The fully automatic installation of a Linux cluster (FAI)

Thomas Lange
Institut für Informatik
Universität zu Köln

E-mail: lange@informatik.uni-koeln.de

www.informatik.uni-koeln.de/ls_juenger/lange.html

September 8, 1999

Overview

- Motivation
- Hardware of our cluster
- Requirements and preliminary work
- Setting up the install server
- Booting methods for a client
- The installation process
- Defining classes
- Configuration with *cfengine*
- Start your own installation !

Motivation

Have you ever performed identical installations of an operating system several times?

Would you like to be able to install a Linux cluster with dozens of nodes single handedly?

- Repeating the same task time and again is boring
- Boring work surely will lead to mistakes and non-identical installations
- Automated installation guarantees identical installation on all hosts
- Much time can be saved
- Best case: Clusters, since they need equal installations on each node
- Experiences with automated installation of Solaris using Jumpstart and scripts from Casper Dik, which saves much time
- Short reinstallation time, after replacing damaged hard disk

Hardware equipment

Server: named lichtenstein

- Asus P2B-DS Mainboard
- 2 × Intel Pentium II 400 Mhz
- 512 MByte SDRAM (PC100)
- 3Com FastEtherlink XL, 10/100 Mbit, 3c905B
- Adaptec AIC-7890/1 Ultra2 SCSI host adapter
- 2 × 9 GByte harddisk
- CD-ROM, floppy disk, AGP graphic card

16 Clients: named roy01 to roy16, each equipped with

- Gigabyte 6BXD Mainboard
- 2 × Intel Pentium II 400 Mhz
- 256 MByte SDRAM (PC100)
- 3Com FastEtherlink XL, 10/100 Mbit, 3c905B
- 4,3 GByte harddisk, S3 graphic card,
- all clients share one keyboard and one monitor

Switch: 24 ports 10/100Mbit, Cisco Catalyst C2924-XL

Overall costs: 30.000 Euro (end of 1998)

Requirements and preliminary work

- Server with BOOTP, NFS services (default)
- TFTP daemon if booting from network card (default)
- Kernel image with `root=/dev/nfs` (existing)
- Root filesystem (easy)
- Access to Debian packages (currently via NFS)
- Configuration (your major task)
  How should a client be installed ?
  - partition table for local disks
  - mount information
  - software to be installed
  - **changes and supplements for the OS**
- NIS is not needed, but anyway useful
- An install client (client for short) that will be installed
  - it boots from floppy disk or network card
  - a keyboard and a monitor is not required
  - it can also be installed as a server

- Disk space

`/files/install/fai`	5 MB	configuration files, scripts
`/files/install/root`	30 MB	untar'ed base2_1.tgz
`/files/install/debian`	1200 MB	Debian version 2.1
`/usr`	100-300 MB	/usr of a Linux system

All install clients can share these directories, because they are mounted read-only.

- Of no matter if only one or hundreds of clients are involved
- FAI directory can be put onto a single floppy (two kernels can be created from other images)

Installation sequence

1. client boots from floppy or via network card
2. starts a fully functional Linux *without* using local disks
3. local harddisks are partitioned
4. empty filesystems are created
5. software is installed
6. changes to OS are made
7. clients reboot from local disk

Installation time for step 2–6

- client (50 MB): 2 minutes
- client (310 MB): 8 minutes
- checking 3.5 GB for bad blocks: 7 minutes
- installation time is mainly determined by the amount of software

At present, it is safer to reboot a second time using `/etc/rc2.d/S99finish_fai`.

