FAI Guide (Fully Automatic Installation)

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

FAI Guide version 2.5.4, 20 april 2006 for FAI package version 2.10.1

Abstract

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

Copyright Notice

Contents

Chapter 1

Introduction

1.1 Availability

The homepage of FAI is http://www.informatik.uni-koeln.de/fai. There you will find information about FAI, for example the mailing list archive. The FAI package is also available as a Debian package from http://www.informatik.uni-koeln.de/fai/download. It's an official Debian package and is available from all Debian mirrors. To access the newest versions of the FAI packages, you can add the following line to your /etc/apt/sources.list file.

```
deb http://www.informatik.uni-koeln.de/fai/download sarge koeln
```

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 subversion repository containing the newest developer version of FAI from a Bourne shell using the following commands.

```
> svn co svn://svn.debian.org/svn/fai/trunk fai
```

You can also use the web interface for the subversion repository at: svn.debian.org/wsvn/fai/ (and fai-kernels).

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

1.2 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 installations 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 planning 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 installations of Solaris operating systems 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 re-installation of existing workstations. For example, I had to build a temporary LAN with four SUN workstations for a conference, which lasted 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 workstations 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 used 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.

1.3 Overview and concepts

FAI is a non-interactive system to install a Debian GNU/Linux operating system unattended on a single computer or a whole cluster. You can take one or more virgin PCs, 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 who 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. 'Plan your installation, and FAI installs your plans' on page 29 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 and data for all install clients. In the examples of this manual this host is called `kueppers`.

[1] Solaris 8 Advanced Installation Guide at `docs.sun.com`

[2] `ftp://ftp.wins.uva.nl:/pub/solaris/auto-install/`

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 `demohost, nucleus, atom01, atom02,...`

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

- Hard disk layout
- Local filesystems, their types, mount points and mount options
- Software packages
- Keyboard layout, time zone, NIS, XFree86 configuration, remote filesystems, user accounts, printers ...

nfsroot : A (chroot) 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.

1.4 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 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 a 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.

1.5 Features

- A fully automated installation can be performed.
- Very quick unattended installation
- Update of running systems without reinstallation

- Hosts can boot from floppy, network card or CD.
- Easy creation of the boot floppy or CD
- DHCP and BOOTP protocol and PXE boot method are supported.
- No initial ramdisk is needed, runs even on a 486 CPU.
- Remote login via ssh during installation process possible.
- Two additional virtual terminals available during installation
- All similar configurations are shared among all install clients.
- Log files for all installations are saved 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
- Keyboard layout selectable
- Can be used as a rescue system.
- Tested on SUN SPARC hardware running Linux or Solaris.
- 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, ext3 and XFS filesystem support
- Automatic hardware detection

1.6 Quickstart - For the impatient user

So, you do not like to read the whole manual? You like to try an installation without reading the manual? OK. Here's how to succeed in a few minutes.

- Install fai and all recommended packages (see 'Setting up FAI' on page 8 on your install server).
- Edit /etc/fai.conf,run fai-setup -v and read its output.

- Install the simple examples into the configuration space:

  ```
  cp -a /usr/share/doc/fai-doc/examples/simple/*
  /usr/local/share/fai/
  ```

- Get the MAC address of your demo host.
- Add your host (try to name it `demohost`) to `dhcpd.conf` and `/etc/hosts` (=your DNS) on the FAI server.
- When using PXE, tell the install client to boot the install kernel and perform an installation during the next boot

  ```
  fai-chboot -IFv demohost
  ```

- If you want to try FAI without setting up a PXE+DNS+DHCP-Environment: put the hostnames into `/etc/hosts` inside the nfsroot and use a bootfloppy/CD/DVD to boot the client. See `make-fai-bootfloppy(8)`
- Boot your demo host and enjoy the fully automatic installation.
- If the installation has finished successfully, the computer should boot a small Debian system. You can login as user `demo` or `root` with password `fai`.

But now don't forget to read chapters 'Plan your installation, and FAI installs your plans' on page 29, 'Overview of the installation sequence' on page 25 and 'Installation details' on page 31 !

Chapter 2

Installing FAI

2.1 Requirements

The following items are required for an installation via FAI.

A computer: The computer must have a network interface card. Unless a diskless installation should be performed a local hard disk is also needed. No floppy disk, CD-ROM, keyboard or graphic card is needed.

DHCP or BOOTP server: The clients need one of these daemons to obtain boot information. But you can also put all this information onto the boot floppy.

TFTP server: The TFTP daemon is used for transferring the kernel to the clients. It's only needed when booting from network card with a boot PROM.

Client root: It is a mountable directory which contains the whole filesystem for the install clients during installation. It will be created during the 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 or an `apt-proxy(8)` is recommended if you install several computers.

Install kernel: A kernel image that supports the network card and mounts its root filesystem via NFS. The Debian package `fai-kernels` provides a default kernel for fai.

Configuration space: This directory tree which contains the configuration data is mounted via NFS by default. But you can also get this directory from a revision control system like CVS.

The TFTP daemon and an NFS server will be enabled automatically when installing the FAI package. All clients must have a network card which is recognized by the install kernel.

2.2 How to create a local Debian mirror

The script `mkdebmirror` [1] can be used for creating your own local Debian mirror. This script uses the script `debmirror(1)` and `rsync(1)`. A partial Debian mirror only for i386 architecture for Debian 3.1 (aka sarge) without the source packages needs about 9.0GB of disk space. Accessing the mirror via NFS will be the normal and fastest way in most cases. To see more output from the script call `mkdebmirror --debug`. A root account is not necessary to create and maintain the Debian mirror.

You can use the command `fai-mirror(1)` for creating a partial mirror, that only contains the software packages that are used in the classes in your configuration space. A partitial mirror containing all package for the simplex examples from the package fai-doc will only need about 300MB of disk space. To use HTTP access to the local Debian mirror, install a web server and create a symlink to the local directory where your mirror is located:

```
# apt-get install apache2
# ln -s /files/scratch/debmirror /var/www/debmirror
```

Create a file `sources.list(5)` in `/etc/fai` which gives access to your Debian mirror. An example can be found in `/usr/share/doc/fai/examples/etc`. Also add the IP-address of the HTTP server to the variable *NFSROOT_ETC_HOSTS* in `/etc/fai/make-fai-nfsroot.conf` when the install clients have no DNS access.

2.3 Setting up FAI

To setup a FAI install server you need at least the packages `fai-server`, `fai-doc`, `fai-kernels`. The package `fai-quickstart` contains dependencies on all required packages for an install server. Do not install the package `fai-nfsroot` on a normals system. This package can only be installed inside the nfsroot. If you would like to install all packages that are useful for a fai install server, use the following command

```
# aptitude install fai-quickstart
Reading Package Lists... Done
Building Dependency Tree
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be automatically installed:
  dhcp3-common dhcp3-server fai-client fai-doc fai-kernels fai-server
  tftp-hpa tftpd-hpa
The following packages have been kept back:
  kernel-headers-2.6-686
```

[1] You can find the script in `/usr/share/doc/fai/examples/utils/`.

```
The following NEW packages will be installed:
  dhcp3-common dhcp3-server fai-client fai-doc fai-kernels fai-quickstart
  fai-server tftp-hpa tftpd-hpa
0 packages upgraded, 9 newly installed, 0 to remove and 1 not upgraded.
Need to get 13.0MB of archives. After unpacking 17.9MB will be used.
Do you want to continue? [Y/n/?]
```

The suggested packages for FAI are: `ssh`, `debmirror`, `mknbi`, `apt-move`, `mkinitrd-cd`, `grub`.

The configuration for the FAI package (not the configuration data for the install clients) is defined in `/etc/fai/fai.conf`. Definitions that are only used for creating the nfsroot are located in `/etc/fai/make-fai-nfsroot.conf`. Edit these files before calling `fai-setup`. These are important variables in `/etc/fai/make-fai-nfsroot.conf`:

FAI_DEBOOTSTRAP For building the nfsroot there's the command called `debootstrap(8)`. It needs the location of a Debian mirror and the name of the distribution (woody, sarge, sid) for which the basic Debian system should be built.

NFSROOT_ETC_HOSTS If you use HTTP or FTP access to the Debain mirror, add its IP-address and the name to this variable. For a Beowulf master node, add the name and IP-address of both networks to it. This variable is not needed when the clients have access to a DNS server.

FAI_SOURCES_LIST Now OPSOLETE and unsupported. Use the file `/etc/fai/sources.list` instead.

KERNELPACKAGE You must specify the software package - built with `make-kpkg(8)` - which includes the default kernel for booting the install clients. The Debian package `fai-kernels` contains the default install kernels which supports both the DHCP and BOOTP protocols.

NFSROOT_PACKAGES This variable contains a list of additional software packages which will be added to the nfsroot.

