FAI Guide (Fully Automatic Installation)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
3.0.5	Mon, 08 Nov 2010 15	45:36 +0100	TL

Contents

Abstract FAI is a non-interactive system to install, customize and manage Linux systems and software configurations on computers as well as virtual machines and chroot environments, from small networks to large infrastructures and clusters.

This manual describes the Fully Automatic Installation package. This includes the installation of the package, planning and creating of the configuration and how to deal with errors.

1 Introduction

1.1 Availability

Homepage
http://fai-project.org

FAI wiki
http://wiki.fai-project.org

Download
http://fai-project.org/download

Entry for *sources.list*
```
http://fai-project.org/download lenny koeln
```

Mailing list
https://lists.uni-koeln.de/mailman/listinfo/linux-fai

Feedback
Send feedback and comment to fai@fai-project.org or to the mailing list.

Bugs
Use the Debian bug tracking system (BTS) http://bugs.debian.org for reporting errors.

User visible changes
http://fai-project.org/NEWS

Source tree
You can access the subversion repository containing the newest developer version of FAI using the following commands
```
svn co svn://svn.debian.org/svn/fai/trunk fai
```

Source tree via http
http://svn.debian.org/wsvn/fai/

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 again and again is boring — and will surely lead to errors. 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 perform 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, customize and manage Linux systems and software configurations on computers as well as virtual machines and chroot environments, from small networks to large infrastructures and clusters. 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 Linux 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. Chapter [plan] has some useful hints for this topic.

First, some terms used in this manual are described.

install server

> The host where the package *fai-server* is installed. It provides several services and data for all install clients. In the examples of this manual this host is called *faiserver*.

install client

> A host which will be installed using FAI and a configuration provided by 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. All configuration data is stored in a certain directory structure and is also called configuration space or config space for short. It includes information about:

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

nfsroot

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

---

[1]Solaris 8 Advanced Installation Guide at http://docs.sun.com
[2]http://www.science.uva.nl/pub/solaris/auto-install

1.4   How does FAI work

The install client which will be installed using FAI, is booted via network card or from CD or USB stick. It gets an IP address and boots a Linux kernel which mounts its root file system via NFS from the install server. After the kernel is loaded, the FAI startup script performs the automatic installation which doesn't need any interaction. First, the hard disks will be partitioned, file systems 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 file system, 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 re-installation.

- Easy creation of a virtualization environment or a chroot

- Hosts can boot from network card, CD, USB stick.

- Simple creation of the CD and USB stick.

- PXE with DHCP and BOOTP boot methods are supported.

- Lilo and grub support.

- ReiserFS, ext3 and XFS file system support.

- Software RAID and LVM support.

- Automatic hardware detection.

- 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.

- Can be used as a rescue system.

- Flexible system through easy class concept.

- Diskless client support.

- Easily add your own functions via hooks.

- Easily change the default behavior via hooks.

2   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 the package *fai-quickstart* on your install server (see [faisetup]).

- Edit *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/* /srv/fai/config/
```

- 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 by calling `fai-chboot` on the install server.

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

- If you want to try FAI without setting up a PXE+DNS+DHCP-environment: put the host names into */etc/hosts* inside the nfsroot at */srv/fai/nfsroot* and use a CD/DVD to boot the client.

- 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], [instprocess] and [config]!

3   Installing FAI

3.1   Requirements

The following items are required for an installation via FAI.

A computer
> The computer must have a network interface card [3]. Unless a diskless installation should be performed a local hard disk is also needed. No floppy disk, CD-ROM, keyboard or graphics adapter is needed.

DHCP server
> The clients need one of these daemons to obtain boot information.

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.

NFS-Root
> It is a directory which contains the whole file system for the install clients during installation. It must be exported via NFS, so the install clients can mount it. It will be created during the setup of the FAI package and is also called **nfsroot**.

---

[3]If you install from USB stick or CD you do not need a network card

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.

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, subversion or Git.

The NFS server will be enabled automatically when installing the *fai-server* package.

3.2   How to create a local Debian mirror

The script `mkdebmirror` [4] can be used for creating your own local Debian mirror. This script uses the script `debmirror(1)`. A partial Debian mirror only for i386 architecture for Debian 5.0 (aka lenny) without the source packages needs about 22GB of disk space. Accessing the mirror via HTTP will be the default way in most cases. To see more output from the script call `mkdebmirror -v`. 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 partial mirror containing all package for the simple examples from the package fai-doc will only need about 440MB 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:

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

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

3.3   Setting up FAI

To setup a FAI install server you need at least the packages *fai-server* and *fai-doc*. The package *fai-quickstart* contains dependencies on all required packages for an install server. Do not install the package *fai-nfsroot* on a normal 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 installed:
  apt-move{a} isc-dhcp-server{a} fai-doc{a} fai-quickstart fai-server{a}
  genisoimage{a} inetutils-inetd{a} nfs-kernel-server{a}
  openssh-server{a} syslinux-common{a} tftpd-hpa{a}
0 packages upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 2593kB of archives. After unpacking 8561kB will be used.
Do you want to continue? [Y/n/?]
```

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

---

[4]You can find the script in */usr/share/doc/fai-doc/examples/utils/*

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 (etch, lenny, 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.