Network setup

- All our clients share a class C subnet
- Sun Enterprise 450 is NIS server for all Unix hosts
- TFTP, BOOTP
  - Add to /etc/inetd.conf

```
tftp dgram udp wait nobody /usr/sbin/in.tftpd in.tftpd /tftpboot/
bootps dgram udp wait root /usr/sbin/bootpd bootpd -t 120
```

  - killall -v -HUP inetd
  - or kill -HUP <pid of process>
- Add ethers, hosts, netgroup entry for client

- NFS

```
# cat /etc/exports
/usr                     @linux-cluster(ro,no_root_squash)
/files/install/root      @linux-cluster(ro,no_root_squash)
/files/install/fai       @linux-cluster(ro,no_root_squash)
/files/install/debian    @linux-cluster(ro,no_root_squash)
```

  – /etc/init.d/nfs-server start
  – killall -v -HUP rpc.mountd

- Create `/files/install/root` with `create_client_root.sh`,

```
tar zxf base2_1.tgz
ln -s /proc/mounts etc/mtab
cp fai/scripts/rcS etc/init.d
```

- Create fai directory using: `tar zxf fai.tgz`

Structure of a Debian Distribution

```
lichtenstein[~]> tree /files/install/debian -d
files/install/debian
-- dists
 |-- Debian2.1r2 -> slink
 |-- slink
 |   |-- contrib
 |   |   |-- binary-all
 |   |   |   |-- admin
 |   |   |   |-- base
 |   |   |   `-- x11
 |   |   `-- binary-i386
 |   |       |-- admin
 |   |       |-- base
 |   |   |   `-- x11
 |   |-- main
 |   |   |-- binary-all
 |   |   |-- binary-i386
 |   |   |-- disks-i386
 |   |   |   |-- 2.1.9-1999-03-03
 |   |   |   `-- current -> 2.1.9-1999-03-03
 |   |   `-- upgrade-2.0-i386
 |   `-- non-free
 |       |-- binary-all
 |-- stable -> slink
```

BOOTP Configuration

/etc/bootptab:

```
.global.prof:\
        :ms=1024:\
        :sa=lichtenstein:\
        :hd=/tftpboot/:\
        :hn:bs=auto:\
        :rp=/files/install/root:\
        :sm=255.255.255.0:\
        :gw=134.95.9.254:\
        :ts=rubens:\
        :T170="134.95.9.100:/files/install/fai":\
        :T171="install":\
        :dn=informatik.uni-koeln.de:\
        :ds=134.95.9.136,134.95.100.209,134.95.100.208:\
        :ys=rubens:yd=informatik4711.YP:\
        :nt=time.rrz.uni-koeln.de,time2.rrz.uni-koeln.de:
# T170 is used for location of fai directory
# T171 "install" means do the installation, else execute a shell
roy01:ha=0x00105A270c08:bf=roy01:tc=.global.prof:
```

sa	TFTP server address
rp	Root path to mount as root
T170	generic tag. The location of the FAI directory.
T171	perform installation or execute shell
ts	Time server address list
dn	Domain name that is used in resolv.conf
ds	Domain name server address list
ys	Name of NIS server
yd	Name of NIS domain
nt	NTP (network time protocol) server list

Booting client

- booting with kernel 2.0.36
- from floppy: `dd if=bzImage.install of=/dev/fd0`
- special hardware needs compilation of new kernel
- from network card: set up TFTP and create boot floppy image; create a link using: `fai/scripts/ilink roy01`

```
lichtenstein[/]# ls -l /tftpboot
-r--r--r--  1 root  1475584 Aug 26 20:03 clusterimage
-r--r--r--  1 root  1475584 Aug 24 13:44 installimage
lrwxrwxrwx  1 fai        12 Aug 31 15:34 roy01 -> installimage
```

