FAI Guide (Fully Automatic Installation)

Thomas Lange <lange@informatik.uni-koeln.de>

Version 1.3.1 for FAI version 2.2.3, 15 november 2001

Abstract

This manual describes the fully automatic installation package for Debian GNU/Linux. This includes the installation of the package, the planing and creating of the configuration and how to deal with errors.

Copyright Notice

Contents

Chapter 1

Introduction

The homepage of FAI is `http://www.informatik.uni-koeln.de/fai`. There you will find any information about FAI, for e.g. the mailing list archive. The FAI package is also available as a Debian package from `www.informatik.uni-koeln.de/fai/download`. It's an official Debian package and is available from all Debian mirrors. Send any bug or comment to `<fai@informatik.uni-koeln.de>`. You can also use the Debian bug tracking system (BTS) `http://www.debian.org/Bugs` for reporting errors.

You can access the CVS repository from a Bourne shell using the following commands. The login password is empty, so only press return.

```
> CVSROOT=:pserver:anonymous@cvs.debian.org:/cvs/debian-boot
> cvs login
> cvs co -P fai-kernels
> cvs co -P fai
```

You can also use the web interface for the CVS repository at: `http://cvs.debian.org/fai/` (and `fai-kernels`).

Now read this manual, then enjoy the fully automatic installation and your saved time.

1.1 Motivation

Have you ever performed identical installations of an operating system several times? Would you like to be able to install a Linux cluster with dozens of nodes single handedly?

Repeating the same task time and again is boring — and will surely lead to mistakes. Also a whole lot of time could be saved, if the installation were done automatically. An installation process with manual interaction does not scale. But clusters have the habit of growing over the years. Think long–term rather than plan only just a few months into the future.

In 1999, I had to organize an installation of a Linux cluster with one server and 16 clients. Since I had much experience doing automatic installation of Solaris operating system on SUN SPARC hardware, the idea to build an automatic installation for Debian was born. Solaris has an automatic installation feature called JumpStart[1]. In conjunction with the auto–install scripts from Casper Dik[2], I could save a lot of time not only for every new SUN computer, but also for reinstallation of existing workstations. For example, I had to build a temporary LAN with four SUN workstations for a conference, lasting only a few days. I took these workstations out of our normal research network and set up a new installation for the conference. When it was over, I simply integrated the workstation back into the research network, rebooted just once, and after half an hour, everything was up and running as before. The configuration of all workstations was exactly the same as before the conference, because everything was performed by the same installation process. I also use the automatic installation for reinstalling a workstation after a damaged hard disk had been replaced. It took two weeks until I received the new hard disk but only a few minutes after the new disk was installed, the workstation was running as before. And this is why I chose to adapt this technique to a PC cluster running Linux.

1.2 Overview and concepts

FAI is a non interactive system to install a Debian GNU/Linux operating system on a single computer or a whole cluster. You can take one or more virgin PC's, turn on the power and after a few minutes Linux is installed, configured and running on the whole cluster, without any interaction necessary. Thus, it's a scalable method for installing and updating a cluster unattended with little effort involved. FAI uses the Debian GNU/Linux distribution and a collection of shell and perl scripts for the installation process. Changes to the configuration files of the operating system can be made by cfengine, shell, perl and expect scripts.

FAI's target group are system administrators how have to install Debian onto one or even hundreds of computers. Because it's a general purpose installation tool, it can be used for installing a Beowulf cluster, a rendering farm or a linux laboratory or a classroom. Also large–scale linux networks with different hardware or different installation requirements are easy to establish using FAI. But don't forget to plan your installation. 'How to plan your installation' on page 25 has some useful hints for this topic.

First, some terms used in this manual are described.

install server : The host, where the package FAI is installed. It provides several services for all install clients. In the examples of this manual this host is called `kueppers`.

install client : A host, which will be installed using FAI and a configuration from the install server. Also called client for short. In this manual the example hosts are called `bigfoot, ant01, ant02, nucleus, atom01, atom02,...`

configuration : The details, how the installation of the clients should be performed. This includes information about:

- Hard disk layout
- Local filesystems

[1] Solaris 8 Advanced Installation Guide at `docs.sun.com`
[2] `ftp://ftp.wins.uva.nl:/pub/solaris/auto-install/`

- Software packages
- Keyboard layout, time zone, NIS, X11 configuration, remote filesystems, user accounts, printers …

nfsroot : A filesystem located on the install server. It's the complete filesystem for the install clients during the installation process. All clients share the same nfsroot, which they mount read only.

1.3 How does FAI work ?

The install client which will be installed using FAI, is booted from floppy disk or via network card. It gets an IP address and boots a linux kernel which mounts its root filesystem via NFS from the install server. After the operating system is running, the FAI startup script performs the automatic installation which doesn't need any interaction. First, the hard disks will be partitioned, filesystems are created and then software packages are installed. After that, the new installed operating system is configured to your local needs using some scripts. Finally the new operating system will be booted from the local disk.

The details, of how to install the computer (the configuration), are stored in the configuration space on the install server. Configuration files are shared among groups of computers if they are similar using the class concept. So you need not to create a configuration for every new host. Hence, FAI is a scalable method to install a big cluster with a great number of nodes.

FAI can also be used as an network rescue system. You can boot your computer, but it will not perform an installation. Instead it will run a fully functional Debian GNU/Linux without using the local hard disks. Then you can do a remote login and backup or restore a disk partition, check a filesystem, inspect the hardware or do any other task.

1.4 Features

- A fully automated installation can be perfomed
- Very quick unattended installation
- Hosts can boot from floppy or from network card
- BOOTP and DHCP protocol are supported
- No initial ramdisk is needed, 8MB RAM suffice
- Runs even on a 386 CPU
- The installation kernel can use modules
- Remote login via ssh during installation process possible
- Two additional virtual terminals available during installation
- All similar configuration are shared among all install clients

- Log files are saved on to the installation server
- Shell, perl, expect and cfengine scripts are supported for the configuration setup
- Access to a Debian mirror via NFS, FTP or HTTP
- Easy creation of the common boot floppy
- Keyboard layout selectable
- Can be used as a rescue system
- Tested on SUN SPARC hardware
- Flexible system through easy class concept
- Predefined Beowulf classes included
- Diskless client support
- Easily add your own functions via hooks
- Easily change the default behavior via hooks
- Lilo and grub support
- ReiserFS support

Chapter 2

Installing FAI

2.1 Requirements

Following items are required for an installation via FAI.

A computer: The computer must have a network interface card and a local hard disk. No floppy disk, CD–ROM, keyboard or graphic card is needed.

BOOTP or DHCP server: The clients need one of these daemons to obtain boot information.[1]

TFTP server: The TFTP daemon is used for transfering the kernel to the clients. Only needed when booting from network card with a boot PROM.

Client root: A mountable directory which contains the whole filesystem for the install clients during installation. It will be created during setup of the FAI package and is also called **nfsroot**.

Debian mirror: Access to a Debian mirror is needed. A local mirror of all Debian packages is recommended if you install several computers.

Install kernel: A kernel image that supports the network card and mounts its root filesystem via NFS.

Configuration space: A mountable directory which contains the configuration data.

The TFTP daemon and a NFS server will be enabled automatically when installing the FAI package. Different install kernel images for BOOTP and DHCP are available within the package `fai-kernels`. All clients must have a network card, which is recognized by the install kernel.

[1] It's also possible without a server, if all information is put on the boot floppy.

2.2 How to create a local Debian mirror

The script `mkdebmirror` [2] can be used for creating your own local Debian mirror. This script uses the script `debmirror`[3] and `rsync(1)`. A partitial Debian mirror only for i386 architecture for Debian 2.2r4 (aka potato) without the source packages needs about 2.4 GB of disk space.

2.3 Setting up FAI

Before installing FAI, you have to install the package `fai-kernels`, which contains the install kernels for FAI. [4]

```
kueppers[~]# dpkg -i fai-kernels_1.1.3_i386.deb
Selecting previously deselected package fai-kernels.
(Reading database ... 40562 files and directories currently installed.)
Unpacking fai-kernels (from .../lange/fai-kernels_1.1.3_i386.deb) ...
Setting up fai-kernels (1.1.3) ...
```

Get the newest version of FAI and install it using the `dpkg` command:

```
kueppers[~]# dpkg -i fai_2.2.3_all.deb
Selecting previously deselected package fai.
(Reading database ... 39564 files and directories currently installed.)
Unpacking fai (from fai_2.2.3_all.deb) ...
Setting up fai (2.2.3) ...
To set up FAI edit /etc/fai.conf and call fai-setup
```

All definitions for the FAI package (not the configuration data) are defined in `/etc/fai.conf`. Since FAI doesn't use `debconf` yet, edit this file before calling `fai-setup`. These are important variables in `/etc/fai.conf`:

FAI_DEBOOTSTRAP For building the nfsroot there's a new command called `debootstrap(8)`, which replaces the use of the *FAI_BASETGZ* variable. It needs the location of a Debian mirror and the distribution (potato,woody,stable,unstable) for which the basic Debian system should be build.