These are important variables in *fai.conf* :

FAI_CONFIG_SRC

This variables described how to access the configuration space on the install clients. It's an Universal Resource Identifier (URI). Currently supported methods are:

- `nfs://host/path/to/exported/config` The config space is mounted from host via NFS.
- `cvs[+ssh]://user@host/path/to/cvsroot module[=tag]` The config space is received from a cvs checkout.
- `svn://user@host/svnpath` The config space checked out from a subversion repository. Also supported are svn+file, svn+http, svn+ssh, svn+https and checkouts without a user name.
- `git://host/path` The config space checked out from a git repository, host can be empty. Also supported is git+http.

If `$FAI\_CONFIG\_SRC` is undefined in *fai.conf*, then the default is to use an NFS mount from the fai install server onto the install client. It's the same as `nfs://`hostname`\`/$FAI_CONFIGDIR` with the host name determined on the install server. 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 file system. It will be mounted to `$MNTPOINT`, which must also be defined. It's not needed if you use access via FTP or HTTP.

A list of variables used by FAI can be found at http://wiki.fai-project.org/index.php/Variables.

The content of */etc/fai/apt/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 host names for each card (as for a Beowulf server), use the install server name which is known by the install clients.

FAI uses `debootstrap(8)` and `apt-get(8)` to create the nfsroot file system in */srv/fai/nfsroot*. It needs about 380 of free disk space. After editing *fai.conf* and *make-fai-nfsroot.conf* call `fai-setup`.

```
faiserver[~]# fai-setup -v
Using configuration files from /etc/fai
Creating FAI nfsroot in /srv/fai/nfsroot/live/filesystem.dir.
By default it needs more than 380 MBytes disk space.
This may take a long time.
Creating base system using debootstrap version 1.0.10lenny1
Calling debootstrap lenny /srv/fai/nfsroot/live/filesystem.dir http://cdn.debian.net/debian
.
Creating base.tgz
.
Upgrading /srv/fai/nfsroot/live/filesystem.dir
.
nfs-common fai-nfsroot module-init-tools ssh rdate lshw portmap rsync lftp less dump ←
    reiserfsprogs e2fsprogs usbutils hwinfo psmisc pciutils hdparm smartmontools parted ←
    mdadm lvm2 dnsutils ntpdate dosfstools jove xfsprogs xfsdump procinfo dialog discover ←
    console-tools console-common iproute udev subversion liblinux-lvm-perl cfengine2 libapt- ←
    pkg-perl grub lilo read-edid linux-image-486
install_packages: reading config files from directory /etc/fai
install_packages: read config file NFSROOT
.
```

```
.
'/etc/fai/NFSROOT' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/NFSROOT'
'/etc/fai/apt' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/apt'
'/etc/fai/apt/sources.list' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/apt/sources. ↩
    list'
'/etc/fai/fai.conf' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/fai.conf'
'/etc/fai/live.conf' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/live.conf'
'/etc/fai/make-fai-nfsroot.conf' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/make-fai- ↩
    nfsroot.conf'
'/etc/fai/menu.lst' -> '/srv/fai/nfsroot/live/filesystem.dir/etc/fai/menu.lst'
Shadow passwords are now on.
Removing 'local diversion of /usr/sbin/update-initramfs to /usr/sbin/update-initramfs. ↩
    distrib'
update-initramfs: Generating /boot/initrd.img-2.6.26-2-486
W: mdadm: unchecked configuration file: /etc/mdadm/mdadm.conf
W: mdadm: please read /usr/share/doc/mdadm/README.upgrading-2.5.3.gz .
W: mkconf: MD subsystem is not loaded, thus I cannot scan for arrays.
W: mdadm: failed to auto-generate temporary mdadm.conf file.
W: mdadm: no configuration file available.
'/srv/fai/nfsroot/live/filesystem.dir/boot/vmlinuz-2.6.26-2-486' -> '/srv/tftp/fai/vmlinuz ↩
    -2.6.26-2-486'
'/srv/fai/nfsroot/live/filesystem.dir/boot/initrd.img-2.6.26-2-486' -> '/srv/tftp/fai/ ↩
    initrd.img-2.6.26-2-486'
DHCP environment prepared. If you want to use it, you have to enable the dhcpd and the tftp ↩
    -hpa daemon.
Removing 'local diversion of /sbin/discover-modprobe to /sbin/discover-modprobe.distrib'
make-fai-nfsroot finished properly.    <=== *
No diversion 'any diversion of /sbin/discover-modprobe', none removed
Log file written to /var/log/fai/make-fai-nfsroot.log
Re-exporting directories for NFS kernel daemon....

   You have no FAI configuration space yet. Copy the simple examples with:
   cp -a /usr/share/doc/fai-doc/examples/simple/* /srv/fai/config
   Then change the configuration files to meet your local needs.
Please don't forget to fill out the FAI questionnaire after you've finished your project  ↩
    with FAI.

FAI setup finished.                    <=== *
Log file written to /var/log/fai/fai-setup.log
```

A complete example of *fai-setup.log* is available on the FAI web page. It's important that you find both lines that are marked with an asterisk in your output. 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)`. Adding *-v* gives you a more verbose output which may help you pinpoint the error. The output is written to */var/log/fai/make-fai-nfsroot.log*. It may help to enter the chroot environment manually using this command.

```
faiserver# chroot /srv/fai/nfsroot/live/filesystem.dir
```

