option only makes sense on diskful systems.

- **RELOAD_CONFIG**
  If set to an non-empty string, forces all config files to be loaded from the server. Used mainly for debugging purposes, this option only makes sense on diskful systems.

3.1.4 The netboot client Control File

This section describes the netboot client control file:

```
hwtype.<MAC Address>
```

The control file is primarily used to set up new netboot clients. In this case, there is no configuration file corresponding to the client MAC address available. Using the MAC address information, the control file is created, which is uploaded to the TFTP servers upload directory */var/lib/tftpboot/upload*.

3.2 The Xen virtual machine

To use an image within Xen the most important point is to use the para virtualized xen kernel within the operating system image. While creating a xen based image **kiwi** will take care for the correct kernel to be used.

3.2.1 Activate a KIWI xen image

To activate an image including the xen kernel the following steps needs to be performed:

1. loop mount the image

   ```
   mount -o loop <image> <loop path>
   ```

2. create a xen configuration file and use the loop mounted path as disk parameter:

   ```
   disk = [ 'phy:<loop path>' ]
   ```

   for more information on how to configure xen refer to
   http://en.opensuse.org/Installing_Xen3

3. boot the system using the command:

   ```
   xm create -c <xen config file>
   ```

Index