FAI_BASETGZ The location, where the the base file is fetched from. For building the nfsroot, the Debian base system is needed. It's a big tar file (approx. 16MB for `base2_2.tgz`), which is a minimal collection of all required packages for Debian. If a file `base2_2.tgz` is already available in `/tmp` this file will be used instead and no file will be fetched from the defined location. Fetching via ftp or http could take much time, if you do not have a fast connection to your Debian mirror. You can find the current version of this archive in the directory `debian/dists/stable/main/disks-i386/current/` of a Debian mirror.

[2] Available in `/usr/share/fai/utils/`.

[3] Available at `http://cvs.kitenet.net/joey-cvs/bin/debmirror` or at the FAI homepage.

[4] `# apt-get install fai` should also work.

FAI_SOURCES_LIST This multi line string is the definition for `sources.list` (used by `apt-get(8)`), which defines the location and access method of the Debian mirror. If this variable is undefined, the definitions of `/etc/apt/sources.list` will be used. For more information on the file format see `sources.list(5)`.

FAI_DEBMIRROR If you have NFS access to your local Debian mirror, specify the remote filesystem. It will be mounted to *$MNTPOINT* (defined in `/etc/fai.conf`). It's not needed if you use access via FTP or HTTP.

KERNELPACKAGE You must specify the software package – build with `make-kpkg(8)` – which includes the default kernel for booting the install clients. The Debian package `fai-kernels` contains the default install kernels with BOOTP and DHCP support. Do not use the kernel with BOOTP support when you have a DHCP server running in your network and vice versa. This could lead to missing information.

NFSROOT_PACKAGES Contains a list of additional software package, that will be added to nfsroot.

The variables *FAI_SOURCES_LIST* and *FAI_DEBMIRROR* are used by the install server and also by the clients. If your install server has multiple network card and different host names for each card (as for a Beowulf server), use the install server name which is known by the install clients.

FAI uses `apt-get(8)` to create the nfsroot filesystem in `/usr/lib/fai/nfsroot`, which needs about 100MB of free disk space. Before setting up FAI, you should get the program `imggen`,[5] if you like to boot from a 3Com network card. This executable converts netboot images created by `mknbi-linux(8)`, so they can be booted by network cards from 3Com. Put that executable in your path (e.g. `/usr/local/bin`). After editing `/etc/fai.conf` call `fai-setup`.

```
kueppers[~]# fai-setup
Adding system user fai...
Stopping Name Service Cache Daemon: nscd.
Adding new user fai (100) with group nogroup.
Starting Name Service Cache Daemon: nscd.
Creating home directory /home/fai.
/home/fai/.rhosts created.
User account fai created.
Creating FAI nfsroot can take a long time and will
need more than 100MB disk space in /usr/lib/fai/nfsroot.
Unpacking /tmp/base2_2.tgz
Upgrading /usr/lib/fai/nfsroot
Adding additional packages to /usr/lib/fai/nfsroot:
dhcp-client file rdate cfengine bootpc wget rsh-client less dump
ext2resize strace hdparm parted dnsutils grub ntpdate psmisc hwtools
dosfstools sysutils ssh expect5.31
grep: /etc/ssh/sshd_config: No such file or directory
```

[5] Available at the download page `http://www.ltsp.org` or from the FAI download page `www.informatik.uni-koeln.de/fai/download`.

```
grep: /etc/ssh/sshd_config: No such file or directory
Not starting OpenBSD Secure Shell server (/etc/ssh/NOSERVER)
make-fai-nfsroot finished.
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon...done.
Starting NFS kernel daemon: nfsd mountd.
Exporting directories for NFS kernel daemon...done.
Kernel image file name  = /usr/lib/fai/nfsroot/boot/vmlinuz-2.2.19
Output file name        = /boot/fai/installimage
Kernel command line     = "auto rw root=/dev/nfs nfsroot=kernel nfsaddrs=kerne

Image Creator for MBA ROMs v1.00
Usage: imggen [OPTION] inputfile outputfile
  -a,   Add 3Com MBA/BootWare support
  -r,   Remove 3Com MBA/BootWare support from image file
  -i,   Show information on an image
  -h,   Help screen

In filename: /boot/fai/installimage
Out filename: /boot/fai/installimage_3com
Adding MBA support...
MBA support has been successfully added
You have no FAI configuration. Copy FAI template files with:
cp -a /usr/share/fai/templates/* /usr/local/share/fai
Then change the configuration files to meet your local needs.
FAI setup finished.
```

The setup routine adds lines to `/etc/exports` to export the nfsroot and the configuration space to all hosts that belong to the netgroup *faiclients*. If you already export a parent directory of these directories, you may comment out these lines, since the kernel nfs server has problems exporting a directory and one of its subdirectories with different options. All install clients must belong to this netgroup, in order to mount these directories successfully. Netgroups are defined in `/etc/netgroup` or in the corresponding NIS map. An example for the netgroup file can be found in `/usr/share/doc/fai/examples/etc/netgroup`. For more information, read the manual pages `netgroup(5)` and the NIS HOWTO. After changing the netgroups , the NFS server has to reload its configuration. Use one of the following commands, depending on which NFS server you are using:

```
kueppers[~]# /etc/init.d/nfs-kernel-server reload
kueppers[~]# /etc/init.d/nfs-server reload
```

The setup also creates the account `fai` ($LOGUSER). The log files of all install clients are saved to the home directory of this account. If you boot from network card, you should change the primary group of this account, so this account has write permissions to `/boot/fai` in order to change symbolic links to the kernel image, that is booted by a client. See also variable *TFTPLINK* in `class/DEFAULT.var`.

After that, FAI is installed successfully on your server, but has no configuration for the install clients. Start with the templates from `/usr/share/fai/templates` using the copy command above and read 'Installation details' on page 27. Before you can set up a DHCP or BOOTP daemon, you should collect some network information of all your install clients. This is described in section 'Creating a boot floppy' on page 12.

When you make changes to `/etc/fai.conf` or want to install a new kernel to nfsroot, the nfsroot has to be rebuild by calling `make-fai-nfsroot`.

2.3.1 Troubleshooting the setup

The setup of FAI adds the FAI account, exports file systems and calls `make-fai-nfsroot`. If you call `make-fai-nfsroot -v` you will see more messages. When using a local Debian mirror it's important, that the install server can mounted this directory via NFS. If this mount fails, check `/etc/exports` and `/etc/netgroup`. An example can be found in `/usr/share/doc/fai/examples/etc/netgroup`.

Chapter 3

Preparing booting

Before booting for the first time, you have to choose which medium you use for booting. You can use the boot floppy or configure the computer to boot via network card using a boot PROM, which is much smarter.

3.1 Booting from 3Com network card with boot PROM

If you have a 3Com network card that is equipped with a boot ROM by Lanworks Technologie or already includes the DynamicAccess Managed PC Boot Agent (MBA) software[1], you can enter the MBA setup by typing `Ctrl+Alt+B` during boot. The setup will look like this:

```
Managed PC Boot Agent (MBA) v4.00
(C) Copyright 1999 Lanworks Technologies Co. a subsidiary of 3Com Corporation
All rights reserved.
==============================================================================
                            Configuration

Boot Method:                TCP/IP
Protocol:                   BOOTP
Default Boot:               Network
Local Boot:                 Enabled
Config Message:             Enabled
Message Timeout:            3 Seconds
Boot Failure Prompt:        Wait for key
==============================================================================
  Use cursor keys to edit: Up/Down change field, Left/Right change value
  ESC to quit, F9 restore previous settings, F10 to save
```

Set the boot method to `TCP/IP` and the protocol to either `BOOTP` or `DHCP`. I prefer BOOTP because the daemon automaticly reloads its configuration when it's changed. Make a symbolic link from the

[1] `http://support.3com.com/ infodeli/tools/nic/mba.htm`

hostname of your client to the appropriate kernel image in /boot/fai. In the following example the host is called bigfoot. The file installimage_3com is created by imggen and suitable for booting 3Com network cards.

```
kueppers[~]# cd /boot/fai
kueppers[~]# ln -s installimage_3com bigfoot
```

3.2 Booting from network card with a PXE conforming boot ROM

Some network cards (e.g. Intel EtherExpress PRO 100) have a fixed configuration for booting using the PXE protocol. This requires a PXE Linux boot loader an a special version of the TFTP daemon. See /usr/share/doc/syslinux/pxelinux.doc.gz for more information, how to boot such an environment. There are also some mails in the FAI mailing list archive concerning this topic. Sometimes you do not need a netboot image for booting instead you can use the raw kernel image. So, copy the kernel image from the nfsroot to the TFTP directory:

```
# cp /usr/lib/fai/nfsroot/vmlinuz-2.2.19 /boot/fai/installimage
```

3.3 Creating a boot floppy

If your network card can't boot itself, you have to boot via floppy. Use the command make-fai-bootfloppy to create the boot floppy. Since there's no client specific information on this floppy, it's suitable for all your install clients. You can also specify additional kernel parameters for this boot floppy, if desired. There's more help available.

```
# make-fai-bootfloppy -h
```

If you have no BOOTP or DHCP server, supply the network configuration as kernel parameters. [2]

3.4 Collecting Ethernet addresses

Now it's time to boot your install clients for the first time. They will fail to boot completely, because no BOOTP or DHCP daemon is running yet or recognizes the hosts. But you can use this first boot attempt to easily collect all Ethernet addresses of the network cards.