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 the file */etc/netgroup* or in the corresponding NIS map. An example can be found in */usr/share/doc/fai-doc/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 the following command:

```
faiserver# /etc/init.d/nfs-kernel-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 */srv/tftp/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 [config]. Before you can set up a DHCP daemon, you should collect some network information of all your install clients. This is described in section [mac].

When you make changes to *fai.conf*, *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 or add additional packages into the nfsroot.

3.4   Troubleshooting the setup

The setup of FAI adds the *fai* account, exports file systems 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*.

4   Preparing booting

Before booting the client for the first time, you have to choose which medium you use for booting. Normally, you will configure the computer to boot via network card. The preferred method for booting is using PXE. PXE is the Preboot Execution Environment which most modern network cards support. Also booting from CD-ROM or from an USB stick is easy to set up.

4.1   Booting from network card with a PXE conforming boot ROM

Almost all modern bootable network cards support the PXE boot environment. Some network cards (e.g. on notebooks) 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:

```
faiserver# apt-get install isc-dhcp-server syslinux-common tftpd-hpa
```

Then set up the DHCP daemon. A sample configuration file can be found in */usr/share/doc/fai-doc/examples/etc/dhcpd.conf*. Copy this file to */etc/dhcp/dhcpd.conf*.

The install client then loads the pxelinux boot loader which receives its configuration via TFTP from a file in the directory */srv/tftp/fai/pxelinux.cfg* (defined by the variable $TFTPROOT in *make-fai-nfsroot.conf*). 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 detailed information.

4.2   Creating a boot floppy

If your network card can't boot by itself, you can create a small boot floppy that uses etherboot, which will provide the PXE feature for your network card. 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 if a special boot EPROM is inserted or booted from floppy provided by http://rom-o-matic.net. In depth documentation about booting via ethernet may be found at http://www.etherboot.org.

4.3   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 partial 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 http://fai-project.org/fai-cd.

4.4   Booting from USB stick

Using the command `fai-cd(8)` you can also create a bootable USB stick. First, format your stick with an ext2 file system (ext3 makes no sense on flash memory devices). Then mount it. After that call:

```
faiserver# fai-cd -m /path/to/mirror -u /path/to/mounted/stick
```

Then unmount the USB stick. The USB stick must be formatted with an ext2 file system. VFAT is not yet supported. Currently the file system that will be written onto the stick is not compressed.

4.5   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 host name 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:

```
faiserver# tcpdump -qtel 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:

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

After that, you only have to assign these MAC addresses to host names and IP addresses (*/etc/ethers* and */etc/hosts* or corresponding NIS maps). With this information you can configure your DHCP daemon (see the section [bootdhcp]). I recommend to write the MAC addresses (last three bytes will suffice if you have network cards from the same vendor) and the host name in the front of each chassis.

4.6   Configuration of the DHCP daemon

An example for `dhcpd.conf(5)` is available in */usr/share/doc/fai-doc/examples/etc/dhcpd.conf*, which is working with version 3.x of the DHCP daemon. Start using this example and look at all options used therein. The only FAI specific information inside this configuration file is to set *filename* to *fai/pxelinux.0* and to set *next-server* and *server-name*. All other information is only network related data, which is used in almost all DHCP configurations.

If you make any changes to the DHCP daemon configuration, you must restart the daemon.

```
# /etc/init.d/isc-dhcp-server restart
```

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.

4.7 Boot messages

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 6E 4A
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 3.71 (Debian-2008-09-06)  Copyright (C) 1994-2008 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: fai/
Trying to load pxelinux.cfg/01-00-04-75-74-6e-4a
Trying to load pxelinux.cfg/C0A801C0
Loading vmlinuz-2.6.26-2-486....................Ready.
Loading initrd.img-2.6.26-2-486................................
Ready
Uncompressing Linux... OK, booting the Kernel.
Linux version 2.6.26-2-486 (Debian 2.6.26-4)
.
.
Done.
Mounting root file system.....
eth0: link up
RPC: Registered upd transport module.
RPC: Registered tcp transport module.
aufs 20080714
.
.
Begin: Running /scripts/live-premount ... done.
IP-Config: eth0 hardware address 00:0c:29:c9:81:38 mtu 1500 DHCP RARP
IP-Config: eth0 guessed broadcast address 192.168.1.255
IP-Config: eth0 complete (from 192.168.1.250):
 address: 192.168.1.12   broadcast: 192.168.1.255  netmask: 255.255.255.0
 gateway: 192.168.1.254  dns0     : 192.168.1.250  dns1   : 192.168.8.9
 host   : demohost
 domain : informatik.uni-koeln.de
 rootserver: 192.168.1.250 rootpath:
 filename  : pxelinux.0
Begin: Trying netboot from 192.168.1.250:/srv/fai/nfsroot ...
Begin: Trying nfsmount -o nolock -o ro 192.168.1.250:/srv/fai/nfsroot /live/image ...
.
.
         -------------------------------------------------
              Fully Automatic Installation  -  FAI

              FAI 3.3, 03 Nov 2009   (c) 1999-2009
           Thomas Lange  <lange@informatik.uni-koeln.de>
         -------------------------------------------------
```