FAI_BOOT which of DHCP and/or BOOTP the server should create setups for (when make-fai-nfsroot is run). The default is to create the setup for both protocols.

These are important variables in `/etc/fai/fai.conf`:

FAI_LOCATION This is the host name and the remote directory of the configuration space, which will be mounted via NFS. Its default value is `/usr/local/share/fai` but some like to use `/home/fai/config` or `/var/fai/config`. Remember that this directory must be exported to all install clients, so that all files can be read by root.

FAI_DEBMIRROR If you have NFS access to your local Debian mirror, specify the remote filesystem. It will be mounted to *$MNTPOINT*, which must also be defined. It's not needed if you use access via FTP or HTTP.

The content of `/etc/fai/sources.list` and *FAI_DEBMIRROR* are used by the install server and also by the clients. If your install server has multiple network cards and different hostnames 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`. It needs about 230MB of free disk space. Before setting up FAI, you should get the program `imggen`,[2] 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/fai.conf` and `/etc/fai/make-fai-nfsroot.conf` call `fai-setup`.

```
faiserver[~]# fai-setup
Creating FAI nfsroot can take a long time and will
need more than 230MB disk space in /usr/lib/fai/nfsroot.
Creating nfsroot for sarge using debootstrap
dpkg: base-passwd: dependency problems, but configuring anyway as you request
 base-passwd depends on libc6 (>= 2.3.2.ds1-4); however:
  Package libc6 is not installed.
dpkg: base-files: dependency problems, but configuring anyway as you request:
.
.
.
Creating base.tgz
‘/etc/fai/sources.list’ -> ‘/usr/lib/fai/nfsroot/etc/apt/sources.list’
Upgrading /usr/lib/fai/nfsroot
Adding additional packages to /usr/lib/fai/nfsroot:
fai-nfsroot module-init-tools dhcp3-client ssh rdate lshw hwinfo portmap
bootpc rsync wget rsh-client less dump reiserfsprogs usbutils
psmisc pciutils hdparm smartmontools parted mdadm lvm2
dnsutils ntpdate dosfstools cvs jove xfsprogs xfsdump
sysutils dialog discover mdetect libnet-perl netcat libapt-pkg-perl
grub lilo dmidecode hwtools read-edid
Backing up any LVM2 metadata that may exist...done.
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Restarting OpenBSD Secure Shell server: sshd.
Checking available versions of rmt, updating links in /etc/alternatives ...
(You may modify the symlinks there yourself if desired - see ‘man ln’.)
Updating rmt (/usr/sbin/rmt) to point to /usr/sbin/rmt-dump.
Updating rmt.8.gz (/usr/share/man/man8/rmt.8.gz) to point to /usr/share/man/m
Recovering jove files ... Done.
Error : Temporary failure in name resolution
‘/etc/fai/fai.conf’ -> ‘/usr/lib/fai/nfsroot/etc/fai/fai.conf’
```

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

```
‘/etc/fai/make-fai-nfsroot.conf’ -> ‘/usr/lib/fai/nfsroot/etc/fai/make-fai-nf
‘/etc/fai/menu.lst’ -> ‘/usr/lib/fai/nfsroot/etc/fai/menu.lst’
‘/etc/fai/sources.list’ -> ‘/usr/lib/fai/nfsroot/etc/fai/sources.list’
Shadow passwords are now on.
Kernel 2.6.16-fai-kernels installed into the nfsroot.
DHCP environment prepared. If you want to use it, you have to enable the dhcp

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 succesfully added
BOOTP environment prepared.
Removing ‘diversion of /sbin/discover-modprobe to /sbin/discover-modprobe.dis
make-fai-nfsroot finished properly.     <= *
Adding line to /etc/exports: /usr/local/share/fai 123.45.6.0/24(async,ro)
Adding line to /etc/exports: /usr/lib/fai/nfsroot 123.45.6.0/24(async,ro,no_r
Re-exporting directories for NFS kernel daemon...done.
You have no FAI configuration. Copy the simple examples with:
cp -a /usr/share/doc/fai/examples/simple/* /usr/local/share/fai
Then change the configuration files to meet your local needs.
FAI setup finished.                     <= *
```

A complete log of fai-setup is available on the fai web page. It's important that you will see both lines that are marked with an asterisk. Otherwise something went wrong. If you'll get a lot of blank lines, it's likely that you are using `konsole`, the X terminal emulation for KDE which has a bug. Try again using `xterm`.

The warning messages from dpkg about dependency problems can be ignored. If you have problems running fai-setup, they usually stem from `make-fai-nfsroot(8)`. You may restart it by calling 'make-fai-nfsroot -r' (recover). Adding '-v' gives you a more verbose output which may help you pinpoint the error. If you want to create a log file you may use

```
sudo /usr/sbin/make-fai-nfsroot -r -v 2>&1 | tee make-fai-nfsroot.log
```

It may help to enter the chroot environment manually

```
sudo chroot /usr/lib/fai/nfsroot
```

The setup routine adds some 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-user-server reload
```

The setup also creates the account `fai` (defined by *$LOGUSER*) if not already available. So you can add a user before calling `fai-setup(8)` using the command `adduser(8)` and use this as your local account for saving log files. 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 the symbolic links to the kernel image which is booted by a client.

After that, FAI is installed successfully on your server, but has no configuration for the install clients. Start with the examples from `/usr/share/doc/fai-doc/examples/simple/` using the copy command above and read 'Installation details' on page 31. 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 14.

When you make changes to `/etc/fai/fai.conf`, `/etc/fai/make-fai-nfsroot.conf` the nfsroot has to be rebuilt by calling `make-fai-nfsroot(8)`. If you only like to install a new kernel to the nfsroot add the flags `-k` or `-K` to `make-fai-nfsroot`. This will not recreate your nfsroot, but only updates your kernel and kernel modules inside the nfsroot

2.3.1 Troubleshooting the setup

The setup of FAI adds the FAI account, exports filesystems and calls `make-fai-nfsroot(8)`. 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 mount 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 Technologies 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 should 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:                PXE

Default Boot:               Network
Local Boot:                 Enabled
Config Message:             Enabled
Message Timeout:            3 Seconds
Boot Failure Prompt:        Wait for timeout
================================================================================
  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 PXE and enable local boot in this menu. So the first boot device will be the network card using PXE, and the second should be the local hard disk. This has to

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

be configured in the BIOS of your computer. If you like to use the BOOTP protocol choose `TCP/IP` and set the protocol to `BOOTP`. When using BOOTP, you have to make a symbolic link from the hostname of your client to the appropriate kernel image in `/boot/fai`. You can also use the utility `tlink` (`/usr/share/doc/fai/examples/utils/tlink`) to create this link. The file `installimage_3com` is created by `imggen` and is suitable for booting 3Com network cards[2].

3.2 Booting from network card with a PXE conforming boot ROM

Most modern bootable network cards support the PXE boot environment. Some network cards (e.g. Intel EtherExpress PRO 100) have a fixed boot configuration, so they can only use the PXE boot protocol. This requires a PXE Linux boot loader and a special version of the `TFTP` daemon, which is available in the Debian package `tftpd-hpa`. First install following additional needed packages:

```
# apt-get install dhcp3-server syslinux tftpd-hpa
```

Then set up the DHCP daemon. A sample configuration file can be found in `/usr/share/doc/fai/examples/etc/dhcpd.conf`. Copy this file to `/etc/dhcp3/dhcpd.conf`. Then enable the special tftp daemon using this line in file `/etc/inetd.conf`:

```
tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd -s /boot/fai
```

The install client then loads the pxelinux boot loader which receives its configuration via TFTP from a file in the directory `/boot/fai/pxelinux.cfg`. Using the command `fai-chboot(8)` you can define which kernel will be loaded by the PXE Linux loader and which additional parameters are passed to this kernel. You should read the manual pages, which give you some good examples. See `/usr/share/doc/syslinux/pxelinux.doc` for more information about how to boot such an environment. The PXE environment uses the original kernel image (not the netboot image made by mknbi-linux) which is copied to `/boot/fai/vmlinuz-install`.

3.3 Creating a boot floppy

If your network card can't boot by itself, you have two options. The first is to create a small boot floppy that uses etherboot, so you can use DHCP and TFTP to get the install kernel that was created with `mknbi-linux(8)`. A lot of ethernet cards support booting via ethernet

[2] If you have problems booting with a 3Com network card and get the error "BOOTP record too large" after the kernel is transfered to the computer, try the imggen-1.00 program to convert the netboot image to a installimage_3com image. I had this problem using netboot 0.8.1-4 and Image Creator for MBA ROMs v1.01, Date: Nov 26, 2001 but only on an Athlon computer.

if a special boot EPROM is inserted or booted from floppy rom-o-matic.net/. In depth documentation about booting via ethernet may be found at etherboot.sourceforge.net. The second option is to boot via floppy disk that is created with the command make-fai-bootfloppy(8). 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 or set other variables, if desired. Do not enable BOOTP support when you have a DHCP server running in your network and vice versa. This could lead to missing information. There's also a manual page for make-fai-bootfloppy(8). If you have no BOOTP or DHCP server, supply the network configuration as kernel parameters. The format is

ip=<client-ip>:<server-ip>:<gw-ip>:<netmask>:<hostname>:<device>:<autoconf>
for setting up the network and nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>]
for specifying the nfsroot (which is required as the default path is not suitable for FAI.)

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

3.4 Booting from a CD-ROM

It's possible to perform an automatic installation from CD-ROM without an install server. The CD-ROM contains all data needed for the installation. The command fai-cd(8) puts the nfsroot, the configuration space and a subset of the Debian mirror onto a CD-ROM. The partitial mirror is created using the command fai-mirror(1) which contains all packages that are used by the classes used in your configuration space. A sample ISO image is available at www.informatik.uni-koeln.de/fai/fai-cd/.

3.5 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.

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 by running the following command simultaneously on the server:

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

After the hosts have 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 this 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.

3.6 Configuration of the BOOTP daemon

You should only use this method if you can't use a DHCP server, since it's easier to create and manage the configuration for DHCP. An example configuration for the BOOTP daemon can be found in `/usr/share/doc/fai/examples/etc/bootptab`.

```
# /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:

.failocal:\
 :tc=.faiglobal:\
 :sa=FAISERVER:\
 :ts=FAISERVER:\
 :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:\
 :nt=time.rrz.uni-koeln.de,time2.rrz.uni-koeln.de:

# now one entry for each install client
demohost:ha=0x00105A240012:bf=demohost:tc=.failocal:T172="verbose sshd create
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 the hosts *demohost* 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. The `/etc/resolv.conf` file will be created using this list of DNS servers and the domain name.

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

The tag for time servers (`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[3] The 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 an additional kernel parameter, the flags must be separated with a comma. If you do not have full control over the BOOTP or DHCP daemon (because this service is managed by a central service group) you can also define the variable *FAI_ACTION* in the `/fai/class/*.var` scripts. 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
```

The BOOTP daemon 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.

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 careful 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.

3.6.1 Troubleshooting BOOTP daemon

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

```
# bootpd -d7
```

[3] T170=FAI_LOCATION (now defined in `fai.conf` and T171=FAI_ACTION. You can define theses variables in a class/*.var script. But for backward compatibility, you can define theses variables also from a BOOTP or DHCP server.

3.7 Configuration of the DHCP daemon

An example for `dhcp.conf(5)` is available in `/usr/share/doc/fai/examples/etc`, which is working with version 3.x of the DHCP daemon. Start using this example and look at all options used therein. One issue to bear in mind when configuring your DHCP daemon is that the daemon needs to supply the nfsroot path to the client because the kernel uses a path different from `/usr/lib/fai/nfsroot` by default. If you make any changes to the DHCP daemon configuration, you must restart the daemon.

```
# /etc/init.d/dhcp3-server restart
```

Therefore it's recommended to only supply data into this configuration file, which doesn't change frequently. By default, the DHCP daemon writes its log files to `/var/log /daemon.log`. The command `fai-chboot(8)` is used for creating a per host configuration for the pxelinux environment.

3.8 Boot messages

These are the messages when booting from floppy disk.

```
GRUB loading stage2..............
< now the grub menu with multiple boot options is displayed >
BOOTING 'FAI-BOTH'
kernel (fd0)/vmlinuz-2.4.27 root=/dev/nfs ip=both
  [Linux-bzImage, setup=0x1400, size=0xd8450]

Uncompressing Linux... OK, booting the Kernel.
Linux version 2.4.27 (root@kueppers) (gcc version 2.95.4 20011002
.
.
.
```

After this, the rest of the boot messages will be equal to those when booting from network card. When booting from network card with PXE you will see:

```
Managed PC Boot Agent (MBA) v4.00
.
.
Pre-boot eXecution Environment (PXE) v2.00
.
.
DHCP MAC ADDR: 00 04 75 74 A2 43
```

```
DHCP.../
CLIENT IP: 192.168.1.12 MASK: 255.255.255.0  DHCP IP: 192.168.1.250
GATEWAY IP: 192.168.1.254

PXELINUX 2.11 (Debian, 2004-09-19)  Copyright (C) 1994-2004 H. Peter Anvin
UNDI data segment at:   0009D740
UNDI data segment size: 3284
UNDI code segment at:   00090000
UNDI code segment size: 24C0
PXE entry point found (we hope) at 9D74:00F6
My Ip address seems to be C0A801C0 192.168.1.12
ip=192.168.1.12:192.168.1.250:192.168.1.254:255.255.255.0
TFTP prefix:
Trying to load pxelinnux.cfg/00-04-75-74-A2-43
Trying to load pxelinnux.cfg/C0A801C0
Loading vmlinuz-install.....................Ready.
Uncompressing Linux... OK, booting the Kernel.
Linux version 2.4.27 (lange@dom) (gcc version 3.3.5 (Debian 1:3.3.5-8))
.
.
.
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 192.168.1.250, my address is 192.168.1.12
IP-Config: Complete:
   device=eth0, addr=192.168.1.12, mask=255.255.255.0, gw=192.168.1.254,
   host=demohost, domain=localdomain, nis-domain=(none),
   bootserver=192.168.1.250, rootserver=192.168.1.250, rootpath=/usr/lib/fai/
Looking up port of RPC 1000003/2 on 192.168.1.250
Looking up port of RPC 1000005/1 on 192.168.1.250
VFS: Mounted root (nfs filesystem).
.
.
             ------------------------------------------------------
               Fully Automatic Installation for Debian GNU/Linux
               FAI 2.10.1, 20 april 2006    Copyright (c) 1999-2005

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

Calling task_confdir
Kernel parameters: ip=dhcp devfs=nomount FAI_ACTION=install root=/dev/nfs FAI
Reading /tmp/fai/boot.log
Configuration space /fai mounted from faiserver:/usr/local/share/fai
Can't connect daemon on faiserver. Monitoring disabled.
Calling task_setup
FAI_FLAGS: verbose sshd createvt syslogd
```

```
Press ctrl-c to interrupt FAI and to get a shell
Calling task_defclass
/usr/bin/fai-class: Defining classes.
Executing /fai/class/10-base-classes.
10-base-classes      OK.
Executing /fai/class/20-hwdetect.source.
loading kernel module rtc
.
.
List of all classes: DEFAULT LINUX I386 FAIBASE GRUB DHCPC DEMO demohost LAST
Calling task_defvar
Executing FAIBASE.var
Loading keymap(s) us-latin1 ...done.
Calling task_action
FAI_ACTION: install
Performing FAI installation. All data may be overwritten!
.
.
Calling task_configure
Executing    shell: FAIBASE/10-misc
FAIBASE/10-misc      OK.
Executing    shell: FAIBASE/20-save_diskvar
FAIBASE/20-save_diskvar OK.
Executing    shell: FAIBASE/30-interface
.
.
Sun Apr 10 01:03:10 CEST 2005
The installation took 480 seconds.
Calling task_chboot
Calling hook: savelog.LAST
Congratulations! No errors found in log files.
savelog.LAST         OK.
Calling task_savelog
Calling task_faiend
Press <RETURN> to reboot or ctrl-c to execute a shell
```

When the copyright message of FAI is shown, the install client has mounted the nfsroot[4] to the clients' root directory `/`. This is the whole filesystem for the client at this moment. After `task_confdir` is executed, the configuration space is mounted or received from a CVS repository. Before the installation is started (*FAI_ACTION=install*) the computer beeps three times. So, be watchful when you hear three beeps but you do not want to perform an installation!

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

3.8.1 Troubleshooting the boot messages

This is the error message you will see, when your network card is working, but the install server does not export the configuration space directory to the install clients, mostly a problem of missing permissions on the server side.

```
Root-NFS: Server returned error -13 while mounting /usr/lib/fai/nfsroot
VFS: Unable to mount root fs via NFS, trying floppy.
VFS: Cannot open root device "nfs" or 02:00
Kernel panic: VFS Unable to mount root fs on 02:00
```

Use the following command to see which directories are exported from the install server (named kueppers):

```
showmount -e kueppers
```

The following error message indicates that your install client doesn't get an answer from a DHCP server. Check your cables or start the `dhcpd(8)` daemon with the debug flag enabled.

```
PXE-E51: No DHCP or BOOTP offers received
Network boot aborted
```

These are the messages when you are using the BOOTP method and no BOOTP server replies.