[2] The format is:

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

For additional information see /usr/src/linux/Documentation/nfsroot.txt in the kernel sources.

You have to collect all Ethernet (MAC) addresses of the install clients and assign a hostname and IP address to each client. To collect all MAC addresses, now boot all your install clients. While the install clients are booting, they send broadcast packets to the LAN. You can log the MAC addresses of these hosts, if following command is running simultaneously on the server:

```
# tcpdump -qte broadcast and port bootpc >/tmp/mac.lis
```

After the hosts has been sent some broadcast packets (they will fail to boot because `bootpd` isn't running or does not recognize the MAC address yet) abort `tcpdump` by typing `ctrl-c`. You get a list of all unique MAC addresses with these commands:

```
# perl -ane 'print "\U$F[0]\n"' /tmp/mac.lis|sort|uniq
```

After that, you only have to assign these MAC addresses to hostnames and IP addresses (`/etc/ethers` and `/etc/hosts` or corresponding NIS maps). With these information you can configure your `BOOTP` or `DHCP` daemon (see the section 'Configuration of the BOOTP daemon' on the current page). I recommend to write the MAC addresses (last three bytes will suffice if you have network cards from the same vendor) and the hostname in the front of each chassis.

3.5 Configuration of the BOOTP daemon

An example configuration for the BOOTP daemon is included in FAI. If you have no `/etc/bootptab` file you can use `/usr/share/doc/fai/examples/etc/bootptab` as template.

```
# /etc/bootptab example for FAI
# replace FAISERVER with the name of your install server

.faiglobal:\
 :ms=1024:\
 :hd=/boot/fai:\
 :hn:bs=auto:\
 :rp=/usr/lib/fai/nfsroot:
# rp: $NFSROOT

# your local values
# sa: your tftp server (install server)
# ts: your timeserver (time enabled in inetd.conf)

# these are optional
# ys: NIS server
# yd: NIS domainname
# nt: list of NTP servers
```

```
    .failocal:\
     :tc=.faiglobal:\
     :sa=FAISERVER:\
     :ts=FAISERVER:\
     :T170="FAISERVER:/usr/local/share/fai":\
     :T171="sysinfo":\
     :sm=255.255.255.0:\
     :gw=134.95.9.254:\
     :dn=informatik.uni-koeln.de:\
     :ds=134.95.9.136,134.95.100.209,134.95.100.208,134.95.140.208:\
     :ys=rubens:yd=informatik4711.YP:\
     :nt=time.rrz.uni-koeln.de,time2.rrz.uni-koeln.de:

    # now one entry for each install client
    bigfoot:ha=0x00105A240012:bf=bigfoot:tc=.failocal:T171="sysinfo":T172="verbose
atevt debug":
    ant01:ha=0x00105A000000:bf=ant01:tc=.failocal:T172="sshd":
```

Insert one line for each install client at the end of this file as done for *bigfoot* and *ant01*. Replace the string `FAISERVER` with the name of your install server. If the install server has multiple network cards and host names, use the host name of the network card to which the install clients are connected. Then adjust the other network tags (`sm, gw, dn, ds`) to your local needs.

sm : Subnet mask

gw : Default gateway / router

dn : Domain name

ds : List of DNS server

T170 Location of the FAI configuration directory

T171 FAI_ACTION

T172 List of *FAI_FLAGS*; e.g. verbose, debug, reboot, createvt, sshd

T173 Reserved for future use

T174 Reserved for backup devices and backup options; NOT YET USED

The tags for NIS and time servers (`yp, yd, nt`) are optional. Tags with prefix `T` (starting from T170) are generic tags which are used to transfer some FAI specific data to the clients. **It is important, that T171 (equivalent to variable *FAI_ACTION*[3]. is set to `sysinfo` !** Later you can set it to `install`, in order to start the automatic installation. For more information on all tags see `bootptab(5)`. The

[3] Theses names are used in the main installation script `rcS_fai`. The configuration files for DHCP and BOOTP daemons use other names. Example: *FAI_ACTION* is equal to `T171` in bootptab or `option-171` in dhcp.conf.

list of *FAI_FLAGS* can be space or comma separated. *FAI_FLAGS* in `bootptab` must be separated by whitespace. If you define *FAI_FLAGS* as additional kernel parameter, the flags must be separated with a comma.

When you have created your `bootptab` file, you have to enable the BOOTP daemon once. It's installed but Debian does not enable it by default. Edit `/etc/inetd.conf` and remove the comment (the hash) in the line containing `#bootps`. Then tell `inetd` to reload its configuration.

```
# /etc/init.d/inetd reload
```

I recommend to use the BOOTP daemon and protocol for booting because it automatically reloads the configuration file if any changes are made to it. The daemon for DHCP must always be manually restarted after changes to the configuration file are made [4]. Now it's time to boot all install clients again! FAI can perform several actions when the client is booting. This action is defined in the variable *FAI_ACTION* Be very carefully if you set *FAI_ACTION* to *install*. This can destroy all your data on the install client, indeed most time it should do this ;–). It's recommended to change this only on a per client base in the BOOTP configuration. Do not change it in the section `.failocal` in `/etc/bootptab`, which is a definition for all clients.

3.5.1 Troubleshooting BOOTP daemon

The BOOTP daemon can also be started in debug modus, if it is not enabled in `inetd.conf`:

```
# bootpd -d7
```

3.6 Configuration of the DHCP daemon

An example for `dhcp.conf(5)` is available in `/usr/share/doc/fai/examples/etc`. Start using this example and look at all options used therein. If you make any changes to this configuration, you must restart the daemon.

3.7 Boot messages

These are the messages, when booting from floppy.

```
LILO Loading FAI-BOOTP.
Uncompressing Linux... OK, booting the Kernel.
Linux version 2.2.19 (root@kueppers) (gcc version 2.95.2 20000220
.
.
.
```

[4] If you can't use one of these, there's also the possibility to supply all information via a file or compile them into the kernel, but it's easier to use a daemon.

The rest of the boot message will be equal to those when booting from network card. When booting from network card you will see:

```
BOOTP.
TFTP....
Linux Net Boot Image Loader Version 0.8.1 (netboot)
.
Uncompressing Linux... OK, booting the Kernel.
Linux version 2.2.19 (root@kueppers) (gcc version 2.95.2 20000220
.
.
.
Sending BOOTP requests ..... OK
IP-Config: Got BOOTP answer from 134.95.9.149
IP-Config: Complete:
 device=eth0, addr=134.95.9.200, mask=255.255.255.0, gw=134.95.9.254,
 host=ant01, domain=informatik.uni-koeln.de, nis-domain=informatik4711.YP,
 bootserver=134.95.9.149, rootserver=134.95.9.149, rootpath=/usr/lib/fai/nfsr
.
.
   -------------------------------------------------------
     FAI 2.2.3, 15 november, 2001
     Fully Automatic Installation for Debian GNU/Linux

     Copyright (c) 1999-2001, Thomas Lange
             lange@informatik.uni-koeln.de
   -------------------------------------------------------

Calling task_confdir
Kernel parameters:auto rw root=/dev/nfs
Defining variable: root=/dev/nfs
Sending BOOTP request using device eth0
/fai mounted from kueppers:/usr/local/share/fai
Calling task_setup
.
.
Calling task_action
FAI_ACTION: install
Performing FAI installation. All data may be overwritten !
.
.
Press <RETURN> to reboot or ctrl-c to execute a shell
```

When the copyright message is shown, the install client has mounted the nfsroot[5] to the clients root directory `/`. This is the whole filesystem for the client at this moment. When `/fai` is mounted, the configuration data from the install server is available on the client.