```
Calling task_confdir
Kernel parameters: initrd=initrd.img-2.6.26-2-486 ip=dhcp root=/dev/nfs nfsroot=/srv/fai/ ←
    nfsroot \
 boot=live FAI_FLAGS=verbose,sshd,createvt FAI_ACTION=install \
 FAI_CONFIG_SRC=nfs://kueppers/srv/fai/config BOOT_IMAGE=vmlinuz-2.6.26-2-486
Reading /tmp/fai/boot.log
FAI_FLAGS: verbose sshd createvt
FAI_CONFIG_SRC is set to nfs://kueppers/srv/fai/config
Configuration space kueppers:/srv/fai/config mounted to /var/lib/fai/config
Calling task_setup
Calling task_setup
FAI_FLAGS: verbose sshd createvt
Fri Oct 30 14:34:37 UTC 2009
30 Oct 14:34:37 ntpdate[3279]: step time server 134.95.4.129 offset 5.691554 sec
Press ctrl-c to interrupt FAI and to get a shell
Calling task_defclass
fai-class: Defining classes.
Executing /var/lib/fai/config/class/10-base-classes.
10-base-classes      OK.
Executing /var/lib/fai/config/class/20-hwdetect.source.
.
.
.
50-host-classes      OK.
List of all classes: DEFAULT LINUX I386 FAIBASE DHCPC DEMO GRUB demohost LAST
Calling task_defvar
Executing FAIBASE.var
++ FAI_ALLOW_UNSIGNED=1
++ CONSOLEFONT=
++ KEYMAP=us-latin1
++ UTC=yes
++ TIMEZONE=Europe/Berlin
++ ROOTPW='$1$kBnWcO.E$djxB128U7dMkrltJHPf6d1'
++ STOP_ON_ERROR=700
++ USE_SETUP_STORAGE=1
Loading keymap(s) us-latin1 ...done.
Calling task_action
FAI_ACTION: install
Performing FAI installation. All data may be overwritten!
```

When the copyright message of FAI is shown, the install client has mounted the nfsroot [5] to the clients' root directory. This is the whole file system for the client at this moment.

After *task_confdir* is executed, the configuration space is mounted via NFS.

Before the installation is started (`$FAI_ACTION=install`) the computer beeps three times. So, be careful when you hear three beeps but you do not want to perform an installation!

4.7.1 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.

```
Begin: Mounting root file system... ...
eth0: link up

BusyBox v1.10.2 (Debian 1:1.10.2-1) Built-in shell (ash)
Enter 'help' for a list of built-in commands.
/bin/sh: can't access tty: job control turned off
(initramfs)
```

---

[5]*/srv/fai/nfsroot* from the install server

You will get a shell prompt and can look at the log files, for examples */live.log* or */tmp/net-eth0.conf*. You can also enter the initramfs shell for debugging if you append the parameter *break* to the kernel command line.

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

```
$ showmount -e faiserver
```

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
```

If you get the following error message, the install kernel could not detect your network card, for example because of a missing driver:

```
    Begin: Mounting root file system... ...
    Kernel panic - not syncing: Attempted to kill init!
```

Check the initrd in the nfsroot if the kernel driver of your network card is included there.

4.8   Collecting other system information

Now the clients have booted with `$FAI_ACTION` set to *sysinfo*. Type *ctrl-c* to get a shell or use *Alt-F2* or *Alt-F3* and you will get another console terminal, if you have added *createvt* to `$FAI_FLAGS`.

Remote login is available via the secure shell if *sshd* is added to `$FAI_FLAGS`. The encrypted password is set with the variable `$FAI_ROOTPW` in *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 `$LOGSERVER` (if not defined: the server defined by `$SERVER` from *get-boot-info*) into the directory *~fai/demohost/sysinfo/*.

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.

FAI mounts all file systems it finds on the local disks read only. It also tells you on which partition a file */etc/fstab* exists. When only one file system 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              4099064     414088    3645296  11% /
udev                  10240         76      10164   1% /dev
192.168.1.250:/srv/fai/nfsroot
                    3905600     410976    3454944  11% /live/image
aufs                4099064     414088    3645296  11% /
tmpfs                193464          0     193416   0% /live
tmpfs                193464       3112     190352   2% /live/cow
```

```
faiserver:/srv/fai/config
                  3905600    410976   3454944   11% /var/lib/fai/config
/dev/sda1          241116     74519    154149   33% /target
/dev/sda9         4364212    139888   4179988    4% /target/home
/dev/sda7          553376     16840    536536    4% /target/tmp
/dev/sda8         2221628    275936   1832840   14% /target/usr
/dev/sda6          577096    172924    374856   32% /target/var
aufs               193464      2376    191243    2% /target/dev
```

This method can be used as a rescue environment! If you need a file system with read-write access use the `rwmount` command:

```
demohost# rwmount /target/home
```

4.9   Checking parameters from DHCP servers

If the install client boots with action *sysinfo*, you can also check if all information from the DHCP daemon 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 [setuproutines].

4.10   Rebooting the computer

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

5   Overview of the installation sequence

The following tasks are performed during an installation after the Linux kernel has booted on the install clients.