```
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 the driver for your network card into a new kernel. This driver must not be a kernel module. The README file of the package fai-kernels `/usr/share/doc/fai-kernels/README` describes how to compile custom fai kernels. After that, adjust the variable *KERNELPACKAGE* in `/etc/fai/make-fai-nfsroot.conf` and rebuild the nfsroot with the command `make-fai-nfsroot(8)`. After that, you have to create a new boot floppy if you need it. Now your network card should be recognized and the install kernel should mount the nfsroot successfully.

3.9 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` is added to *FAI_FLAGS*. The encrypted password is set with the variable *FAI_ROOTPW* in `/etc/fai/make-fai-nfsroot.conf` and defaults to "fai". You can create the encrypted password using `mkpasswd(1)` and use the `crypt(3)` or md5 algorithm. This is only the root password during the installation process, not for the new installed system. You can also log in without a password when using *SSH_IDENTITY*. To log in from your server to the install client (named demohost in this example) use:

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

You now have 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 you can execute various commands (`dmesg`, `lsmod`, `df`, `lspci`, ...). Look at the log file in `/tmp/fai`. There you can find much information about the boot process. All log files from `/tmp/fai` are also written to the *$LOGSERVER* (if not defined: the install server) into the directory `~fai/demohost/sysinfo/`[5]

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 filesystem table is found, the partitions are mounted according to this information. Here's an example:

```
demohost:~# df

Filesystem 1k-blocks     Used Available Use% Mounted on
rootfs       2064192  1071184    888152  55% /
/dev/root    2064192  1071184    888152  55% /
shm            63548       76     63472   1% /tmp
kueppers:/usr/local/share/fai
             2064192   994480   964856   51% /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
```

[5] More general: `~$LOGUSER/$HOSTNAME/$FAI_ACTION/`. Two additional symbolic links are created. The symlink `last` points to the log directory of the last fai action performed. The symlinks `last-install` and `last-sysinfo` point to the directory with of the last corresponding action. Examples of the log files can be found on the FAI homepage.

This method can be used as a rescue environment! In the 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:

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

3.10 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. The received information is written to `/tmp/fai/boot.log`. An example of the result of a DHCP request can be found in 'The setup routines of the install clients' on page 34.

3.11 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

The following tasks 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 which 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 44.

The installation time is determined by the amount of software but also by the speed of the processor and hard disk. Here are some sample times. All install clients have a 100Mbit network card installed. Using a 10 Mbit LAN does not increase the installation time considerably, so the network will not be the bottleneck when installing several clients simultaneously.

```
Athlon XP1600+    , 896MB,SCSI disk,   1 GB software  6 min
AMD-K7, 500MHz    , 320MB, IDE disk, 780 MB software 12 min
PentiumPro 200MHz , 128MB, IDE disk, 800 MB software 28 min
Pentium III 850MHz, 256MB, IDE disk, 820 MB software 10 min
Pentium III 850MHz, 256MB, IDE disk, 180 MB software  3 min
```

4.1 Monitoring the installation

You can monitor the installation of all install clients with the command `faimond(8)`. All clients check if this daemon is running on the install server (or the machine defined by the variable *monserver*. Then, a message is sent when a task starts and ends. The fai monitor daemon prints this messages to standard output. In the future, there will be a graphical frontend available.

4.2 Set up FAI

After the install client has booted, only the script `/usr/sbin/fai`[1] is executed. This is the main script which 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.

4.3 Defining classes, variables and loading kernel modules

Now the script `fai-class(1)` is used to define classes. Therefore several scripts in `/fai /class/` are executed to define classes. All scripts matching `[0-9]*` (they start with a digit) are executed in alphabetical order. Every word that these scripts print to the standard output are interpreted as class names. Scripts ending in `.source` are sourced, so they can define new classes by adding these classes to the variable *newclasses* (see `06hwdetect.source` for an example). The output of these scripts is ignored. These classes are defined for the 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. The 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 the whole configuration from classes. The scripts `06hwdetectsource` loads

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

kernel modules on demand. The complete description of all these scripts can be found in 'Scripts in `/fai/scripts`' on page 42.

The script `30menu.source` pops up a little menu and asks the user which kind of installation should be performed (e.g. CAD workstation, notebook, scientific workstation, work group server, Gnome desktop...). Keep in mind that this won't lead to a fully automatic installation ;-)

After defining the classes, every file matching `*.var` with a prefix which matches a defined class is executed to define variables. There, you should define the variable *FAI_ACTION* and others. By default, *FAI_ACTION* is defined via the command `fai-chboot(8)`.

4.4 Partitioning local disks, creating filesystems

For disk partitioning exactly one disk configuration file from `/fai/disk_config` is selected using classes. This file describes how all the local disks will be partitioned, where filesystems should be created (and their types like ext2, ext3, reiserfs), and how they are mounted. It's also possible to preserve the disk layout or to preserve the data on certain partitions. It's done by the command `setup_harddisks`, which uses `sfdisk` for partitioning. The format of the configuration file is described in 'Hard disk configuration' on page 38.

During the installation process all local filesystems are mounted relative to `/tmp/target`. For example `/tmp/target/home` will become `/home` in the new installed system.

4.5 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(8)` installs all packages using `apt-get(8)` or `aptitude(1)` without any manual interaction needed. If a packages requires another package, both commands resolve 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 41.

4.6 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 which match a class name in `/fai/scripts` will be executed. If `/fai/scripts/`*classname*`/` is a directory,

all scripts that match `[0-9][0-9]*` in this directory are executed. So it is possible to have several scripts of different types (shell, cfengine, ...) to be executed for one class. FAI comes with some examples 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 42.

4.7 Save log files

When all installation tasks are finished, the log files are written to `/var/log/fai/$HOSTNAME/install/` [2] on the new system and to the account on the install server if *$LOGUSER* is defined in `/etc/fai/fai.conf`. It is also possible to specify another host as log saving destination through a file in `/fai/class/`. Additionally, two symlinks will be created to indicated the last directory written to. This method uses rsh/rcp or ssh/scp and is the default.

You can use other methods to save logs to the remote server. The currently selected method is defined by the *$FAI_LOGPROTO* variable in file `/etc/fai/fai.conf`:

ftp This option saves logs to the remote FTP server defined by the *$LOGSERVER* variable (*$SERVER* value is used if not set). Connection to the FTP server is done as user *$LOGUSER* using password *$LOGPASSWD*. The FTP server log directory is defined in *$LOGREMOTEDIR*. These variables are also defined in file `/etc/fai/fai.conf`. You need write access for the *$LOGREMOTEDIR* on the FTP server.

All files in the directory `/tmp/fai` are copied to the FTP server following this example: `ftp://$LOGUSER:$LOGPASSWD@$LOGSERVER/$LOGREMOTEDIR/`.

none Don't save the log file to the install server.

4.8 Reboot the new installed system

At last the system is automatically rebooted if "reboot" was added to *FAI_FLAGS*. Normally this should boot the new installed system from its second boot device, the local hard disk. To skip booting from network card, you can use the command `fai-chboot(8)` to enable local-boot. If using a boot floppy you have to remove the floppy from the floppy drive. Otherwise the installation would be performed again. Read 'Changing the boot device' on page 43 for how to change the boot device.

[2] `/var/log/fai/localhost/install/` is a link to this directory.

Chapter 5

Plan your installation, and FAI installs your plans

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

Will I create a Beowulf cluster, or do I have to install some desktop machines?

How does my LAN topology looks like?

Do I have uniform hardware? Will the hardware stay uniform in the future?

Does the hardware need a special kernel?

How should the hosts be named?

How should the local hard disks be partitioned?

Which applications will be run by the users?

Do the users need a queueing system?

What software should be installed?

Which daemons should be started, and what should the configuration for these look like?

Which remote filesystems should be mounted?

How should backups be performed?

Do you have sufficient power supply?

How much heat do the cluster nodes produce and how are they cooled?

You also have to think about user accounts, printers, a mail system, 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 about how exactly to install a computer. The central configuration space for all install clients is located on the install server in `/usr/local/share/fai` and its subdirectories. This will be mounted by the install clients to `/fai`. It's also possible to receive all the configuration data from a `cvs(1)` repository. The 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 filesystem creation.

`debconf/` This directory holds all `debconf(8)` data. The format is the same that is used by `debconf-set-selections(8)`.

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

`scripts/` Script for local site customization.

`files/` Files used by customization scripts. Most files are located in a subtree structure which reflects the ordinary directory tree. For example, the templates for `nsswitch.conf` are located in `/fai/files/etc/nsswitch.conf` and are named according to the classes that they should match: `/fai/files/etc/nsswitch.conf/NIS` is the version of `/etc/nsswitch.conf` to use for the NIS class. Note that the contents of the files directory are not automatically copied to the target machine, rather they must be explicitly copied by customization scripts using the `fcopy(8)` command.