Following error message indicates, that your install client doesn't get an answer from a BOOTP server. Check your cables or start the `bootpd(8)` daemon with the debug flag enabled.

```
Sending BOOTP requests ........ timed out!
IP-Config: Retrying forever (NFS root)...
```

If you get the following error message, the install kernel has no driver compiled in for your network card.

```
IP-Config: No network devices available
Partition check:
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 >
Root-NFS: No NFS server available, giving up.
VFS: Unable to mount root fs via NFS, trying floppy.
VFS: Insert root floppy and press ENTER
```

Then you have to compile a new kernel has a driver compiled in, which supports your network card. This driver must no be a kernel module. To compile the new kernel, start using the default kernel configuration of FAI.

```
kueppers# cd /usr/src/kernel-source-2.2.19
kueppers# cp /usr/lib/fai/nfsroot/boot/config-2.2.19 .config
kueppers# make menuconfig
```

Call `make menuconfig` and add the driver in menu `Network device support/Ethernet` which supports your network card. Then create a Debian package using `make-kpkg(8)`:

```
kueppers# make-kpkg clean
kueppers# make-kpkg --revision BOOTP2 kernel-image
```

This command creates the file `/usr/src/kernel-image-2.2.19_BOOTP2_i386.deb`. Adjust the variable *KERNELPACKAGE* in `/etc/fai.conf` and rebuild the nfsroot.

```
kueppers# make-fai-nfsroot
```

After that, you have to create a new boot floppy. Now your network card should be recognized and the install kernel should mount the nfsroot successfully. More information how to compile an install kernel can be found in the `README` of the package `fai-kernels`.

[5] `/usr/lib/fai/nfsroot` from the install server

3.8 Collecting other system information

Now the clients have booted with *FAI_ACTION* set to *sysinfo*. Type `ctrl-c` to get a shell or use `Alt-F2` or `Alt-F3` and you will get another console terminal, if you have added `createvt` to *FAI_FLAGS*. Remote login is available via the secure shell, if `sshd` if added to *FAI_FLAGS*. The encrypted password is set with variable *FAI_ROOTPW* in `/etc/fai.conf` and is default to "fai". You can also log in without a password when using *SSH_IDENTITY*. To log in from your server to the install client (for eg. named ant01) use:

```
> ssh -l root ant01
Warning: Permanently added 'ant01,134.95.9.200' to the list of known hosts.
root@ant01's password:
```

You have now a running Linux system on the install client without using the local hard disk. Use this as a rescue system, if your local disk is damaged or the computer can't boot properly from hard disk. You will get a shell and can execute various commands (dmesg, lsmod, df, lspci, . . .). Look at the log file in `/tmp`. There you can find much information about the boot process. All log files from `/tmp` are also written to the install server into the directory `~fai/ant01/sysinfo/`[6]

A very nice feature is, that FAI mounts all filesystems it finds on the local disks read only. It also tells you on which partition a file `/etc/fstab` exists. When only one file is found, the partitions are mounted according to this information. Here's an example:

```
ant01:~# df
Filesystem   1k-blocks    Used Available Use% Mounted on
/dev/root      1249132  855648    330032  72% /
/dev/ram0         3963      36      3927   1% /tmp
kueppers:/usr/local/share/fai
               1249132  855648    330032  72% /fai
/dev/hda1        54447    9859     41777  19% /tmp/target
/dev/hda10     1153576      20   1141992   0% /tmp/target/files/install
/dev/hda9       711540      20    711520   0% /tmp/target/home
/dev/hda8       303336      13    300191   0% /tmp/target/tmp
/dev/hda7      1517948   98252   1342588   7% /tmp/target/usr
/dev/hda6       202225    8834    182949   5% /tmp/target/var
```

This method can be used as an rescue environment ! In future it will be possible to make backups or restore data to existing filesystems. If you need a filesystem with read–write access use the `rwmount` command:

```
ant01:~# rwmount /tmp/target/home
```

[6]More general: `~$LOGUSER/$HOSTNAME/$FAI_ACTION/`. Examples of log files can be found on the FAI homepage.

3.9 Checking parameters from BOOTP and DHCP servers

If the install client boots with action *sysinfo*, you can also check if all information from the BOOTP or DHCP daemons are received correctly. If the kernel uses BOOTP requests to receive data, these information will be written to `/tmp/bootp.log`. If the kernel support the DHCP protocol all is written to `/tmp/dhcp.log`. And example for the result of a BOOTP requests can be found in 'The setup routines of the install clients' on page 29.

3.10 Rebooting the computer

At any time you can reboot the computer using the command `faireboot`, also if logged in from remote. If the installation hasn't finished, use `faireboot -s`, so the log files are also copied to the install server.

Chapter 4

Overview of the installation sequence

Following steps are performed during an installation after the linux kernel has booted on the install clients.

1. Set up FAI
2. Load kernel modules
3. Define classes
4. Define variables
5. Partition local disks
6. Create and mount local filesystems
7. Install software packages
8. Call site specific configuration scripts
9. Save log files
10. Reboot the new installed system

You can also define additional programs or scripts, that will be run on particular occasions. They are called `hooks`. Hooks can add additional functions to the installation process or replace the default subtasks of FAI. So it's very easy to customize the whole installation process. Hooks are explained in detail in 'Hooks' on page 37.

The installation time is mainly determined by the amount of software that will be installed. A dual Pentium II, 400 MHZ with 128 MB RAM and a 10 Mbit network card needs about 11 min when installing 520 MB of software to local disk. An installation with 90 MB software (a dataless client) only needs two minutes. The checking for bad blocks on a 4 GB partition needs additional 6 minutes. Using a 100 Mbit LAN does not decrease the installation time considerable, so the network will not be the bottleneck when installing several clients simultaneously.

4.1 Set up FAI

After the install client has booted, only the script /sbin/rcS_fai[1] is executed. This is the main script, that controls the sequence of tasks for FAI. No other scripts in /etc/init.d/ are executed.

A ramdisk is created and mounted to /tmp, which is the only writable directory, until local filesystems are mounted. Additional parameters are received from the BOOTP or DHCP daemon and the configuration space if mounted via NFS from the install server to /fai. The setup is finished after additional virtual terminals are created and the secure shell daemon for remote access is started on demand.

4.2 Defining classes, variables and loading kernel modules

Now scripts in /fai/class/ are executed to define classes and variables that are later used in the site specific configuration scripts. All scripts matching S[0-9]*.{sh,pl} are executed in alphabetical order. Every word that these scripts print to the standard output are interpreted as class names. These classes are defined for this install client. You can also say this client belongs to these classes. A class is defined or undefined and has no value. Only defined classes are of interest for an install client. A description of all classes can be found in /usr/share/doc/fai/classes_description.txt. It is advisable to document the job a new class performs. Then, this documentation is the base for composing a configuration from classes.

The scripts S05modules.sh loads kernel modules on demand. So you can use classes when loading modules and also define more classes after the kernel has loaded modules and recognized new hardware. A complete description of all these scripts can be found in 'Scripts in /fai/scripts' on page 36.

After defining the classes, every file matching *.var, which prefix is a defined class, is executed to define variables.

4.3 Partitioning local disks, creating filesystems

Then exactly one disk configuration file from /fai/disk_config is selected using classes. It's the description how all the local disks will be partitioned, which filesystems should be created and how they are mounted. It's also possible to preserve the disk layout or to preserve the data on a partition. It's done by the command setup_harddisks, which uses sfdisk for partitioning. The format of the configuration file is described in /usr/share/doc/fai/README.disk_config.

During the installation process all local filesystems are mounted relative to /tmp/target . So, for e.g. /tmp/target/home will become /home in the new installed system.

[1] Since the root filesystem on the clients is mounted via NFS, rcS_fai is located in /usr/lib/fai/nfsroot/sbin on the install server.

4.4 Installing software packages

When local filesystems are created, they are all empty (except for preserved partitions). Now the Debian base system and all requested software packages are installed on the new filesystems. First the base archive is unpacked, then the command `install_packages` installs all packages using `apt-get(8)` without any manually interaction needed. If a packages requires an other packages it resolves this dependency by installing the required package.

Classes are also used when selecting the configuration files in `/fai/package_config/` for software installation. The format of the configuration files is described in 'Software package configuration' on page 34.

4.5 Site specific configuration

After all requested software packages are installed, the system is nearly ready to go. But not all default configurations of the software packages will meet your site specific needs. So you can call arbitrary scripts, which adjust the system configuration. Therefore scripts, that match a class name in `/fai/scripts`[2] will be executed. FAI comes with some templates for these scripts, but you can write your own Bourne, bash, perl, cfengine or expect scripts.

These important scripts are described in detail in 'Scripts in `/fai/scripts`' on page 36.

4.6 Save log files

When all installation tasks are finished, the log files are written to `/var/log/fai/$HOSTNAME/install/` [3] on the new system and to the a account on the install server, if *$LOGUSER* is defined in `/etc/fai.conf`.

4.7 Reboot the new installed system

At least the system is automatically rebooted if the flag *reboot* is set [4]. This is only useful if booting from network card or if you can change the boot device using the command `bootsector`. Otherwise, you have to remove the floppy disk and type return or call `faireboot` from a remote login. You must change the boot device, to boot the new installed system otherwise the installation would be performed again. Read 'Changing the boot device' on page 36 how to change the boot device.

[2] `/usr/local/share/fai/scripts` on the install server.
[3] `/var/log/fai/localhost/install/` is a link to this directory.
[4] Add "reboot" to *FAI_FLAGS* (= T172).

Chapter 5

How to plan your installation

Plan your installation, and FAI installs your plans.