```
    confdir              # get config space
    setup                # early part of initialization
    defclass             # define classes
    defvar               # define variables
    action               # evaluate FAI_ACTION
    install              # Do the initial installation
    partition            # partition the harddisks, create file systems
    mountdisks           # mount the file systems
    extrbase             # extract the minimal base.tgz
    mirror               # get a Debian mirror via NFS
    debconf              # do Debian preseeding
    prepareapt           # preapre some apt-related files in target directory
    updatebase           # Debian specific
    HOOK instsoft.FAIBASE # fcopy kernel-img.conf
    instsoft             # install software packages
    configure            # call customization scripts
    finish               # do some cleanup, show installation time
    tests                # call tests if defined
    chboot               # call fai-chboot on the install server
    HOOK savelog.LAST    # grep for error messages in all log files
    savelog              # save log file to local dir and remote
    faiend               # reboot host, eject CD if needed
```

These are tasks, which are executed when a different action is performed

```
dirinstal              # install a chroot environment
softupdate             # do a system configuration without the partitioning part
```

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].

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 decrease the installation time considerably, so the network will not be the bottleneck when installing several clients simultaneously.

```
Intel Core2 Duo    ,    2GB,SATA disk,    3 GB software     14 min
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
```

5.1 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. There's also a graphical frontend available, called `faimond-gui(1)`.

5.2 Booting the kernel

The install client receives and loads the kernel and initial RAM disk. The kernel boots up and load the RAM disk. It does some hardware detection and then tries to figure where the root file system is located. When booting from network, this is determined by parameters from additional kernel parameters (*root=/dev/nfs* and *nfsroot=/srv/fai/nfsroot*). When booting from CD-ROM or USB stick the kernel and initial RAM disk probes removable devices and tries to figure out where the root file system is located. This may also be a compressed file system (using squashfs).

After the root file system is mounted read only, it is made writable by mounting a RAM disk via aufs (another unionfs) on top of it. So it's possible for programms or daemons to write to files inside a read only mounted file system. We are using the package `live-initramfs(7)` to mount the nfsroot and to make this file system writable using aufs. The package *live-initramfs* is only needed inside the nfsroot and adds some initramfs hooks.

5.3 Start and setup FAI

After the install client has booted, only the script */usr/sbin/fai* [6] is executed. This is the main script which controls the sequence of tasks for FAI. No other scripts in */etc/init.d/* are executed.

Additional parameters are received from the DHCP daemon and the configuration space is made available via the configured method (an NFS mount by default) 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.

The variable $FAI_CONFIG_SRC is used to get the FAI configuration space, which is very important, since FAI cannot proceed without the config space.

---

[6]Since the root file system on the clients is mounted via NFS, `fai` is located in */srv/fai/nfsroot/live/filesystem.dir/usr/sbin* on the install server.

5.4   Defining classes, variables and loading kernel modules

Now the script `fai-class(1)` is used to define classes. Therefore several scripts in *$FAI/class/* are executed to define classes. All scripts matching *[0-9][0-9]** (they start with two digits) are executed in alphabetical order. Every word that these scripts print to the standard output are interpreted as class names. Scripts ending in *.source* are sourced, so they can define new classes by adding these classes to the variable `$newclasses`.

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-doc/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 `20-hwdetect.source` loads kernel modules on demand. The complete description of all these scripts can be found in [cscripts].

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)`.

5.5   Partitioning local disks, creating file systems

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 file systems 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.

The old tool for partitioning the hard disks is called `setup_harddisks`. This tool is deprecated. With FAI 3.2.8 the new partitioning tool called `setup-storage(8)` was added to FAI. It uses `parted(8)` for editing the partition table and now has support for software RAID and LVM. This tool uses a slightly different format for the configuration files in *disk_config* than the old tool. Read the manual page for a detailed description of the new format. The variable *USE_SETUP_STORAGE* now determines which tool to use. When set to 1 it uses the new tool which is now defined in *FAIBASE.var* by default.

During the installation process all local file systems are mounted relative to */target*. For example */target/home* will become */home* in the new installed system.

5.6   Installing software packages

When local file systems 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 file systems. 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 [packageconfig].

5.7   Site specific customization

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.

More information about these scripts are described in [cscripts].

5.8 Automated tests

After the customization scripts are executed, FAI will execute some tests if available. Using these test, you can check for errors of the installation or of the softupdate. Test scripts are called via `fai-do-scripts(1)` and should append it's messages to *$LOGDIR/test.log*. A Perl module including some useful subroutines can be found in *Faitest.pm*. A test can also define a new class for executing another tests during next boot via the variable `$ADDCLASSES`.

5.9 Save log files

When all installation tasks are finished, the log files are written to */var/log/fai/$HOSTNAME/install/* [7] on the new system and to the account on the install server if `$LOGUSER` is defined in *fai.conf*. It is also possible to specify another host as log saving destination through the variable `$LOGSERVER`. If `$LOGSERVER` is not defined, FAI uses the variable `$SERVER` which is only defined during an initial installation (by get-boot-info). Make sure to set `$LOGSERVER` in a *class/*.var* script if you are using the action *softupdate*.

Additionally, two symlinks will be created to indicated the last directory written to. By default log files will be copied to the log server using scp.

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 *fai.conf*:

rsh
Use the rcp command to copy the log files to the log server.

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 `$LOGREMOTEDIR`. These variables are also defined in file *fai.conf*. You need write access for the `$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.

5.10 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 localboot.

6 Plan your installation, and FAI installs your plans

Before starting your installation, you should spend a lot of 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 reality. 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?

- What does my LAN topology look like?

---

[7]*/var/log/fai/localhost/install/* is a link to this directory.

- 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 file systems should be mounted?

- How should backups be performed?

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 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 called "Bootstrapping an Infrastructure" with more aspects of building an infrastructure is available at http://www.infrastructures.org/papers/bootstrap

7 Installation details

7.1 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 */srv/fai/config* and its subdirectories. This will be mounted by the install clients to */var/lib/fai/config*. It's also possible to receive all the configuration data from a `cvs(1)`, subversion (`svn(1)`) or Git (`git(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 file system creation.

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

package_config/
> Files with class names contain lists of software packages to be installed or removed. Files named **.asc* are added to the key list of apt.

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.

basefiles/

> Normally the file */var/tmp/base.tgz* is extracted on the install client after the new file systems are created and before package are installed. This is a minimal base image, created right after calling debootstrap during the make-fai-nfsroot process on the install server. If you want to install another distribution than the nfsroot is, you can put a tar file into the subdirectory *basefiles/* and name it after a class. Then the command `ftar(8)` is used to extract the tar file based on the classes defined. This is done in task *extrbase*.

hooks/

> Hooks are user defined programs or scripts, which are called during the installation process. The can extend or replace the default tasks.

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-doc/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/* /srv/fai/config
# chown -R fai /srv/fai/config
```