Boot messages without errors:

```
Memory: sized by int13 0e801h
Console: 16 point font, 400 scans
pcibios_init : BIOS32 Service Directory structure at 0x000fab60
Calibrating delay loop.. ok - 398.13 BogoMIPS
Memory: 256920k/262144k available (876k kernel code, 384k reserved, 3964k data)
Swansea University Computer Society NET3.035 for Linux 2.0
Checking 'hlt' instruction... Ok.
Linux version 2.0.36 (root@pittermaennche) (gcc version 2.7.2.3) #4 Fri Aug 20
 23:29:57 CEST 1999
Starting kswapd v 1.4.2.2
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
Ramdisk driver initialized : 16 ramdisks of 4096K size
ide: i82371 PIIX (Triton) on PCI bus 0 function 57
hda: WDC AC24300L, 4112MB w/256kB Cache, CHS=524/255/63, UDMA
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
eth0: 3Com 3c905B Cyclone 100baseTx at 0xe400, 00:10:5a:27:0b:29, IRQ 11
  8K byte-wide RAM 5:3 Rx:Tx split, autoselect/NWay Autonegotiation interface.
  Enabling bus-master transmits and whole-frame receives.
3c59x.c:v0.99E 5/12/98 Donald Becker http://cesdis.gsfc.nasa.gov/linux/drivers
 /vortex.html
Partition check:
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 hda9 hda10 hda11 >
Sending BOOTP requests..... OK
Root-NFS: Got BOOTP answer from 134.95.9.100, my address is 134.95.9.101
Root-NFS: Got file handle for /files/install/root via RPC
VFS: Mounted root (nfs filesystem).
```

Kernel ok, but BOOTP not enabled:

```
Sending BOOTP request.............. timed out!
```

Network card unknown, compile new kernel if:

```
Root-NFS unable to open at least one network device
```


Compiling new kernel

- no module support needed
- IP: Kernel level autoconfiguration / BOOTP
- Root file system on NFS
- ramdisk
- proc filesystem
- rtc (real time clock)
- do not enable initrd

```
mknod /dev/boot255 c 0 255
rdev bzImage /dev/boot255
dd if=bzImage of=/dev/fd0
```

Kernel for booting via network card using:

```
kernel2image.sh installimage bzImage /dev/nfs

kernel2image.sh clusterimage bzImage /dev/hda1
```

Installation process

- `/etc/init.d/rcS` is our FAI installation script
- initialize Linux
- setup FAI
- define classes
- format local disks
- install software packages
- call *cfengine*
- save log files
- reboot

fai_init

```
fai_init() {

    PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/sbin:/usr/local/bin:/fai/scripts
    export PATH
    umask 022

    mount -n -t proc proc /proc
    cat /proc/kmsg >/dev/tty4 &
    [ -x /sbin/update ] && update
    create_ramdisk /dev/ram0
    > /tmp/FAI_INSTALLATION_IN_PROGRESS
    trap 'exec sh' 2
    dmesg > /tmp/dmesg.log

    echo ""
    echo "$0: starting fully automatic installation FAI ..."
    echo "Press ctrl-c to interrupt installation process and to get a shell"

    # XXX TODO: if timeout for bootpc exit installation
    # define all bootpc information as variables
    bootpc | sed -e 's/^/export /' > /tmp/bootpc.log
    . /tmp/bootpc.log
    hostname $HOSTNAME
    # generic tag 170 (bootptab) used for location of fai directory
    export FAI_LOCATION=$T170

    if [ "$T171" != "install" ]; then
        echo /etc/bootptab: T171 != install. Not performing FAI installation.
        exec sh
    fi
}
```

- mount /proc
- create ramdisk and mount it to /tmp
- ctrl-c interrupts installation and executes a shell
- get BOOTP data via bootpc

bootpc.log

Data from /etc/bootptab is received by `/sbin/bootpc`

```
lichtenstein[~]# cat ~fai/roy01/bootpc.log
export SERVER='134.95.9.100'
export IPADDR='134.95.9.101'
export BOOTFILE='/tftpboot//roy01'
export NETMASK='255.255.255.0'
export NETWORK='134.95.9.0'
export BROADCAST='134.95.9.255'
export GATEWAYS_1='134.95.9.254'
export GATEWAYS='134.95.9.254'
export ROOT_PATH='/files/install/root'
export DNSSRVS_1='134.95.9.136'
export DNSSRVS_2='134.95.100.209'
export DNSSRVS_3='134.95.100.208'
export DNSSRVS='134.95.9.136 134.95.100.209 134.95.100.208'
export DOMAIN='informatik.uni-koeln.de'
export SEARCH='informatik.uni-koeln.de uni-koeln.de'
export YPSRVR_1='134.95.9.10'
export YPSRVR='134.95.9.10'
export YPDOMAIN='informatik4711.YP'
export TIMESRVS_1='134.95.9.10'
export TIMESRVS='134.95.9.10'
export NTPSRVS_1='134.95.100.209'
export NTPSRVS_2='134.95.170.8'
export NTPSRVS='134.95.100.209 134.95.170.8'
export HOSTNAME='roy01'
export T170='134.95.9.100:/files/install/fai'
export T171='install'
```