`files/packages/` THE USE OF THIS DIRECTORY IS NOW OBSOLETE.

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

The main installation command `fai(8)` uses all these subdirectories in the order listed except for hooks. The FAI package contains examples for all these configuration scripts and files in `/usr/share/doc/fai/examples`. Copy the configuration examples 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/doc/fai-doc/examples/simple/* /usr/local/share/fai
# chown -R fai /usr/local/share/fai
```

These files contain simple configuration for some example hosts. Depending on the hostname used, your computer will be configured as follows:

demohost A machine which needs only a small hard disk. This machine is configured with network (as DHCP client), and an account demo is created.

gnomehost A GNOME desktop is installed, and the account demo is created.

other hostnames Hosts with other hostname will only use the main class FAIBASE.

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 filesystem via NFS from the install server and `init(8)` starts the script `/usr/sbin/fai`. This script 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`, and an operating system specific file [1]. All important tasks of the installation are called via the subroutine `task` appended by the name of the task as an option (e.g. `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 and its hooks can be skipped on demand by using the subroutine `skiptask()`.

Now follows the description of all default tasks.

confdir The kernel appended parameters define variables, the syslog and kernel log daemon are started. The list of network devices is stored in *$netdevices*. Then additional parameters are fetched from a DHCP or BOOTP server and also additional variables are defined. The DNS resolver configuration file is created. The configuration space is mounted from the install server to `/fai` or it is checked out from the corresponding `cvs(1)` repository. To use a cvs repository, you have to set the variables *$FAI_CVSROOT, $FAI_CVSTAG,*

[1] `/usr/share/fai/subroutines-linux` for Linux, `/usr/share/fai/subroutines-sunos` for Solaris.

$FAI_CVSMODULE. For details look a the subroutine `get_fai_cvs()`. After that, 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.

defclass Calls `fai-class(1)` to define classes using scripts and files in `/fai/class` and classes from `/tmp/fai/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/fai/additional.var`, this file is also sourced.

action Depending on the value of *$FAI_ACTION* this subroutine decides which action FAI should perform. The default available actions are: `sysinfo`, `install` and `softupdate`. If *$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.

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

softupdate This task, executed inside a running system via the `fai(8)` command line interface, performs a softupdate. See chapter 'Using FAI for online updates' on page 56 for details.

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

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

extrbase Extracts the base tar file `base.tgz`, which consists of all required packages. This is a snapshot of a basic Debian system created by `debootstrap(8)`

mirror If a local Debian mirror is accessed via NFS (when *$FAI_DEBMIRROR* is defined), this directory will be mounted to *$MNTPOINT*.

debconf Calls `fai-debconf(8)` to set the values for the debconf database.

prepareapt Set up resolv.conf and some other file, for the next task updatebase.

updatebase Updates the base packages of the new system and updates the list of available packages. It also fakes some commands (called diversions) inside the new installed system using `dpkg-divert(8)`.

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

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

finish Unmounts all filesystems in the new installed system and removes diversions of files using the command `fai-divert`.

faiend Wait for background jobs to finish (e.g. emacs compiling lisp files) and automatically reboots the install clients or waits for manual input before reboot.

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

savelog Saves log files to local disk and to the account *$LOGUSER* on *$LOGSERVER* (defaults to the install server). Currently the file `error.log` will not be copied to the log server.

6.3 The setup routines of the install clients

After the subroutine `fai_init` has done some basic initialization (create ramdisk, read `fai.conf` and all subroutines definitions, set path, print copyright notice), the setup continues by calling the task `confdir` and the task `setup`. The command `get-boot-info` is called to get all information from the BOOTP or DHCP server. This command writes the file `/tmp/fai/boot.log`, which then is sourced to define the corresponding global variables. This is an example for this log file when using a DHCP server.

```
# cat /tmp/fai/boot.log

netdevices_all="eth0
eth0 eth0"
netdevices_up="eth0"
netdevices="eth0"
BROADCAST='192.168.1.255'
DOMAIN='localdomain'
DNSSRVS='192.168.1.1'
DNSSRVS_1='192.168.1.1'
HOSTNAME='demohost'
IPADDR='192.168.1.12'
NETWORK='192.168.1.0'
GATEWAYS='192.168.1.250'
GATEWAYS_1='192.168.1.250'
SERVER='faiserver'
NETMASK='255.255.255.0'
```

Additional information is passed via the kernel command line or read from `/etc/fai/fai.conf`. When booting with PXE, command line parameters are created using `fai-chboot(8)`. The variable *$FAI_FLAGS* contains a space separated list of flags. The following flags are known:

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

debug Create debug output. No unattended installation is performed. During package installation you have to answer all questions of the postinstall scripts on the client's console. A lot of debug information will be printed out. This flag is only useful for FAI developers.

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. Setting this flag is useful for debugging. If you want an installation which should not be interruptible, do not set this flag.

reboot Reboot the install client after installation is finished without typing RETURN on the console. This is 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.

6.4 The class concept

Classes determine which configuration file to choose from a list of available templates. Classes are used in all further tasks of the installation. To determine which config file to use, an install client searches the list of defined classes and uses all configuration files that match a class name. It's also possible to use only the configuration file with the highest priority since the order of classes define the priority from low to high. There are some predefined classes (DEFAULT, LAST and the hostname), but classes can also be listed in a file or defined dynamically by scripts. So it's easy to define a class depending on the subnet information or on some hardware that is available on the install client.

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 (or on the first one) whose file names are also a class. The following loop implements this function in pseudo shell code:

```
for class in $all_classes; do
if [ -r $config_dir/$class ]; then
   your_command $config_dir/$class
```

```
   # exit if only the first matching file is needed
fi
done
```

Therefore it is possible to add a new file to the configuration without changing the script. This is because the loop automatically detects new configuration files that should be used. Unfortunately cfengine does not support this nice feature, so all classes being used in cfengine also need 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 can be configured to become an FTP server by just adding the class `FTP` to it. 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 for 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 be 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 be defined and then added for a given host. This is because most of the time the configuration data is not specific for one host, but can be shared among several hosts.

6.5 Defining classes

The task *defclass* calls the script `fai-class(1)` to define classes. Therefore, scripts matching `[0-9][0-9]*` in `/fai/class` are executed. Additionally, a file with the hostname may contain a list of classes. For more information on defining class, read the manual pages for `fai-class(1)`.

The list of all defined classes is stored in the variable *$classes* and saved to `/tmp/fai/FAI_CLASSES`. The list of all classes is transfered to `cfengine`, so it can use them too. The script `10-base-classes` (below is a stripped version) is used to define classes depending on the host name. First this script defines the class with the name of the hardware architecture in uppercase letters.

```
# cat 10-base-classes

# echo architecture and OS name in upper case. Do NOT remove these two lines
uname -s | tr '[:lower:]' '[:upper:]'
dpkg --print-installation-architecture | tr /a-z/ /A-Z/

[ -f /boot/RUNNING_FROM_FAICD ] && echo "FAICD"

# use a list of classes for our demo machine
case $HOSTNAME in
    demohost)
        echo "FAIBASE GRUB DHCPC DEMO" ;;
    gnomehost)
        echo "FAIBASE GRUB DHCPC DEMO XFREE GNOME";;
    *)
        echo "FAIBASE GRUB DHCPC" ;;
esac
```

The script `20-hwdetect.source` uses the default Debian commands to detect hardware and to load some kernel modules. If some specific hardware is found, it can also define a new class for it. You can find messages from modprobe in `/tmp/fai/kernel.log` and on the fourth console terminal by pressing `Alt-F4`.

6.6 Defining Variables

The task `defvar` defines the variables for the install client. Variables are defined by scripts in `class/*.var`. 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. The following variables are used in the examples and may also be useful for your installation:

FAI_ACTION Set the action fai should perform. Normally this is done by `fai-chboot(8)`. If you can't use this command and are not using a BOOTP server, define it in the script `LAST.var`.

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

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

rootpw The encrypted root password for the new system. You can use `crypt(3)` or md5 encryption for the password.

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.

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

6.7 Hard disk configuration

The script `setup_harddisks.pl` partitions and formats the local disks. It uses all configuration files in `/fai/disk_config/` which are also defined as classes. Lines beginning with # are comments. The config file `/fai/disk_config/FAIBASE` is a generic description for one hard disk (IDE or SCSI), which most installations should be able to adapt. 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 filesystems and create the files `/tmp/fai/fstab` and `/tmp/fai/disk_var.sh`, which contains definitions of boot and root partitions.

The following example is a configuration for the first IDE disk `disk1` and for the second SCSI disk `disk2` The numbering of the disks comes from the order in `/proc/partitions`.