These files contain simple configuration for some example hosts. Depending on the host name 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 host names

> Hosts with other host name will most notably use the classes FAIBASE, DHCPC and GRUB.

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!

7.2 The default tasks

After the kernel has booted, it mounts the root file system 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/lib/fai/subroutines*. 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, listed in the order they are executed.

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 server and also additional variables are defined. The DNS resolver configuration file is created.

> The location of the configuration space is defined by the variable `$FAI_CONFIG_SRC`. You can use NFS, cvs, svn or git to access the configuration space. See section [isetup] for how to set the variable.

> 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* and the variable $ADDCLASSES.

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 */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/fai.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 [softupdate] for details.

partition

Calls setup-storage(8) to partition the hard disks and to create file systems. 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 a minimal system after that a chroot can be made into it. By default the base tar file */var/tmp/base.tgz* will be extracted. The command ftar -lv -s $FAI/basefiles / is used for unpacking a different tar file depending on classes defined. This can be used for installing different Linux distributions than the one used for creating the nfsroot. The default file *base.tgz* 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 files, 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.

tests

Calls test scripts in *$FAI/tests/* and its subdirectories for every defined class.

finish

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

chboot

Changes the PXE configuration for a host on the install server which indicates which kernel image to load on the next boot from network card via TFTP. Therefore the `fai-chboot(8)` command is executed remotely on the install server.

savelog

Saves log files to local disk and to the account `$LOGUSER` on `$LOGSERVER` (defaults to the install server).

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.

7.3 The setup routines of the install clients

After the subroutine `fai_init` has done some basic initialization (create RAM disk, 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 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"
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 *fai.conf*. When booting with PXE, command line parameters are created using `fai-chboot(8)`.

If you do not boot from network card but from CD-ROM or USB stick, you may also give network parameters to the kernel via the kernel command line. Two interesting parameters are

```
nfsroot=<server-ip>:]<root-dir>[,<nfs-options>]

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

Those parameters are described in the documentation of the Linux kernel sources in */usr/src/linux/Documentation/nfsroot.txt*.

7.3.1 FAI flags

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.

initial

Used by `setup-storage(8)`. Partitions marked with `preserve_reinstall` are preserved unless this flag ist set. Often, this flag is set in a file *class/*.var* by using setting *flag_initial=1*.

7.4 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 host name), 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 host name.

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 host name, 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-doc/classes_description.txt*.

host names 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.

7.5 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 host name 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 transferred 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.

```
10-base-classes:

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

[ -f /etc/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 XORG 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*.

7.6 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, define it in the script *LAST.var*.

FAI_ALLOW_UNSIGNED
    If set to 1, FAI allows the installation of packages from unsigned repositories.

CONSOLEFONT

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

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. You can create the encrypted password using `mkpasswd(1)`. See chapter various hints for more info.

UTC

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

TIMEZONE

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).

USE_SETUP_STORAGE

If set to one (the default when using the class *FAIBASE* the new partitioning tool `setup-storage(8)` will be used. Otherwise the old `setup_harddisks` program is used.

7.7 Hard disk configuration

Read the manual page of `setup-storage(8)` for a detailed description of the new format. This is used by default since FAI 3.2.8 or when setting the variable `$USE_SETUP_STORAGE` to 1.

7.8 Software package configuration

Before installing packages, fai will add the content of all files named *package_config/*.asc* to the list of apt keys. If your local repository is signed by your keyid AB12CD34 you can easily add this key, so fai will use it during installation. Use this command for creating the *.asc* file:

```
faiserver$ gpg -a --export AB12CD34 > /srv/fai/config/package_config/myrepo.asc
```

The script `install_packages(8)` 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 defines which command will be used to install the packages named after this command. The list of all available commands can be listed using *install_packages -H*. Supported package tools are: *aptitude, apt-get, smart, y2pmsh, yast, yum, urpm, rpm*

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. If you like to install packages from another release than the default, you can append the release name to the package name like in *openoffice.org/etch-backports*. You can also specify a certain version like *apt=0.3.1*. More information on these features are described in `aptitude(8)`.