Before starting your installation, you should spend much time in planing your installation. When you're happy with your installation concept, FAI can do all the boring, repeating tasks to put your plans into practice. FAI can't do good installations, if your concept is imperfect or lacks some important details. Start planing the installations by answering yourself following questions:

- Will I create a Beowulf cluster, or do I have multiple workstations, each only used by a single user ?
- How does my LAN topology looks like ?
- Which applications will be run ?
- Do the users need a queueing system ?
- Do I have uniform hardware ?
- Will the hardware stay uniform in the future ?
- How should the hosts be named ?
- How should the local hard disks be partitioned ?
- Which software should be installed ?
- Which daemons should be started, and how should the configuration for these looks like ?
- Which remote filesystems should be mounted ?
- Does the hardware needs a special kernel ?

You have also to think about user accounts, printing, mail, cron jobs, graphic cards, dual boot, NIS, NTP, timezone, keyboard layout, exporting and mounting directories via NFS and many other things. So, there's a lot to do before starting an installation. And remember that knowledge is power, and it's up to you to use it. Installation and administration is a process, not a product. FAI can't do things you don't tell it to do.

But you need not to start from scratch. Look at all files and scripts in the configuration space. There are a lot of things you can use for your own installation. A good paper with more aspects of building an infrastructure is `http://www.infrastructures.org/papers/bootstrap/` "Bootstrapping an Infrastructure".

Chapter 6

Installation details

6.1 The configuration space

The configuration is the collection of information how exactly to install a computer. The central configuration space for all install clients is located on the server in `/usr/local/share/fai` and its subdirectories. This will be mounted by the install clients to `/fai`. Following subdirectories are present and include several files:

class/ Scripts and files to define classes and variables and to load kernel modules.

disk_config/ Configuration files for disk partitioning and file system creation.

package_config/ File with lists of software packages to be installed or removed.

scripts/ Script for customization.

files/ Files used by customization scripts, eg. user created kernel packages. Most files are located in a subtree structure, which reflects the ordinary directory tree. For eg. the templates for `nsswitch.conf` are located in `/fai/files/etc/nsswich.conf`.

hooks/ Hooks are user defined programs or scripts, which are called during the installation process.

The main installation script `rcS_fai` uses all these subdirectories in the order listed. The FAI package contains templates for all these configuration scripts and files in `/usr/share/fai/templaes`. Copy the configuration templates to the configuration space and start an installation. These files need not belong to the root account. You can change their ownership and then edit the configuration with a normal user account.

```
# cp -a /usr/share/fai/templates/* /usr/local/share/fai
# chown -R fai /usr/local/share/fai
```

All these files contain configuration for some example hosts. A cluster of workstations (*bigfoot, ant01, ant02,...*) and a Beowulf cluster with a master node called *nucleus* and computing nodes *atom01, atom02,...* are included.

bigfoot This is a server with much software. It provides the home directory `/usr` for its NFS clients. Also some daemons are installed and activated by default.

ant01,... These dataless clients mount `/usr` and `/home` from *bigfoot*. Most of the disk space is spend for a scratch partition, which is exported to a netgroup of hosts.

nucleus This Beowulf master node is a server with much software. It provides the home directory `/usr/local` for its computing nodes. Also some daemons are installed and activated by default.

atom01,... These Beowulf clients mount `/usr/local` and `/home` from *nucleus*. Most of the disk space is spend for a scratch partition, which is exported to a netgroup of hosts.

Start looking at these examples and study them. Then change or add things to these examples. But don't forget to plan your own installation !

6.2 The default tasks

After the kernel has booted, it mounts the root file system via NFS from the install server and init starts the script `/sbin/rcS_fai`. This is the script which controls the sequence of the installation. No other scripts in `/etc/init.d` are used.

The installation script uses many subroutines, which are defined in `/usr/share/fai/subroutines`. All important tasks of the installation are called via the subroutine `task` *name* appended by the name of the task as an option (eg. `task instsoft`). The subroutine `task` calls hooks with prefix *name* if available and then calls the default task (defined as `task_`*name* in `subroutines`). The default task can be skipped on demand if the file `/tmp/skip.\textit{name}` exists.

This is the description of all default tasks.

confdir The kernel append parameters define variables, the syslog and kernel log daemon are started. The list of network devices is stored in *$netdevices*. Then additional parameters are fetched from a DHCP or BOOTP server and also define variables. The resolver configuration file is created and the configuration space is mounted from the install server to `/fai`. The file `/fai/hooks/subroutines` is sourced, if it exists. Using this file, you can define your own subroutines or override the definition of FAI's subroutines.

setup This task sets the system time, all *FAI_FLAGS* are defined and two additional virtual terminals are opened on demand. A secure shell daemon is started on demand for remote logins. A list of all local hard disks is stored to *$disklist* and *$device_size* contains a list of disk devices and their size.

defclass Defines classes using scripts and files in `/fai/class` and classes from `/tmp/additional-classes`.

defvar Sources all files `/fai/class/*.var` for every defined class. If a hook has written some variable definitions to the file `/tmp/additional.var`, this file is also sourced.

action Depending on the value of *$FAI_ACTION* this subroutine decides which action FAI should perform. The default actions are: `sysinfo, install, backup`. If *$FAI_ACTION* has another value, a user defined action is called, if a file `/fai/hooks/`*$FAI_ACTION* exists. So you can easily define your own actions.

sysinfo Called when no installation is performed but the action is `sysinfo`. It shows information about the detected hardware and mounts the local hard disks read only to `/tmp/target/`*partitionname* or with regard to a `fstab` file found inside a partition. Log files are stored to the install server.

backup Called when action is `backup`. Currently the default backup subroutine performs no action.

install This task controls the installation sequence. The major work is to call other tasks and to save the output to `/tmp/rcS.log`. If you have any problems during installation, look for errors there. You can find examples of the log files for some hosts in the download directory of the FAI homepage.

partition Calls `setup_harddisk` to partition the hard disks. The task writes variable definitions for the root and boot device (*$ROOT_PARTITION, $BOOT_PARTITION*) to `/tmp/disk_var.sh`.

mountdisks Mounts the created partitions according to the created `/tmp/fstab` file relative to *$FAI_ROOT*.

extrbase Extracts the base tar file `base.tgz`, which is created by `make-fai-nfsroot`.

mirror If a local mirror is accessed via NFS when *$FAI_DEBMIRROR* is defined, this directory will be mounted to *$MNTPOINT*, which is defined in `/etc/fai.conf`.

updatebase Prepares the extracted base system for further installation and updates the list of available packages. Updates the packages to the newest version. Creates fake scripts for some commands inside the new installed system using `dpkg-divert(8)`.

instsoft Installs the desired software packages using class files in `/fai/packages_config`.

configure Calls scripts in `/fai/scripts/` for every defined class.

finish Unmounts the proc filesystem in the new installed system and removes diversions of files (`rmdivert`).

faiend Automaticly reboots the install clients or waits for manual input before reboot.

chboot Changes symbolic link on the install server, which indicates which kernel image to load on the next boot from network card.

savelog Saves log files to local disk and to the install server.

6.3 The setup routines of the install clients

After the subroutine `fai_init` has done some basic initialization, the setup continues by calling the task `confdir` and the task `setup`. The command `bootpc(8)` is called and its output is used to define the corresponding global variables which are save in `/tmp/bootp.log`. This is an example for this log file.

```
/tmp/bootp.log:

# --- network device eth0 ---
SERVER='134.95.9.149'
IPADDR='134.95.9.200'
BOOTFILE='/boot/fai/bigfoot'
NETMASK='255.255.255.0'
NETWORK='134.95.9.0'
BROADCAST='134.95.9.255'
GATEWAYS_1='134.95.9.254'
GATEWAYS='134.95.9.254'
ROOT_PATH='/usr/lib/fai/nfsroot'
DNSSRVS_1='134.95.9.136'
DNSSRVS_3='134.95.100.208'
DNSSRVS_4='134.95.140.208'
DNSSRVS='134.95.9.136 134.95.100.208 134.95.140.208'
DOMAIN='informatik.uni-koeln.de'
SEARCH='informatik.uni-koeln.de uni-koeln.de'
YPSRVR_1='134.95.9.10'
YPSRVR='134.95.9.10'
YPDOMAIN='informatik4711.YP'
TIMESRVS_1='134.95.9.10'
TIMESRVS='134.95.9.10'
NTPSRVS_1='134.95.81.172'
NTPSRVS_2='134.95.140.172'
NTPSRVS='134.95.81.172 134.95.140.172'
HOSTNAME='bigfoot'
T170='kueppers:/usr/local/share/fai'
T171='sysinfo'
T171='install'
T172='createvt sshd'
```

It's not a bug, if a variable (`T171` in the example above) is defined twice. The second definition is created by the host specific entry in `/etc/bootptab` and supersedes the first one.