```
# <type> <mount point> <size in MB> [mount options]  [;extra options]

disk_config disk1

primary   /           200          defaults,errors=remount-ro
logical   /home       100-300
logical   /scratch1   10-          defaults,nosuid   ; -i 15000 -m 0


disk_config disk2

primary   /tmp        300-500      rw                ;ext2
primary   /backup     preserve2    rw
logical   swap        50-100
logical   /scratch2   100-300      rw                ;-m 30
logical   -           preserve7
logical   /var        100                            ;-j
logical   /var/tmp    preserve9                      ;format
primary   /tmp/mytmp  -300
```

Every disk configuration starts with the command `disk_config` followed by `diskX` where `X` is the number of the HDD. The Linux device names `/dev/hda` and `/dev/sda` correspond to `disk1`, `disk2` corresponds to `/dev/hdb` and `/dev/sdb` and so on.

[2] Currently this script uses the command `sfdisk(8)`, which isn't available on SUN SPARC, IA64 and PowerPC.

After `disk_config` one line containing the type, mount point and size is added for each partition on the harddisk. Mount options and additional parameters for `mke2fs` – separated from the mount options by a semicolon – can be added.

Type There are two types of partitions: primary and logical. Primary partitions are bootable, but there is a maximum of four primary partitions on each disk. The Linux root filesystem must be of this type.

All other partitions are called logical. Because logical partitions are gathered internally in one big primary partition, only three primary partitions can be used if logical partitions are defined. Normally only one primary partition for the root filesystem is created and all others are logical, like `disk1` in the example above.

Mount point The mount point is the full path (beginning with a slash) for the filesystem. The value `swap` defines a Linux swap partition. Both types will be automatically added to `/etc/fstab`. A dash `-` indicates that the partition will not be mounted and can be used for other types of filesystems (FAT, NTFS, UFS, MINIX, ...)

Size This is the size of the partition in megabytes. This value is rounded up to fit to a cylinder number. There are several ways of defining the size:

```
"200" means about 200MB, no more no less
"100-300" sets a 100MB minimum and a 300MB maximum
"10-" sets a minimum of 10MB and a maximum of the disk size
"-300" sets a minimum of 1MB and a 300MB maximum
```

By default, a new filesystem (currently of type ext2 or swap) will be created, and all data on the partition is lost. The meaning of `preserve<no>` will be described later.

Calculating the partition size: If an interval is defined for several partition sizes, the script maximizes the values by preserving the ratio between them.

Mount options The mount options will be copied to `/etc/fstab`. An empty field sets the option to `defaults` (see `mount(8)`).

Extra options The last field is a space separated extra options list. The following options are known:

```
boot          : Make this partition the boot-partition (the
                Linux root filesystem is the default).
-i <bytes>    : bytes per inode (ext2/3 only)
-m <blocks>   : reserved blocks percentage (ext2/3 only)
-j            : Create the filesystem with an ext3 journal.
-c            : Check for bad blocks.
ext2          : Flag as ext2 instead of auto in /etc/fstab.
ext3          : Flag as ext3 instead of auto in /etc/fstab.
swap          : swap partition
dosfat16      : DOS 16 bit FAT filesystem
```

```
winfat32     : Win95 FAT32 filesystem
reiser       : Create a ReiserFS filesystem, not an ext2.
xfs          : XFS
format       : Always format even if preserve is specified.
writable     : Mounts a preserved partition writable.
lazyformat   : Do not format if partition has not moved.
```

The order of the extra options is not relevant. For more information see `mke2fs(8)`.

Thus, we have the following interactions between `-j`, `ext2` and `ext3` :

```
<no option> : An ext2 fs flagged as auto in the fstab
-j          : An ext3 fs flagged as auto in the fstab.
ext2        : An ext2 fs flagged as ext2 in the fstab.
-j ext2     : An ext3 fs flagged as ext2 in the fstab.
-j ext3     : An ext3 fs flagged as ext3 in the fstab.
ext3        : An ext2 fs flagged as ext3 in the fstab. !!BAD!!
```

Using `auto` in the fstab for ext3 filesystems enables a non-ext3-enabled kernel or tool to cope with these partitions.

It is possible to preserve the size and even the existing data on a partition. To preserve only the partition size, the number of the partition must be unchanged and the size must be specified as `preserve<no>` The number `<no>` is the device number (as in `/dev/hda<no>`, or see the output of `df`) of the partition. Primary partitions are numbered from one to four, the numbers for logical partitions begin at five.

Problems were reported (February 2003) when using more than two primary partitions and trying to preserve a logical partition. If you have this problem, try to use only two primary partitions.

In the following example, the partition numbers (= device number) are also shown for disk `disk2`:

```
primary   /tmp        300-500     #  1
primary   /backup     preserve2   #  2
logical   swap        50-100      # (3)   5
logical   /scratch2   100-300     # (3)   6
logical   -           preserve7   # (3)   7
logical   /var        100         # (3)   8
logical   /var/tmp    preserve9   # (3)   9
primary   /tmp/mytmp  -300        #  4
```

The first two partitions are of type primary, so they get the numbers 1 and 2. The logical partitions start at 5 and the last gets number 9. All logical partitions define the primary partition 3, but this number is not used. So if you want to preserve `/dev/hda7` you have to insert a minimum of two logical partitions before it.

Lazyformating partitions is another method to preserve partitions after they were formatted once. This is useful to design systems which can be reinstalled without loosing data on partitions like `/home` or `/var/log` or `/var/lib/mysql` or whatever. You can even lazyformat the swap partition to gain a minor installation speed improvement after the first installation!

If you have a separate `/boot` partition, you must add the extra option `boot` to make it your boot partition. Otherwise your system will not be bootable. By default (if no boot option was specified) the root partition (`/`) will become the boot partition. `setup_harddisks` will write some variables containing the information about boot partition and boot device to `/tmp/fai/disk_var.sh`.

6.8 Software package configuration

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

```
# an example package class

PACKAGES taskinst
german

PACKAGES aptitude
adduser netstd ae
less passwd

PACKAGES remove
gpm xdm

PACKAGES aptitude GRUB
lilo- grub

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 package 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 misspellings. Any package which does not

exist, will be removed from the list of packages to install. So be careful not to misspell any package names.

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 tasks that are specified in the following lines using `tasksel(1)`. You can also use `aptitude` for installing tasks.

aptitude: Install all packages with the command `aptitude`. This will be the default in the future and may replace apt-get and taskinst. Aptitude can also install task packages.

aptitude-r: Same as aptitude with option `--with-recommends`.

unpack: Download package and unpack only. Do not configure the package.

dselect-upgrade Set package selections using the following lines and install or remove the packages specified. These lines are the output of the command `dpkg --get-selections`.

Multiple lines with lists of space separated names of packages follow the PACKAGES lines. All dependencies are resolved. Packages with suffix `-` (eg. `lilo-`)will be removed instead of installed. The order of the packages is of no matter.

A line which 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 example 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 command `PRELOAD` instead.

It's possible to append a list of class names after the command for apt-get. So this `PACKAGE` command will only be executed when the corresponding class is defined. So you can combine many small files into the file DEFAULT. `WARNING!` Use this feature only in the file DEFAULT to keep everything simple. See this file for some examples.

If you specify a package that does not exist this package will be removed from the installation list. You can also test all software package configuration files with the utility `chkdebnames`, which is available in `/usr/share/doc/fai/examples/utils/`.

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

6.9 Scripts in `/fai/scripts`

The default set of scripts in `/fai/scripts` 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.

The command `fai-do-scripts(1)` is called to execute all scripts in this directory. If a directory with a class name exists, all scripts matching `[0-9][0-9]*` are executed in alphabetical order. So it's possible to use scripts of different lanuages (shell, cfengine, Perl,..) for one class.

6.9.1 Shell scripts

Most scripts are Bourne shell scripts. 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 many short scripts, 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 want 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 script is 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 which are undefined. If a variable is undefined, the whole cfengine script will abort. Study the examples that are included in the fai package. More information can be found in the manual page `cfengine(8)` or at the cfengine homepage `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.

So, normally the boot sequence of the BIOS will remain unchanged and your computer should always boot first from its network card and the second boot device should be the local disk. Then, it will get an install kernel image from the install server, when an installation should be performed, or we can tell pxelinux to boot from local disk. This is done using `fai-chboot(8)`. Here is how to set up a 3Com network card as first boot device. 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 second boot device after the network card. If booting from a FAI floppy disk, another solution can be used to skip a re-installation if the BIOS is configured to boot from the floppy disk first and you are not there to remove the floppy disk:

```
# lilo -R ...
```

will instruct the FAI floppy to boot from the hard disk only once (see `lilo(8)`). Thus after this first reboot, the FAI floppy disk can be used for another FAI installation.