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 want to remove a package name from a certain class was part of this class before, you should not remove the package name from the class file, but instead append a dash (-) to it. This will make sure that the package is remove during a softupdate on hosts which were installed using the old class definition which included this package name.

If you specify a package that does not exist this package will be removed from the installation list when the command *install* is used.

7.9 Customization 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 languages (shell, cfengine, Perl,..) for one class.

7.9.1 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.

7.9.2 Perl scripts

Currently no Perl script are used in the simple examples for modifying the system configuration.

7.9.3 Expect scripts

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

7.9.4 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 http://www.cfengine.org.

7.10 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.

7.11 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 several hooks for a task exists, they are called in the order defined by the classes. 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. A hook is an executable file following the naming scheme *taskname.CLASSNAME[.source]* (e.g. *partition.DISKLESS* or *partition.DISKLESS.source*), a task name and a class name separated by a dot, optionally followed by *.source*. The task name specifies which task to precede executing this hook, if the specified class is defined for the installing client. See section [tasks] for a complete list of default tasks that can be used.

In our 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 file systems 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 *$FAI_CONFIGDIR/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 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*.

7.12 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) and dhcpd3(8) :

```
faiserver$ egrep "tftpd|dhcpd" /var/log/*
```

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@fai-project.org if you have any questions. Sample log files from successfully installed computers are available on the FAI homepage.

7.13 Log files

FAI is creating several log files. During installation they are stored in */tmp/fai* on the install client itself. At the end of the installation they will be copied to the install server (see [isavelog]). After the install client rebooted into his newly installed system, you can find the FAI logs in */var/log/fai*. Log files are also created when doing the softupdate or dirinstall action.

These a some log files which are created by FAI.

FAI_CLASSES
> Contains a list of all classes defined.

dmesg.log
> Output of the `dmesg` command. Contains useful messages of the kernel ring buffer.

fai.log
> The main log file. Contains all important information. You should always read this file.

boot.log
> A list of variables of network parameters, mostly defined by the DHCP daemon.

format.log
> Output of the partition tool `setup-storage(8)`.

shell.log
> Output of all shell scripts, that are used for customization.

variables.log
> A list of all shell variables which are available during an installation.

error.log
> A summary of possible errors in all log files.

If the installation process finishes, the hook *savelog.LAST.source* 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 more details in all log files.

8 How to build a Beowulf cluster using FAI

This chapter describes the details about building a Beowulf cluster using Debian/GNU Linux and FAI. This chapter was written for FAI version 2.x for Debian woody and was not yet updated. The example configuration files were removed from the fai packages after FAI 2.8.4.

For more information about the Beowulf concept look at http://www.beowulf.org.

8.1 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 hard disk. 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.

8.2 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 22 GB space available.

8.2.1 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*.

8.2.2 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.

```
/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:

```
master# cd /var/yp; make
```

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

8.2.3 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 250 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)
```

8.2.4 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
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/d-hcp3/dhcpd.conf* and *make-fai-nfsroot.conf*. Replace the strings FAISERVER with *atom00* and uncomment the following line in *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"
```

8.2.5 Prepare network booting

Set up the install server daemon as described in [pxeboot]. If you will have many cluster nodes (more than about 10) and you will use *rsh* in *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 /srv/tftp/fai; chmod -R g+rwx /srv/tftp/fai
# cp /usr/share/doc/fai-doc/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.

8.3 Tools for Beowulf clusters

The following tools are useful for a Beowulf cluster:

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-doc/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 http://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 http://ganglia.sourceforge.net

8.4 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 `etherwake(8)`.

9 FAI on other architectures

If you want to use FAI on other architectures than i386 or amd64 you might need to take care of some things yourself.

These are things that may have to be changed on other architectures:

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

```
faiserver#make-fai-nfsroot -c /etc/fai-sarge
```

9.1 How to install i386 systems from an amd64 system

To install a computer with a 32bit i386 system, you need an i386 nfsroot. Creating this 32bit nfsroot on an install server runnning amd64 is quiet simple. Install and setup the fai packages. Then copy your fai config files to a new subdirectory.

```
faiserver# cp -a /etc/fai /etc/fai-i386
```

Edit the variable `$FAI_DEBOOTSTRAP_OPTS` in */etc/fai-i386/make-fai-nfsroot.conf* and add the option `--arch i386`. Also choose a different directory for your new nfsroot. Here are the two lines after editing.

```
NFSROOT=/srv/fai/nfsroot-i386
FAI_DEBOOTSTRAP_OPTS="--arch i386 --exclude=info"
```

Now call make-fai-nfsroot which creates the 32bit i386 nfsroot in */srv/fai/nfsroot-i386*

```
faiserver# make-fai-nfsroot -v -C/etc/fai-i386
```

Creating a partitial mirror using `fai-mirror(1)` that is needed for a bootable CD or USB stick is also possible on a different architecture. Due to a bug in apt-move (#441231), you have to specify the architecture when calling fai-mirror.