- rcS sources this file to define data as variables
- two forms of data: list and single items (_1, _2,...)

fai_setup

```
fai_setup() {

    mount -o ro $FAI_LOCATION /fai
    # read global config for fai
    if [ -r /fai/fai.conf ]; then
        echo  mounting FAI directory from $FAI_LOCATION
        . /fai/fai.conf
        echo $FAI_VERSION
        echo ""
    else
        echo mounting $FAI_LOCATION failed
        echo "or can't read /fai/fai.conf"
        echo "Can't start fully automatic installation."
        sh
    fi

    # after mounting /usr, we have everything needed
    mount -o ro -n -t nfs ${FAI_NFSSERVER}:/usr /usr &&
        echo /usr mounted from ${FAI_NFSSERVER}

    rdate ${TIMESRVS_1}
}
```

- mount /fai (BOOTP tag T170)
- read global fai.conf (define variables `FAI_`)
- mount /usr
- set time and date

define_classes

```
define_classes() {

    cd /fai/class

    # alphabetical sort is important
    for f in `ls S[0-9]*.{sh,pl,source}`
    do
        if [ -x $f ]; then
        echo executing $f

        case $f in

            *.pl)
            newclasses=`perl $f </dev/null`
            ;;

            *.sh)
            newclasses=`sh $f </dev/null`
            ;;

            # source files, which can set variables
            *.source)
            set -v
            . $f </dev/null
            set +v
            newclasses=
            ;;

        esac
        echo "$f: new classes= $newclasses"
        export classes="$classes $newclasses"
        fi
    done
}
```

- call `S[0-9]*.{sh,pl,source}` in alphabetical order
- the scripts print classes to standard output
- the files *.source only define variables, no classes
- classes are stored into /tmp/FAI_CLASSES and $classes

Partitioning disks

Partitioning local disks is done by `setup_harddisks.pl`

- reads the first file matching a class name
- writes new partition table to disk
- partition size can be an interval (1–200, 200–)
- creates empty filesystem by default
- optional parameters for mke2fs after ";"
- mounts filesystems relative to `$FAI_ROOT` according to mount points
- adds lines to /etc/fstab
- preserving partition size and data via `preserve<no>`
- preserving partition but create new filesystem via `preserve<no>` and `;format`

```
lichtenstein# cat 4GB
# disk configuration for one disk with 1000-4000kb

# <type> <mountpoint> <size in mb> [mount options]   [;extra options]

disk_config hda

primary   /         30        rw,errors=remount-ro ;-c
logical   swap      200       rw
logical   /var      30-200    rw
logical   /usr      70        rw
logical   /tmp      100-150                        ;-m 0
#logical  /scratch  0-        rw,nosuid            ;-m 0 -i 50000
logical   /scratch  preserve9 rw,nosuid            ;-m 0 -i 50000
```


Software installation

- `mount_packages.sh` mount the Debian distribution and extracts base2_1.tgz
- `install_packages.pl` read all config files in /fai/package_config
- installs selected software via `apt-get`
- `yes "" | dpkg --configure -a`
- apt-get is under development, new features will make this part more comfortable

```
lichtenstein[.../package_config]> cat ROY
PACKAGES install
netstd lpr pciutils sysutils time strace ldso
tcsh tcsh-i18n less cfengine
psmisc psutils
cron mpich

lichtenstein[.../package_config]> cat COMPILE
# packages for developing software
PACKAGES install
cpp bin86 binutils m4 make
libc6-dev libg++2.8.2 libstdc++2.9-dev
g++ gcc gdb libstdc++2.9
flex g77 byacc cvs
```