The tag `T172` is the definition for *$FAI_FLAGS*. It contains a space separated list of flags. Following flags are known:

verbose Create verbose output during installation. This should be always the first flag, so consecutive definitions of flags will be verbosely displayed.

debug Create debug output. No interactive installation is performed. During package installation you have to answer all questions of the postinstall scripts from the console.

sshd Start the ssh daemon to enable remote logins.

createvt Create two virtual terminals and execute a bash if `ctrl-c` is typed in the console terminal. The additional terminals can be accessed by typing `Alt-F2` or `Alt-F3`. Otherwise no terminals are available and typing `ctrl-c` will reboot the install client. Useful for installation which should not be interruptible.

reboot Reboot the install client after installation is finished without typing RETURN on the console. Only useful if you can change the boot image or boot device automatically or your assembly robot can remove the boot floppy via remote control. Currently this should only be used when booting from network card and using *$TFTPLINK* or changing the boot device with the command `bootsector`.

6.4 The class concept

The idea of using classes in general and using certain files matching a class name for a configuration is adopted from the installation scripts by Casper Dik for Solaris. This technique proved to be very useful for the SUN workstations, so I also use it for the fully automatic installation of Linux. One simple and very efficient feature of Casper's scripts is to call a command with all files, whose file names are also a class. The following loop may implement this function in a shell script:

```
for class in $all_classes; do
if [ -r $config_dir/$class ]; then
   command $config_dir/$class
   # exit, if only the first matching file is needed
fi
done
```

A variation would be to call the command only for the first file that matches a class name. Therefore it is possible to add a new file to the configuration without changing the script. This is because the loop automaticly detects new configurations files that should be used. Unfortunately cfengine does not support this nice feature, so all classes being used in cfengine need also to be specified inside the cfengine scripts. Classes are very important for the fully automatic installation. If a client belongs to class `A`, we say the class `A` is defined. A class has no value, it is just defined or undefined. Within scripts, the variable *$classes* holds a space separated list with the names of all defined classes. Classes determine how the installation is performed. For example, an install client is configured to become a FTP server by default. Mostly a configuration is created by only changing or appending the classes to which a client belongs, making the installation of a new client very easy. Thus no additional information needs to be added to the configuration files if the existing classes suffice your needs. There are different possibilities to define classes:

1. Some default classes are defined for every host: DEFAULT, LAST and its hostname.
2. Classes may listed within a file.
3. Classes may be defined by scripts.

The last option is a very nice feature, since these scripts will define classes automatically. For example, several classes are defined only if certain hardware is identified. We use Perl and shell scripts to define classes. All names of classes, except the hostname, are written in uppercase. They must not contain a hyphen, a hash or a dot, but may contain underscores. A description of all classes can be found in `/usr/share/doc/fai/classes_description.txt`.

Hostnames should rarely be used for the configuration files in the configuration space. Instead, a class should defined and this class is then added to the host.

6.5 Defining classes

The default task *defclass* defines the classes `DEFAULT, LAST`, *$HOSTNAME* and all classes in the file `/fai/class/`*$HOSTNAME* for every hosts. Additionally, all files that are executable and match the shell regular expression `S[0-9]*.{sh,pl}` are called in alphabetical order. Every output from these scripts is used to define classes. Multiple classes in one line must be space separated. If a hook has written classes to the file `/tmp/additional-classes`, these classes are also defined. The list of all defined classes is stored in the variable *$classes* and saved to `/tmp/FAI_CLASSES`. The list of all classes is transfered to `cfengine`, so it can use them too. Script `S01alias.sh` (see below) is used to define classes for groups of hosts. All hosts with prefix *ant* use all classes in the file `anthill`. Hosts, which have an IP address in subnet 134.95.9.0 also belongs to class `NET_9`. All Beowulf nodes with prefix *atom* except *atom00* (master server) will belong to the classes listed in file `atoms`. Finally this scripts defines the class with the name of the hardware architecture in uppercase letters.

```
S01alias.sh:

# all hosts named ant?? are using the classes in file anthill
case $HOSTNAME in
    ant??) cat anthill ;;
esac

# the Beowulf cluster; all nodes except the master node atom00
# use classes from file class/atoms
case $HOSTNAME in
    atom00) ;;
    atom??) cat atoms ;;
esac

# if host belongs to class C subnet 134.95.9.0 use class NET_9
case $IPADDR in
    134.95.9.*) echo NET_9 ;;
esac

# echo architecture in upper case
dpkg --print-installation-architecture | tr /a-z/ /A-Z/
```

Script `S07disk.pl` can be used to define classes depending on the number of local disks or the size of these scripts[1]. But you can also use a range of partition size in the disk configuration file (in `disk_config`), so you may not need a class for every different disk size. The script `S24nis.sh` automatically defines classes corresponding to NIS. The name of the NIS domain (defined via BOOTP or DHCP) will also become a class (only uppercase letters and minus is replaced by underscore). Depending on several partition names, `S90partitions.pl` defines classes. For eg., if a partition `/files/scratch` exists, the install client will export this directory via NFS therefore installs a NFS server packages.

The script `S05modules.sh` does not define any class, but is responsible for loading kernel modules. Kernel modules are important for detecting hardware. This script calls the script *$HOSTNAME*`.mod` and all scripts that have the format `<classname>.mod` and those class names are already defined. Classes, that are used for loading modules must be defined before this script is called. For e.g., if class `DEFAULT` is defined (this class is always defined) and a file `DEFAULT.mod` exists, this script is executed. These scripts should contain all command for loading kernel modules:

```
DEFAULT.mod:

modprobe parport_probe
modprobe serial
```

You can find messages from modprobe in `/tmp/dmesg.log` and the on the fourth console terminal by pressing `Alt-F4`.

6.6 Variables in `class/*.var`

The task *defvar* defines the variables for the install clients. All global variables can be set in `DEFAULT.var`. For certain groups of hosts use a class file or for a single host use the file *$HOSTNAME*`.var`. Also here, it's useful to study all the examples. Following variables are used in the examples and may be also useful for your installation:

hdparm Multi line commands to tune the hard disks parameters. They are executed during installation and also create the script `/etc/init.d/S61hdparm` which tunes the hard disks during boot.

UTC Set hardware clock to UTC if `$UTC=yes`. Otherwise set clock to local time. See `clock(8)` for more information.

time_zone Is the file relative to `/usr/share/zoneinfo/` which indicates your time zone.

FAI_CONSOLEFONT Is the font, that is loaded during installation by `consolechars(8)`.

FAI_KEYMAP Defines the keyboard map file in `/usr/share/keymaps`. You need not specify the complete path, since this file will be located automatically.

[1] It uses the library `Fai.pm`, which includes some useful subroutines, e.g. `class`, `classes`, `read_memory_info`, `read_ethernet_info`.

kernelimage The kernel that is installed to the new system. If a Debian package `/fai/files/packages/`*$kernelimage* exists, install this kernel package. Otherwise install the package *$kernelimage* from the Debian mirror. For eg., if `kernelimage=kernel-image-2.2.19-idepci` this kernel will be installed. To install the a special kernel for host bigfoot, set `kernelimage=kernel-image-2.2.19_bigfoot1_i386.deb` and this kernel will be installed from `/fai/files/packages/`.

liloappend Append parameters for the kernel of the new system (written to `/etc/lilo.conf`).

moduleslist Can be a multi line definition. List of modules (including kernel parameters), that are loaded during boot of the new system (written to /etc/modules).

rootpw The root password for the new system. Additionally, FAI creates an root account with the same password called `roott`, which uses the `tcsh(1)`.

TFTPLINK Link to the TFTP kernel image, that boots using the root file system from the local disk.

hserver, bserver The names of the NFS servers for `/home` and `/usr`.

printers List of printers, for which a spool directory is created. The config scripts does not set up `/etc/printcap`.

6.7 Hard disk configuration

The format of the hard disk configuration files is described in `/usr/share/doc/fai/README.disk_config.gz`. The config file `/fai/disk_config/CS_KOELN` is a generic description for one IDE hard disk, which should fit for most installations. If you can't partition your hard disk using this script [2], use a hook instead. The hook should write the new partition table, create the file systems and create the file `/tmp/fstab` and `/tmp/disk_var.sh`, which contains definitions of boot and root partitions.

6.8 Software package configuration

The script `install_packages` installs the selected software packages. It uses all configuration files in `/fai/package_config`, which file name is also defined as a class. The syntax is very simple.

```
# an example package class

PRELOADRM http://www.location.org/rp8_linux20_libc6_i386_cs1_rpm /root

PACKAGES taskinst
german science german
```

[2] Currently this script uses the command `sfdisk(8)`, which isn't available on SUN SPARC.

```
PACKAGES install
adduser netstd ae
less passwd
realplayer

PACKAGES remove
gpm xdm

PACKAGES dselect-upgrade
ddd                     install
a2ps                    install
```

Comments are starting with a hash (#) and are ending at the end of the line. Every command begins with the word `PACKAGES` followed by a command name. The command name is similar to those of `apt-get`. Here's the list of supported command names:

hold: Put a package on hold. This packages will not be handled by dpkg, e.g not upgraded.

install: Install all packages that are specified in the following lines. If a hyphen is appended to the package name (with no intervening space), the package will be removed, not installed. All package names are checked for misspelling. If a package doesn't exist, its automaticly removed from the list of packages to install.

remove: Remove all packages that are specified in the following lines. Append a + to the package name, if the package should be installed.

taskinst: Install all packages belonging to the task that are specified in the following lines using `tasksel(1)`. This works only for Debian 3.0 and later.

dselect–upgrade Set package selections using the following lines and install or remove the packages specified. These lines must be the output of the commands `dpkg --get-selections`.