6.11 Hooks

Hooks let you specify functions or programs which are run at certain steps of the installation process. Before a default task is called, FAI searches 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 some default tasks 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 task name as a prefix and a class name, separated by a dot. The prefix describes the time when the hook is called, if the class is 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/fai/fstab`. After that, the installation process will not try to partition and format a local hard disk, because a file `/tmp/fai/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 have the suffix `.source` (e.g. `partition.DEFAULT.source`) must be Bourne shell scripts and are sourced. So it's possible to redefine variables for the installation scripts.

In the first part of fai, 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.
- 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`.
- Partition the hard disk on an IA64 system, which needs a special partition table type that must be created with `parted(8)`. See `hooks/partition.IA64`.

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 `tftpd(8)`, `dhcpd3(8)` or `bootpd(8)`:

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

If the installation process finishes, the hook `faiend.LAST` searches all log files for common errors and writes them to the file `error.log`. So, you should first look into this file for errors. Also the file `status.log` give you the exit code of the last command executed in a script. To be sure, you should look for errors in all log files.

Sometimes the installation seems to stop, but often 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` to make the installation process show all output from the postinst scripts on the console and get its input also from the 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 successfully installed computers are available on the FAI homepage.

Chapter 7

How to build a Beowulf cluster using FAI

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

7.1 Planning the Beowulf setup

The example of a Beowulf cluster consists of one master node and 25 clients. A big rack was assembled which all the cases were put into. A keyboard and a monitor, which are connected to the master server most of the time, were also put into the rack. But since we have very long cables for a monitor and a keyboard, they can also be connected to all nodes if something has to be changed in the 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 too much heat, so you will need an air conditioner. Will the power supplies of each node go to stand-by mode or are all nodes turned on simultaneously after a power failure?

All computers in this example 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 access the master node with the 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 are 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, NFS, ...) must

be available on the master server. I chose 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. If you have more than 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 times, until there's no fault in your configuration scripts. Therefore you should have physical access to the master server and one client node. So, 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 if it is your first computer installed with Debian. If you already have a host running Debian, you can also install the master server via FAI. Create a partition on `/files/scratch/debmirror` for the local Debian mirror with more than 9.0GB GB space available.

7.2.1 Set up the network

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

```
# Beowulf cluster connection
auto eth1
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` file:

```
# create these entries with the Perl one liner
# perl -e 'for (1..25) {printf "192.168.42.%s atom%02s\n",$_,$_;}'

# 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 who edits the configuration space and manages the local Debian mirror:

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

This user should also be in the group *linuxadmin*.

```
# adduser tom linuxadmin
```

First set the NIS domainname name by creating the file `/etc/defaultdomain` and call `domainname(8)`. To initialize the master server as NIS server call `/usr/lib/yp/ypinit -m`. Also edit `/etc/default/nis` so the host becomes a NIS master server. 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
```

You will find much more information about NIS in the `NIS-HOWTO` document.

7.2.3 Create a local Debian mirror

Now the user *tom* can create a local Debian mirror on `/files/scratch/debmirror` using `mkdebmirror`. You can add the option `--debug` to see which files are received. This will need about 9.0GB GB disk space for Debian 3.0 (aka woody). Export this directory to the netgroup `@faiclients` read only. Here's the line for `/etc/exports`

```
/files/scratch/debmirror *(ro)
```

7.2.4 Install FAI package on the master server

Add the following packages to the install server:

```
nucleus:/# apt-get install ntp tftpd-hpa dhcp3-server \
nfs-kernel-server etherwake fai fai-kernels
nucleus:/# tasksel -q -n install dns-server
nucleus:/# apt-get dselect-upgrade
```

Configure NTP so that the master server will have the correct system time.

It's very important to use the internal network name *atom00* for the master server (not the external name *nucleus*) in `/etc/dhcp3/dhcpd.conf` and `/etc/fai/make-fai-nfsroot.conf`. Replace the strings FAISERVER with `atom00` and uncomment the following line in `/etc/fai/make-fai-nfsroot.conf` so the Beowulf nodes can use the name for connecting to their master server.

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

7.2.5 Prepare network booting

Set up the install server daemon as described in 'Booting from network card with a PXE conforming boot ROM' on page 14. If you will have many cluster nodes (more than about 10) and you will use `rsh` in `/etc/fai/fai.conf` raise the number of connects per minute to some services in `inetd.conf`:

```
shell stream tcp  nowait.300  root /usr/sbin/tcpd /usr/sbin/in.rshd
login stream tcp  nowait.300  root /usr/sbin/tcpd /usr/sbin/in.rlogind
```

The user *tom* should have permission to create 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/doc/fai/examples/utils/* /usr/local/bin
```

Now, the user *tom* sets the boot image for the first beowulf node.

```
fai-chboot -IFv atom01
```

Now 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)` and store it in `/usr/local/share/fai/files/packages`.

7.3 Tools for Beowulf clusters

The following tools are useful for a Beowulf cluster:

tlink Change the symbolic link that points to the kernel image for booting from a network card. Only used when you boot using BOOTP.

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).

rup The command `rup(1)` shows briefly the CPU load of every host.

clusterssh The package clusterssh allows you to control multiple ssh or rsh sessions at the same time.

These are some common tools for a cluster environment:

rgang For a huge cluster try `rgang`. It's is a tool which executes commands on or distributes files to many nodes. It uses an algorithm to build a tree-like structure to allow the distribution processing time to scale very well to 1000 or more nodes (available at `fermitools.fnal.gov/` `abstracts/rgang/abstract.html`).

jmon For observing the resources of all clients (CPU, memory, swap,...) you can use `jmon(1)` which installs a simple daemon on every cluster node.

ganglia This toolkit is very good for monitoring your cluster with a nice web frontend. Available at `ganglia.sourceforge.net/`

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 packet to the network card, the computer will be turned on. The 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 3Com card using the 3c59x driver you must enable the WOL feature using the kernel module option `enable_wol`.

To wake up a computer use the command `ether-wake(8)`. Additional information is available from `www.scyld.com/expert/wake-on-lan.html`.

Chapter 8

FAI on other architectures and distributions

If you want to use FAI on other architectures than i386 you might need to take care of some things yourself. These are things that may have to be changed on other architectures:

make-nfsroot.conf FAI_DEBOOTSTRAP_OPTS must be adopted to the architecture you're using.

task partition: Currently setup_harddisks needs the command `sfdisk(8)`. If this is not available then write a short shell script which uses `parted(8)`, to partition the disks and for creating the file `fstab`. Alternatively you can use a hook (see 'Hooks' on page 44) to format and mount your partitions.

Boot loader: There are scripts for setting up `lilo(8)` and `grub(8)`. Here you may add support for your specific boot loader.

If you want to serve multiple nfsroot directories on one FAI server, you need to create specific config directories in `/etc` for fai, like `/etc/fai-sarge` and `/etc/fai-etch`. Then you need to set the *NFSROOT* variables to different directories and run `make-fai-nfsroot -c /etc/fai-sarge`.

8.1 FAI on AMD64

No problems. Have a look at `www.informatik.uni-koeln.de/fai/download/amd64/`

8.2 FAI on PowerPC

There's some stuff on `www.layer-acht.org/fai`. Most notably there are hooks for partitioning and config-files to setup bootloaders for oldworld and newworld.

8.3 FAI on IA64

There's one big IA64 Beowulf cluster running which was installed with FAI. Only the partitioning part has to be replaced by a short script, since sfdisk is not available on IA64.

8.4 FAI on Alpha

There is a Mini-HowTo available at pandora.aei.mpg.de/~steffeng/fai/fai-alpha.html

8.5 FAI for Suse, Redhat and Gentoo

Many people are interested in FAI for other (mostly RPM based) Linux distributions. I made some research and it should not be much work to implement it. But I need more help to implement it. If you are interested and would like to help me, please send an email to <fai@informatik.uni-koeln.de>.

8.6 FAI on SUN SPARC hardware running Linux

Although FAI is architecture independent, there are some packages which are only available for certain architectures (e.g. silo, sparc-utils). SUN SPARC computers can boot from their boot prompt and don't need a boot floppy. To boot a SUN use:

```
boot net:dhcp - ip=::::::dhcp
```

You have to convert the kernel image from ELF format to a.out format. 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 www.ultralinux.org for more information and www.sparc-boot.org/. . A success report is available at www.opossum.ch/fai/ and a HOWTO and a lot of examples can be found at toolbox.rutgers.edu/~amurphy/fai.

8.7 FAI for Solaris

FAI has also been ported for use with SUN Solaris OS installations in cooperation with Solaris jumpstart. Get the FAI sources and change to the `sunos` directory. There you can call `make` which creates the tarball `/tmp/fai-solaris.tar.gz`. You have to read the file `README.sunos` and have some knowledge about Solaris jumpstart.

The file format of the configuration files in `disk_config` and `package_config` are different than those for Linux.