Main part of rcS

```
fai_init

( # execute in a subshell to get all output
fai_setup
define_classes

# partition local harddisks
setup_harddisks.pl  > /tmp/format.log 2>&1
. /tmp/disk_var.sh

# mount debian packages and install baseX_Y.tgz
mount_packages.sh

echo installing software may take a while
install_packages.pl > /tmp/software.log  2>&1

cd /fai/cfengine
for class in $classes
do
if [ -r $class ]; then
    echo "starting cfengine $class"
    cfengine --no-lock -v -f $class -D${cfclass} \
             >> /tmp/cfengine.log 2>&1
fi
done

chroot $FAI_ROOT hwclock --systohc
date
echo "installation completed."
rm -f /tmp/FAI_INSTALLATION_IN_PROGRESS
)  2>&1 | tee /tmp/rcS.log
```



```
if [ -f /tmp/FAI_INSTALLATION_IN_PROGRESS ] ; then
    echo Error while executing commands in subshell.
    echo /tmp/FAI_INSTALLATION_IN_PROGRESS was not removed.
    echo Please look at log files for errors.
    sh
fi

save_log

# now change boot device (local disk or network)
[ -n "$FAI_USER" ] &&
    rsh -l $FAI_USER ${SERVER} "cd /tftpboot/ ; rm $HOSTNAME;\
                                ln -s clusterimage $HOSTNAME"

if [ ! -f /tmp/REBOOT ] ;then
    echo "Press <RETURN> to reboot or ctrl-c to execute a shell"
    read
fi

echo "rebooting now"
cd /
sync
umount -a
exec /sbin/reboot -dfi
```

- during installation all log files are stored in /tmp
- save_log copies log files to `$FAI_LOG` (/var/log/fai)
- `$FAI_USER` also stores log files on the server
- new link in /tftpboot changes boot method
- clusterimage mounts / from local disk

Class concept

- all installation scripts use classes
- name of a class is written uppercase (except hostname) excluding: `- # .`   Use `[0-9A-Z_]`
- rarely use hostname for configuration files. Instead use a class and add the class to the client.
- all files whose names match a class name are used (or only the first)
- add a new configuration file without changing the script
- *cfengine* lacks this feature

```
for class in $classes
do
if [ -r $config_dir/$class ]; then
   <command> $config_dir/$class
   # exit, if only one is needed
fi
done
```

Different possibilities to define classes in /fai/class:

1. The name of the host is defined as a class.
2. Classes may be defined within a file.
3. Classes may be defined by scripts.

Scripts for defining classes

S00hostname.sh : Adds all classes that are stored in a file named as the client. Additionally adds the class with the hostname.

S01alias.sh : For all clients named roy01 to roy16, use the classes in file `ROY`.

S02memory.pl : Different classes are defined for different sizes of RAM. No yet used, for demonstration purpose only.

S03scsi.sh : If a SCSI device is attached, it adds the class *SCSI*.

S05network_card.pl : Depending on certain network cards, a class for this card is defined.

S07disk.pl : Defines classes depending on number of disks, their size or the overall disksize.

S24nis.sh : If a NIS domain is defined in `/etc/bootptab`, the class *NIS* and a class with the uppercase name of the NIS domain are added. Dots are replaced by underscores.

S88dataless.sh : Add class *DATALESS* for all *testclient??* except *testclient99*. This script is not used, but for demonstration purpose.

S90scratch.sh : If the disk layout defines a partition `/scratch` or `/files/scratch`, the classes *NFS_SERVER* and *SCRATCH* respectively *FILES_SCRATCH* are added. This script may uses classes that are defined in S07disk.pl.

S90tmp-partition.sh : If a separate partition `/tmp` exists, it adds the class *TMP_PARTITION*.

S99rootpw.source : Does not add a class, but defines the variable *rootpw*. The root password is mandatory.

S99var.source : Defines some variables for *cfengine*.

roy.classes : A file containing classes for all clients *roy??*. This file will be used by the script S01alias.sh.

faiserver : This file contains classes that are only used by client *faiserver*. It is used by S00hostname.sh.