Multiple lines with lists of space separated names of packages follows the commands install and remove. All dependencies are resolved and `apt-get` is used to perform the installation or removal of packages. The order of the packages is of no matter.

A line, that contains the `PRELOADRM` commands, downloads a file using `wget(1)` into a directory before installing the packages. Using the `file:` URL, this file is copied from *$FAI_ROOT* to the download directory. For examples the package `realplayer` needs an archive to install the software, so this archive is downloaded to the directory `/root`. After installing the packages this file will be removed. If the file shouldn't be removed, use the the command `PRELOAD` instead.

If you specify a package that does not exists (e.g. you make a typo), the the whole installation of software package will not be started. This could also happen when the command `xviddetect(1)` does not recognize the video card, because the configuration file `SERVER` contains following line:

```
xserver-`xviddetect -q`
```

If the video card isn't detected, the software installation tries to install the package `xserver-unknown`. It will not start because this package doesn't exist. You can test all software package configuration files with the utility `chkdebnames`, which is available in `/usr/share/fai/utils/`.

```
> chkdebnames stable /usr/local/share/fai/package_config/*
```

6.9 Scripts in `/fai/scripts`

The default set of scripts in this directory is only an example. But they should do a reasonable job for your installation. You can edit them or add new scripts to match your local needs.

6.9.1 Shell scripts

Most script are Bourne shell script. Shell scripts are useful, if the configuration task only needs to call some shell commands or create a file from scratch. In order not to write much short script, it's possible to distinguish classes within a script using the command `ifclass`. For copying files with classes, use the command `fcopy(8)`. If you like to extract an archive using classes, use `ftar(8)`. But now have a look at the scripts and see what they are doing.

6.9.2 Perl scripts

Currently no Perl scripts are used for modifying the system configuration.

6.9.3 Expect scripts

Currently no expect scripts are used for modifying the system configuration.

6.9.4 Cfengine scripts

Cfengine has a rich set of functions to edit existing configuration files, e.g `LocateLineMatching, ReplaceAll, InsertLine, AppendIfNoSuchLine, HashCommentLinesContaining`. But it can't handle variables, that are undefined. If a variables is undefined, the whole cfengine script will abort. More information can be found in the manual page `cfengine(8)` or at the cfengine homepage `http://www.cfengine.org`.

6.10 Changing the boot device

Changing the boot sequence is normally done in the BIOS setup. But you can't change the BIOS from a running Linux system as far as I know. If you know how to perform this, please send me an email. But there's another way of swapping the boot device of a running Linux system.

Change the boot sequence in the BIOS, so the first boot device is the local disk, where the master boot record is located. The second boot device should be set to LAN or floppy disk, depending from which media you boot when the installation process is performed.

After the installation is performed, `lilo(8)` will write a valid boot sector to the local disk. Since it's the first boot device, the computer will boot the new installed system. If you like to perform an installation again, you have to disable this boot sector using the command `bootsector`[3] . For more information use:

```
# bootsector -h
```

This is how to set up the a 3Com network card as second boot device, even if the BIOS doesn't support this. Enable LAN as first boot device in the BIOS.

```
Boot From LAN First: Enabled
Boot Sequence      : C only
```

Then enter the MBA setup of the 3Com network card and change it as follows:

```
Default Boot           Local
Local Boot             Enabled
Message Timeout        3 Seconds
Boot Failure Prompt    Wait for timeout
Boot Failure           Next boot device
```

This will enable the first IDE hard disk as first boot device. If the boot sector of the hard disk is disabled, the computer will use the network interface as second boot device and boots from it. Maybe the disk partitioning tool can't work on such a disk. So you have to enable the boot sector before you want to partition the disk.

6.11 Hooks

Hooks let you specify functions or programs, that are run at certain steps of the installation process. Before a default task is called, FAI search for existing hooks for this task and executes them. As you might expect, classes are also used when calling hooks. Hooks are executed for every defined class. You only have to create the hook with the name for the desired class and it will be used. If `debug` is included in *$FAI_FLAG* the option `-d` is passed to all hooks, so you can debug your own hooks. If the default task should be skipped, use the subroutine `skiptask` and a list of default tasks as parameters. The example `partition.DISKLESS` skips some default tasks.

The directory `/fai/hooks/` contains all hooks. The file name of a hook consists of a hook name as a prefix and a class name, chained by a dot. The prefix describes the time when the hook is called, if

[3]The command `bootsector` is part of the package `fai` and will be installed to `/usr/local/sbin` on the install clients.

the class if defined for the install client. For example, the hook `partition.DISKLESS` is called for every client belonging to the class `DISKLESS` before the local disks would be partitioned. If it should become a diskless client, this hook can mount remote filesystems via NFS and create a `/tmp/fstab`. After that, the installation process would not try to partition and format a local hard disk, because a file `/tmp/fstab` already exists.

A hook of the form `hookprefix.classname` can't define variables for the installation script, because it's a subprocess. But you can use any binary executable or any script you wrote. Hooks that has the suffix `.sh` (eg. `partition.DEFAULT.sh`) must be Bourne shell scripts and are sourced. So it's possible to redefine variables for the installation scripts.

After some basic initialization, all hooks with prefix `confdir` are called. Since the configuration directory `/fai` is mounted in the default task `confdir`, the hooks for this task are the only hooks located in *$nfsroot*`/fai/hooks` on the install server. All other hooks are found in `/usr/local/share/fai/hooks` on the install server. All hooks that are called before classes are defined can only use the following classes: `DEFAULT $HOSTNAME LAST`. If a hook for class `DEFAULT` should only be called if no hook for class *$HOSTNAME* is available, insert these lines to the default hook:

```
hookexample.DEFAULT:

#! /bin/sh

# skip DEFAULT hook, if a hook for $HOSTNAME exists
scriptname=$(basename $0 .DEFAULT)
[-f /fai/hooks/$scriptname.$HOSTNAME ] && exit
# here follows the actions for class DEFAULT
.
.
```

Some examples for what hooks could be used:

- Use `ssh` in the very beginning to verify that you mounted the configuration from the correct server and not a possible spoofing host.
- Pop up a little menu and ask the user, which kind of installation should be performed (eg. CAD workstation, notebook, scientific workstation, workgroup server, Gnome desktop…). See `hooks/install.MENU` for an example how to use this. Keep in mind that this won't lead to a fully automatic installation ;–)
- Do not mount the configuration directory, instead get a compressed archive via HTTP or from floppy disk and extract it into a new ram disk, then redefine *$FAI_LOCATION*.
- Load kernel modules before classes are defined in `/fai/class`.
- Send an email to the administrator, if the installation is finished.
- Install a diskless client and skip local disk partitioning. See `hooks/partition.DISKLESS`.

6.12 Looking for errors

If the client can't successfully boot from the network card, use `tcpdump(8)` to look for Ethernet packets between the install server and the client. Search also for entries in several log files made by `in.tftpd(8)` and `bootpd(8)`:

```
egrep "tftpd|bootpd" /var/log/*
```

If the installation process stops or even when it finishes, you can parse all log files for errors using:

```
# egrep "no such variable|bad variable|E:|ERROR" *.log
```

Sometimes the installation seems to stop, but there's only a postinstall script of a software package that requires manual input from the console. Change to another virtual terminal and look which process is running with tools like `top(1)` and `pstree(1)`. You can add `debug` to `FAI_FLAGS`, so the installation process will show all output from the postinst scripts on the console and get its input also from console. Don't hesitate to send an email to the mailing list or to `<fai@informatik.uni-koeln.de>` if you have any questions. Sample log files from successful installed computers are available on the FAI homepage.

Chapter 7

How to build a Beowulf cluster using FAI

This chapter describes the particularities about building a Beowulf cluster using Debian GNU/Linux and FAI. For more information about the Beowulf concept look at `http://www.beowulf.org`.

7.1 Planing the Beowulf setup

The example of a Beowulf cluster consists of one master node and 25 clients. A big rack was assembled where all the cases were put into. A keyboard and a monitor were also put into the rack, which are most of the time connected to the master server. But since we have a very long cables for monitor and keyboard, they can also be connected to all nodes, if something has to be changed in BIOS or when looking for errors, when a node does not boot. Power supply is another topic you have to think about. Don't connect many nodes to one power cord and one outlet. Distribute them among several breakout boxes and outlets. And what about the heat emission ? A dozen nodes in a small room can create to much heat, so you will need an air condition. Will the power supplies of each node go to stand by mode or are all nodes are turned on simultaneously after a power failure ?