Chapter 9

Advanced FAI

9.1 Using revision control for FAI configuration

If there is a team of administrators involved, a revision control/sourcecode management system like CVS can make coordination easier: many people can work on the configuration files simultaneously, while the system helps avoiding conflicts (and if they occur it helps resolving them). Another advantage lies in *branching*: while the administrator works out a new configuration and tries it out using a test system, other clients aren't disturbed in any way, because they use another *branch* of the configuration.

9.1.1 Setting up FAI for CVS based configuration

First you should setup a CVS repository and within it a module to store the FAI configuration files. In this example a CVS pserver will be used for *read-only* access to the configuration files by the clients, while ssh is used for the developers access (rw)[1].

The relevant variables in `/etc/fai/fai.conf` and `/etc/fai/make-fai-nfsroot.conf` for CVS are:

FAI_LOCATION This variable **must not** be set if you want to use CVS.

FAI_CVSROOT contains the cvsroot where the configuration is stored.

```
FAI_CVSROOT=":pserver:client@cvs.local.net:/var/lib/cvs"
```

FAI_CVSMODULE contains the module in the cvsroot where the configuration is stored.

```
FAI_CVSMODULE="config"
```

[1] CVS is quite flexible when it comes to different methods of access, so I recommend you to read further documentation to find the optimal solution in your environment

FAI_CVSTAG contains the *tag* of the CVS branch to be checked out by the client[2].

```
FAI_CVSTAG="STABLE"
```

If you use a CVS pserver for storing configuration files, the file `/root/.cvspass` has to exist and be valid in the nfsroot. CVS uses this file to get the password for the pserver. The easiest way to create it is

```
cvs -d$FAI_CVSROOT login
```

and then copy the generated line from your `~/.cvspass` into `/root/.cvspass` in the nfsroot.

9.2 Using FAI for online updates

FAI is even usable for system updates, using the same configuration as if initially installing. System update means updating the running system without doing a reinstallation. An updated client will almost look like a newly installed machine, though all local data is preserved (except of course newer configuration files introduced in the FAI config).

9.2.1 How does a softupdate work?

Softupdates use the same configuration files as a new FAI installation. They even use the default FAI commands, so they behave *nearly* in the same way as an installation, though some things are different:

- By default the old list of classes (created during the initial installation) is used, so `fai-class` is not called to define a new list of classes. This can be changed by calling `fai -N softupdate`.
- No partitioning and filesystem creation is performed.
- The basesytem isn't bootstrapped.
- FAI skips tasks only useful when installing, such as setting up a keymap or starting special daemons.
- FAI doesn't prevent software packages to (re-)start daemons.
- FAI doesn't reboot at the end of a softupdate.

Except these changes, things are the same as when installing a new computer:

[2]This is optional: if not set, *HEAD* will be used, which corresponds to the most recent revision.

1 Define classes (by default use old list) and variables.

2 Update the installed packages.

3 Install new software.

4 Call configuration scripts.

5 Save the logfiles.

9.2.2 How to run a softupdate

As softupdates use the same infrastructure as a FAI installation, you even start them by using the same command `fai(8)` which is used for installation:

```
/usr/sbin/fai softupdate
```

starts a softupdate.

How to do mass softupdates

Probably you don't want to run to each client and start a softupdate there locally, so a mechanism to start an update there has to be thought of.

Cron One possible solution is to use crontab entries on the clients to start an update, but in big installations you have to consider including a random-delay mechanism, because too many updates at the same time may produce too much traffic on your network.

Starting a softupdate remotely If you want more control when exactly a softupdate is run on the clients and maybe want to monitor it while it is running, you can install remote root login mechanisms on your clients, preferably using ssh in connection with a authorized key for root logins.

Tools like `clusterssh` allow you to login onto a group of clients at once and run `/usr/sbin/fai softupdate` there, while the results can be seen immediately in the terminals started for each host.

9.2.3 How to write a configuration suitable for softupdates

When you want to do softupdates, you have to be even more careful when writing your configuration: it has to be **idempotent**, i.e. running all the scripts twice should result in the same system configuration as running them once. Some things to keep an eye on:

- *Never* blindly append to files:

  ```
  echo $SOMETHING >> /etc/fstab
  ```

  is almost certainly wrong. Either check manually if the line already exists **before** appending or use cfengine's `AppendIfNoSuchLine` statement instead.

- Make use of FAI's environment variables to determine what to do in your configuration scripts! Some of the most important ones:

  FAI_ROOT points to the client's rootdir. In case of softupdates: `/`

  ROOTCMD contains a command for `chrooting` into the client. This is empty when doing softupdates (as `/` is already our root...).

  FAI_ACTION contains the currently executed action:

  `install` when installing.
  `softupdate` when updating

- Restart daemons if needed: most daemons only read their configuration when starting; if you modify it, you need to make them reload it using

  ```
  $ROOTCMD invoke-rc.d $somedaemon reload
  ```

  or even restart them

  ```
  $ROOTCMD invoke-rc.d $somedaemon restart
  ```

  when the configuration for `$somedaemon` has been changed[3].

- Other things like scheduling a reboot if a new kernel is installed

9.2.4 What if there are locally changed conffiles?

Short: there shouldn't be any!

Long: *if* you are using FAI `softupdate` to update client's configuration, you shouldn't do any local changes on the install clients, because they may be lost while updating. Backup copies are done by fcopy only on the local disk (by default, they are written to the same directory as the original file, with `.pre_fcopy` appended); if you want to save them together with the logfiles,

```
FAI_BACKUPDIR=$LOGDIR/backup
```

in `class/DEFAULT.var` will do the job.

[3] You can for example use `fcopy(8)`'s `postinst` script support for doing this; if other things than `fcopy` modify your conffiles, you have to keep track of the changes yourself.

How to detect locally changed files?

If you are playing with local configuration changes *despite all the warnings contained in this section*, there must be a way to check what has been changed locally. A simple approach would be to use `debsums -e`, but this method fails miserably if you modify conffiles in your FAI scripts, because it only checks against the version contained in the Debian package. A better proposal is to setup/abuse `tripwire` or `integrit` to scan for local changes and notify you about them.

Chapter 10

Various hints

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

When using HTTP access to a Debian mirror, the local `/var` partition on all install clients must be big enough to keep the downloaded Debian packages. Do not try with less than 250 Mbytes unless you know why. You can limit the number of packages installed at a time with the variable *$MAXPACKAGES*.

You can shorten some scripts by using one single fcopy command `fcopy -r /`.

If you rebuild the nfsroot, you will create a new ssh host key inside the nfsroot. Then logging in to an install client may fail, because the host key changes. You can use this:

```
ssh -o StrictHostKeyChecking=no root@installclient
```

You can calculate the IP subnet adress (which is used in make-fai-nfsroot.conf for the variable *FAICLIENTS* by using the nice tool ipcalc. Following example gives you the notation for a class C network (16) when the server netork interface has the IP address 123.45.6.123

```
ipcalc -nb 123.45.6.123 16|grep Network:
```

You can merge two directories which contain configuration information, if one is a global one, and the other a local one. We use it to merge the templates from the fai package, and our local configuration, which contains encrypted passwords and other information that should not be readable by others. If you remove a file in your local configuration, do not forget to remove this file also in the configuration space, otherwise it will still be used.

After calling `set-disk-info`, a list of all local hard disks is stored in *$disklist* and *$device_size* contains a list of disk devices and their sizes.

Use `fai-divert -a` if a postinst script calls a configuration program, e.g. the postinst script for package apache calls apacheconfig, which needs manual input. You can fake the configuration program so the installation can be fully automatic. But don't forget to use `fai-divert -R` to remove all faked scripts.

During the installation you can execute commands inside the newly installed system in a chroot environment by using `chroot /tmp/target` or just `$ROOTCMD` followed by the command you want to call; for example `$ROOTCMD dpkg -l` shows the packages installed on 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 to define classes for this host if the existing configuration files are not generic enough to deal with this new 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, the `ifclass` test or with class mechanisms for `cfengine` scripts.

If your computer can't boot from the network card, you do not always need to boot from floppy. Add the class `FAI_BOOTPART` and FAI will automatically create a lilo or grub entry for booting the FAI bootfloppy from this partition. So you can start the re-installation without a boot floppy. This will also make the test phase shorter, since booting from hard disk is much faster than booting from floppy. You can also set a password for this boot menu.

10.1 Using FAI after an Installation with the Debian-Installer

On `www.layer-acht.org/fai` you will find an example how to fully automatically install a system using the Debian Installer (d-i) in conjunction with FAI's new softupdate (see 'Using FAI for online updates' on page 56).

10.2 Useful functions for advanced administrators

fai-divert Add or remove a file to the list of diversions and replace the file with a dummy script. This is useful when a postinst script needs manual input. At the end of the installation all diversions are removed.

skiptask This given list of tasks are skipped. For use e.g. in `partition.DISKLESS`.