S07disk.pl

```
#! /usr/bin/perl
# define classes for different disk configurations

# global variables:
# $numdisks              # number of disks
# %disksize {$device}    # size for each devie
# $sum_disk_size         # sum of all disksizes

require "fai.pl";

read_disk_info();

# rules for classes
#----------------------------------------------------------
# two SCSI disks 2-5 GB
($numdisks == 2) and
    disksize(sda,2000,5000) and
    disksize(sdb,2000,5000) and
    class("SD_2_5GB");

# one disk 1-4 GB
($numdisks == 1) and
    testsize($sum_disk_size,1000,4000) and
    class("4GB");

#----------------------------------------------------------
# do not edit beyond this line

exit;
```

```
30 # - - - - - - - - - - - - - - - - - - - - - - - - -
31 sub read_disk_info {
32     open ( DISK,"sfdisk -s|");
33     while (<DISK>) {
34        if (m!^/dev/(.+):\s+(\d+)!) {
35            my ($device,$size) = ($1,$2);
36            $numdisks++;
37            push @devicelist,$device;
38            $size /= 2048;# blocks -> Mbytes
39            $sum_disk_size += $size;
40            $disksize{$device} = $size;
41        }
42     }
43     close DISK;
44 }
45
46 sub disksize {
47
48     my ($disk,$lower,$upper) = @_;
49     testsize($disksize{$disk},$lower,$upper);
50 }
```

- fai.pl contains useful subroutines
- change only between lines 15 and 27
- subroutine *class*: print names of classes and exit
- subroutine *classes*: print names of classes without existing
- `(a) and class();` is like `if(a) then class()`


```
----------------------------------------
            S00hostname.sh

# add class $HOSTNAME
echo $HOSTNAME

# add classes defined in file $HOSTNAME
[ -f $HOSTNAME ] && cat $HOSTNAME
----------------------------------------
             S01alias.sh

# all roy's are using configuration ROY

case $HOSTNAME in
    roy??)
        cat ROY
        ;;
esac
----------------------------------------
            S03scsi.sh

# add class SCSI, if a SCSI adapter is available
if [ -e /proc/scsi/scsi ]; then
    grep -q "Attached devices: none" /proc/scsi/scsi && exit
    echo "SCSI"
fi
----------------------------------------
              S24nis.sh

# add NIS if YPDOMAIN is defined

if [ -n "$YPDOMAIN" ];then
   echo NIS
   echo $YPDOMAIN | tr '.a-z-' '_A-Z_'
fi
```

```
----------------------------------------
           S90tmp-partition.sh

# add class if /tmp has its own partition
for c in $classes
do
    if [ -r /fai/disk_config/$c ]
    then
        grep -v "^#" /fai/disk_config/$c | \
           grep -q '[[:space:]]/tmp[[:space:]]'  && \
           echo "TMP_PARTITION"
        exit
    fi
done
----------------------------------------
                S99rootpw.source

case $HOSTNAME in

    faiserver)
        rootpw="1bUwWgMxxxxxx"
        ;;

    roy??)
        rootpw="/NQ6jAn0xxxxx"
        ;;
esac
export rootpw
----------------------------------------
                  S99var.source

# these variabel are used by cfengine
export chroot=/usr/sbin/chroot
export kernelfile=/boot/vmlinuz
export cf_prefix="cfengine:"
export files=$FAI_FILES
export bserver=lichtenstein
export force=true
```



```
----------------------------------------
           S05network_card.pl

#! /usr/bin/perl

# define classes for different network card configurations

require "fai.pl";

@ethernet = read_ethernet_info();

# rules for classes
#---------------------------------------------------------
foreach (@ethernet) {
  classes("3C905B","100MBIT") if /3Com\s+3c905B/;
  classes("PCI_NE2000") if /PCI\s+NE2000/;
  classes("3C900") if /3Com 3c900/;
  classes("DS211403") if /Digital\s+DS211403/;
  classes("100MBIT") if /100baseTx/;
}
#---------------------------------------------------------
# do not edit beyond this line
exit;
# - - - - - - - - - - - - - - - - - - - - - - - - - - - -
sub read_ethernet_info {
  read_kernel_messages();
  return map { m!\beth\d+:(.+)!} (@dmesg,@messages);
}
```

Cfengine

- *cfengine* adjusts the installation to local requirements
- this part is normally manually done by the system administrator after the computer has booted for the first time

Examples:

- disable ftp daemon,
- set root password,
- configure DNS lookups,
- set up NIS,
- edit /etc/fstab,
- install special kernel and call lilo
- disable unused modules (eg. pcmcia)
- set up E-mail.