All computers are connected to a Fast Ethernet switch. The master node (or master server) is called *nucleus*. It has two network cards. One for the connection to the external Internet, one for the connection to the internal cluster network. If connected from the external Internet, it's called *nucleus*, but the cluster nodes accesses the master node with name *atom00*, which is a name for the second network interface. The master server is also the install server for the computing nodes. A local Debian mirror will be installed on the local harddisk. The home directories of all user accounts is also located on the master server. It will be exported via NFS to all computing nodes. NIS will be used to distribute account, host and printer information to all nodes.

All client nodes *atom01* to *atom25* are connected via the switch with the second interface card of the master node. They can only connect to the other nodes or the master, but can't communicate to any host outside their cluster network. So, all services (NTP, DNS, NIS, ...) must be available on the master server. I choose the class C network address *192.168.42.0* for building the local Beowulf cluster network. You can replace the subnet 42 with any other number you like. I you have more that 253 computing nodes, choose a class A network address (10.X.X.X).

In the phase of preparing the installation, you have to boot the first install client many time, until there's no fault in your configuration scripts. Therefore you should have physical access to the master server and one client node. If you have few space, connect both computers to a switch box, so one keyboard and monitor can be shared among both.

7.2 Set up the master server

The master server will be installed by hand, cause it could be your first computer installed with Debian. It you have already a Debian host running, you can also install it via FAI. Create a partition on `/files/scratch` for the local Debian mirror with more that 3GB space available.

7.2.1 Set up the network

Add following lines for the second network card to `/etc/network/interfaces`:

```
# Beowulf cluster connection
iface eth1 inet static
address 192.168.42.250
netmask 255.255.255.0
broadcast 192.168.42.255
```

Add the IP addresses for the client nodes. The FAI package has an example for this `/etc/hosts`:

```
# Beowulf nodes
# atom00 is the master server
192.168.42.250 atom00
192.168.42.1 atom01
192.168.42.2 atom02
```

You can give the internal Beowulf network a name when you add this line to `/etc/networks`:

```
beowcluster 192.168.42.0
```

Activate the second network interface with: `/etc/init.d/networking start`.

7.2.2 Setting up NIS

Add a normal user account *tom* which is the person how edits the configuration space and manages the local Debian mirror:

```
# adduser tom
# addgroup linuxadmin
```

This user should also be in the group *linuxadmin*. So, add a line to `/etc/group`:

```
linuxadmin:x:101:tom
```

To initialize the master server as NIS server call `ypinit -m`. Then, copy the file `netgroup` from the examples directory to `/etc` and edit other files there. Adjust access to the NIS service.

```
# cat /etc/ypserv.securenets
# Always allow access for localhost
255.0.0.0       127.0.0.0
# This line gives access to the Beowulf cluster
255.255.255.0 192.168.42.0
```

Rebuild the NIS maps: `# cd /var/yp; make`

7.2.3 Create a local Debian mirror

Now user *tom* can create a local Debian mirror on `/files/scratch/` using `mkdebmirror`. This will need about 2.6 GB disk space for Debian 2.2 (aka potato). Export this directory to the netgroup `@faiclients` read only.

7.2.4 Install FAI package on the master server

Add following packages to the install server:

```
     nucleus:/# apt-get install task-dns-server ntp tftp bootp nfs-kernel-
server fai fai-kernels
```

Configure NTP so that the master server and all client nodes will have the same correct system time.

It's very important to use the internal network name *atom00* for the master sever (not the external name *nucleus*) in `/etc/bootptab` and `/etc/fai.conf`. Replace the strings FAISERVER with `atom00` in `/etc/bootptab` and uncomment the following line in `/etc/fai.conf` so the Beowulf nodes can use the name for connecting their master server.

```
NFSROOT_ETC_HOSTS="192.168.42.250 atom00"
```

`/etc/bootptab:`

```
.
.
.failocal:\
```

```
            :tc=.faiglobal:\
            :sa=atom00:\
            :ts=atom00:\
            :T170="atom00:/usr/local/share/fai":\
            :T171="sysinfo":\
            :T172="verbose createvt sshd":\
            :sm=255.255.255.0:\
            :gw=192.168.42.250:\
            :dn=beowulf.debian.org:\
            :ds=192.168.42.250:\
            :ys=atom00:yd=nisnucleus:\
            :nt=atom00:
.
.
```

7.2.5 Prepare network booting

Uncomment the following line in `/etc/inetd.conf`:

```
#bootps dgram udp wait root /usr/sbin/bootpd bootpd -i -t 120
```

and restart the inetd daemon. The user *tom* should have permission to create and the symlinks for booting via network card, so change the group and add some utilities.

```
# chgrp -R linuxadmin /boot/fai; chmod -R g+rwx /boot/fai
# cp /usr/share/fai/utils/* /usr/local/bin
```

Now, the user *tom* can create a symlink in `/boot/fai` using

```
>tlink atom_install atom01
```

to boot the first client node for the first time. Then start to adjust the configuration for your client nodes. Don't forget to build the kernel for the cluster nodes using `make-kpkg(8)`.

7.3 Tools for Beowulf cluster

Following tools for a Beowulf cluster now are available in `/usr/local/bin`:

tlink Change the symbolic link that point to the kernel image for booting from network card.

all_hosts Print a list of all hosts, print only the hosts which respond to a ping or the hosts which do not respond. The complete list of hosts is defined by the netgroup `allhosts`. Look at `/usr/share/doc/fai/examples/etc/netgroup` for an example.

rshall Execute a command on all hosts which are up via rsh. Uses `all_hosts` to get the list of all hosts up. You can also use the `dsh(1)` command (dancer's shell, or distributed shell).

7.4 Wake on LAN with 3Com network cards

Wake on LAN is a very nice feature to power on a computer without having physical access to it. By sending a special ethernet paket to the network card, the computer will be turned on. Following things have to be done, to use the wake on LAN (WOL) feature.

1. Connect the network card to the Wake–On–LAN connector on the motherboard using a 3 pin cable.
2. My ASUS K7M motherboard has a jumper called `Vaux (3VSBSLT)` which allows to select the voltage supplied to add–in PCI cards. Set it to `Add 3VSB` (3 Volt stand by).
3. Turn on the wake on LAN feature in BIOS
4. For a 2.2 kernel you have to use the following driver: `http://www.uow.edu.au/~andrewm/linux/#3c59x-bc`

There's a little problem to enable the wake on LAN feature with a 2.2.19 kernel and a 3Com 3C905C network card. You have to use a patched 3c59x driver. But I managed it to get work. Download the file `poll-ioctl-2.2.18-pre16.c.gz` and copy it to kernel sources to `drivers/net/3c59x.c`. Then make a new kernel package and install this new kernel. To wake up a computer use the tool etherwake (in woody) or get the single C source file. For more information look at `http://www.scyld.com/expert/wake-on-lan.html`.

Chapter 8

FAI on SUN SPARC

Even FAI is architecture independent, there are some packages, that are only available for certain architectures (eg. silo, sparc–utils). SUN SPARC computer can boot from their boot prompt and don't need a boot floppy. To boot a SUN use:

```
boot net
```

You have to convert the kernel image from ELF format to a.out format. Therefore use the program `elftoaout` (mentioned in the FAQ). The symlink to the kernel image to be booted is not the host name. Look at the FAQ at `http://www.ultralinux.org` for more information. A success report is available at `http://www.opossum.ch/fai/`.

Chapter 9

Various Hints

This chapter has various hints which may not be explained in great detail.

Use `mkdivert`[1] if a postinst script calls a configuration program eg. the postinst script for package apache calls apacheconfig needs manual input. You can fake the configuration programm so the installation can be fully automatic. But don't forget to use `rmdivert` to remove this faked script.

During the installation you can execute commands inside the newly installed system in a chroot environment by using `chroot /tmp/target` or just `$ROOTCMD` followed by the command you want to call for eg. `$ROOTCMD dpkg -l` show the packages installed to the new system.

The only task, which has to be done manually for new hardware, is to assign the MAC address to a hostname and to an IP address and define classes for this host.

There's a tradeoff between writing a few large configuration scripts, or many short scripts, one for each class. Large scripts can distinguish classes by using case statements or the `ifclass` test.

If your computer can't boot from network card you need not always boot from floppy. Define a partition `/fai-boot` in your `disk_config` configuration file. Then the class `FAI_BOOTPART` will automaticly defined and will create a lilo entry for booting the FAI bootfloppy from this partition. So you can start the reinstallation without a boot floppy. This will also make the test phase shorter, since booting from hard disk is much faster than booting from floppy.

[1] `mkdivert` and `rmdivert` are subroutines defined in `subroutines`.