```
$ export MAXPACKAGES=800
$ fai-mirror -a i386 -v -cDEFAULT,FAIBASE,I386 /srv/mirror-i386
```

That's all!

9.2 FAI on PowerPC

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

9.3 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. This should not be need any more since the patitioning tool `setup-storage(8)` works on all architectures, were parted is supported.

9.4 Installing other distributions using a Debian nfsroot

You can install all sorts of Linux distributions from a single Debian nfsroot. Therefore you have to create a base.tgz of the distribution you like to install and place it info the `basefiles` directory. Then name it UBUNTU910.tar.gz for example. An install client which belongs to the class UBUNTU910 then extracts this base file into its empty file system. Additionally you have to adjust the *sources.list* or similar configuration files which are needed for specifying the location of the package repository.

9.5 FAI for Ubuntu, Suse, Redhat and Gentoo

All FAI packages are available in Ubuntu and are used by a large number of people since many version.

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@fai-project.org.

A brief description how to install SLES9 with FAI is available at http://www.sourcecode.de/install_sles_with_fai.

There are also some information in the faiwiki.

9.6 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. 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.

A success report as of 2001 is available at http://www.opossum.ch/fai/.

9.7 FAI for Solaris

FAI has also been ported for use with SUN Solaris OS installations in cooperation with Solaris jumpstart. This was done using FAI 2.8.4 and Solaris 9. Get the FAI sources from FAI 2.8.4 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 Solaris support was removed in FAI 2.9.

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

10 Advanced FAI

10.1 Creating chroot and virtualization environments

If you have some chroot environments to install, or a virtualization environment where you neither can nor want to run a normal Debian Installer in to get to a working system (for example, Xen guest domains), there is the FAI action *dirinstall*. By calling

```
faiserver# fai <options> dirinstall <target-directory>
```

and using either the option *-c <classes>* or *-N* you get a FAI installation, without the partitioning action, right into the target directory. The host name for the target installation can be specified using *-u <host-name>*

This, for example, can be used to combine FAI with the tool *xen-tools*, which helps you to build Xen guest domains. *xen-tools* are very nice for generating configuration files and block devices for new guests based on simple commands and/or configuration files, but they can only assign one role per installation for customization. FAI-users need and want more, as they are used to have the class system. They get them even in xen-tools installations, by using the following code as a xen-tools role script:

```
#!/bin/sh
TARGET=$1
CMD="fai -N -v -u ${hostname} dirinstall $TARGET"
echo running $CMD
$CMD
```

Then, you will want to set the variable *install=0* the xen-tools config for that host (in previous versions of xen-tools, this was *no-install=1*).

10.2   Using FAI for 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 re-installation. 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).

10.2.1   How does a softupdate work?

Softupdate 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 file system 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:

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.

10.2.2 How to run a softupdate

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

```
# fai -v softupdate
```

starts a softupdate. Make sure to set the variable $LOGSERVER (done in a *class/*.var* file) if fai should save the log files to a remote machine.

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 *fai softupdate* there, while the results can be seen immediately in the terminals started for each host.

10.2.3 How to write a configuration suitable for softupdate

When you want to do softupdate, 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 the command `ainsl(1)`. This has a similar function to cfengine's *AppendIfNoSuchLine* statement

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

  $FAI_CONFIG_SRC
  is the URI of the configuration space.

  $FAI_ROOT
  points to the client's rootdir. In case of softupdate it's the root directory */*

  $ROOTCMD
  contains a command for *chrooting* into the client. This is empty when doing softupdate (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 [8]

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

10.2.4 What if there are locally changed config files?

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, add following line to your *class/DEFAULT.var*:

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

10.2.5 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(8) or integrit(1) to scan for local changes and notify you about them.

11 Various hints

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

- To generate a md5 hash for the password use this *echo "yoursecrectpassword" | mkpasswd -Hmd5 -s*

- 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 remove the red logo on the install client by simply calling reset once. If will also not appear if you create a file using this command on the install server:

```
touch /srv/fai/nfsroot/live/filesystem.dir/.nocolorlogo
```

- If you like to define some additional classes (for e.g. A,B,C) on the kernel command line add this: *ADDCLASSES=A,B,C*

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

---

[8]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.

- 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 also delete the host entry on your install client in your *~/.ssh/known_hosts* file by using the *ssh-keygen -R* command.

- In the tasks chboot and savelog, a connection using secure shell is opened to the FAI server (see [isavelog]). To ensure that this works non-interactively, a proper entry in *NFSROOT/root/.ssh/known_hosts* must be created. When using fai-setup, this is done automatically, but it may require manual editing in case the name of your FAI server was not determined correctly. If you stumble over ssh connections that require typing "yes" to accept the host key during installation, please check the contents of your *NFSROOT/root/.ssh/known_hosts file*

- You can calculate the IP subnet address by using the nice tool ipcalc. Following example gives you the notation for a class C network (24) when the server network interface has the IP address 123.45.6.123

```
$ ipcalc -nb 123.45.6.123 24|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`.

- 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 /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.

- 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.

- How can I define classes on the kernel command line?

  Read the man page of `fai-class(8)`

- How to use a custom kernel inside the nfsroot?

  Build your customized kernel by building a kernel package using `make-kpkg(8)` and use the option `--initrd`. Copy this Debian package to a local repository and add it to /etc/fai/sources.list. Add the name of your package to /etc/fai/NFSROOT. Then rebuild the nfsroot.

- On http://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 softupdate (see [softupdate]).