- *cfengine* also uses classes
- we pass classes via `-D${cfclass}`
- *cfengine* can make changes to a running system
- if a *cfengine* script is only run for a certain class, no classes need be used inside the script
- drawbacks:
  - all used variables have to be defined
  - all classes must be specified inside *cfengine*
  - no iteration over classes

- *cfengine* scripts are composed of sections

- section

```
action_type:
        class1::
             actions
        class2::
             actions
```

- some action types: `control`, `directories`, `files`, `tidy`, `links`, `editfiles`, `shellcommands`

- action editfiles has a rich set of commands

- by default copies are only made, if the "master file" is newer than the existing file

- compound class for logical operation: `CLASS1|CLASS2::` or `CLASS1.!CLASS2::` (currently few times used)

- predefined classes for system architecture and time classes (not used)

- classes can be defined by modules (not used)
- import not used because variable scope is unsuitable
- shellcommands need full path of the executables

Cfengine configuration

```
control:

   OutputPrefix = ("${cf_prefix}")

   actionsequence = (
                      tidy
                      files
                      copy
                      editfiles
                      links
                      shellcommands
                    )
tidy:
   any::
        ${target}/root/ pat=* R=10 age=0 rmdirs=true

   !TMP_PARTITION::
        ${target}/tmp  pat=*   age=0 rmdirs=true
        ${target}/     pat=tmp age=0 rmdirs=true
files:
   any::
        ${target}/tmp m=1777 o=root g=root action=fixall
        ${target}/etc/mailname m=644 o=root g=root action=touch

copy:
   any::
        /tmp/fstab      dest=${target}/etc/fstab
           m=644 o=root g=root

   KEYBOARD_GERMAN::
        ${files}/etc/kbd/KEYBOARD_GERMAN
              dest=${target}/etc/kbd/default.map.gz
           m=644 o=root g=root force=${force} backup=${backup}
```

```
links:
   any::
        ${target}/etc/localtime ->! /usr/share/zoneinfo/MET
           nofile=force

   !TMP_PARTITION::
        ${target}/tmp ->! /var/tmp        nofile=force

editfiles:
   any::
        { ${target}/etc/hostname
          AutoCreate
          EmptyEntireFilePlease
          Append        '${HOSTNAME}'
        }

        { ${target}/etc/hosts
          AutoCreate
          AppendIfNoSuchLine    '${IPADDR}${tab}${HOSTNAME}'
        }

        { ${target}/etc/passwd
          LocateLineMatching "^root:.*"
          InsertLine      "roott::0:0:root:/root:/usr/bin/tcsh"
          ReplaceAll      "^root::" With "root:${rootpw}:"
          ReplaceAll      "^roott::" With "roott:${rootpw}:"
        }

   HOME_CLIENT::
    { ${target}/etc/fstab
      AppendIfNoSuchLine "${hserver}:/home /home nfs rw,nosuid 0 0"
    }
   USR_MOUNT::
        { ${target}/etc/fstab
          HashCommentLinesContaining "/usr"
          AppendIfNoSuchLine "${bserver}:/usr /usr nfs ro 0 0"
        }
```



```
   USR_LOCAL_MOUNT::
    { ${target}/etc/fstab
     HashCommentLinesContaining "/usr"
     AppendIfNoSuchLine "${bserver}:/usr/local /usr/local nfs ro 0 0"
    }

shellcommands:
   NOPCMCIA::
        "${chroot} ${target} /usr/sbin/update-rc.d -f pcmcia remove"

   NOPPP::
        "${chroot} ${target} /usr/sbin/update-rc.d -f ppp remove"

   # create this tar-file on you binserver with:
   # tar -cf ROY -C /etc/alternatives .
   ROY.DATALESS::
        "/bin/tar -C ${target}/etc/alternatives -xvpf
             ${files}/etc/alternatives/ROY"

   USR_LOCAL_COPY::
        "/bin/cp -dpR /usr/local ${target}/usr"
```

```
# NONIS - for informatik.uni-koeln.de
control:

   OutputPrefix = ("${cf_prefix}")

   actionsequence = ( copy )

copy:
   NONIS::
        ${files}/etc/nsswitch.conf/NONIS
        dest=${target}/etc/nsswitch.conf
        m=644 o=root g=root force=${force} backup=${backup}
```

- `/fai/class/S99var.source` defines variables for *cfengine*
- master files for copy action are stored in /fai/cfengine/files/.... preserving the normal directory structure
- during installation *cfengine* copies and edits all files in $target
- for maintaining an OS with *cfengine*, set `$target=/`

Used classes

CF_BASE some base configurations
CF_BOOT copy kernel and modules and call lilo
CF_LAST remove old version of some files
CF_NETWORK configure network related parts like printer, xntp, network, inetd
COMPILE selects software packages for software development
KERNEL_SOFT installs kernel sources and kernel headers
KEYBOARD_GERMAN default.map for german keyboard
MINI_SOFT minimal software list
SOFT extensive software list
NIS configures system as NIS client
NONIS do not use NIS
ROY several little changes
TFTP_SERVER enable tftpd and copy clusterimage and installimage to /tftpboot
XNTP configures system to use NTP (Network Time Protocol)
4GB disk layout for one disk up to 4 GB
K2_2_10 kernel version 2.2.10, System.map and .config
KONGRESS1999 some special tasks for faiserver

NET_9 network related things that belongs to our class C subnet
USR_MOUNT mount /usr from `$bserver`
USR_LOCAL_MOUNT mount /usr/local from `$bserver`
USR_LOCAL_COPY make a copy of /usr/local to local filesystem
SCRATCH export /scratch to netgroup sundomain, linux-cluster
FILES_SCRATCH /export /files/scratch to netgroup @sundomain, @linux-cluster
FAISERVER export filesystem to netgroup fai
NOPCMCIA remove startup scripts for pcmcia
NOPPP remove startup scripts for ppp
3C905B ,NFS_SERVER not yet used

- if a *cfengine* script has the name of a class, we need not use classes inside the script (any::). This is because it is only used if this class is defined (eg. NIS, NONIS, TFTP_SERVER, KONGRESS1999

- scripts that starts with `CF_` are normally used for all clients

Tutorial

- server **faiserver** has already been set up
- clients: urmel01 .. urmel10
- each group has its own account: faitut1 .. 10 (no password)
- FAI directory /files/install/fai1..10
- FAI template is `~fai/fai.tgz`
- each group has one install client and a terminal to log onto the server
- bootptab, exports, ethers, netgroup are shared for all hosts, no NIS
- /files/install/debian is also identical for all groups
- login on faiserver via: rsh faiserver -l faitut1
- accounts have a tcsh (use tab to expand names, commands, files)
- emacs, less are installed
- only root can change bootptab on faiserver

Some hints

- bypass memory test by pressing ESC
- in the beginning /etc/bootptab has `T171="Xinstall"`
- start with few software packages !
- do not use class `REBOOT` during debugging process
- interrupt installation process by ctrl-c to look at the log files on /tmp
- look at log files on faiserver: `less ~fai/urmel01/*`
- read the *cfengine* manuals:
  `less ~fai/cfengine.txt.gz` or use
  "F1 i m cfengine Return" inside emacs
- *cfengine* errors: search for `error` or `no such`

Some tasks

- call updatedb (chroot $FAI_ROOT)

- install client as dataless

- install client as server (all data on local disk)

- create a script that defines class HOME_CLIENT, if no partition for /home is defined

- prepare X11