UNIVERSITY OF OSLO

DEPARTMENT OF PHYSICS

REPORT SERIES

A unix configuration engine

Mark Burgess

Department of Physics
University of Oslo, Box 1048
N-0316 Oslo 3, Norway

UIO/PHYS/94-11
ISSN-0332-5571

Received: 1994-06-01

A unix configuration engine

Mark Burgess

June 1, 1994

Abstract

A high level description language is presented for the purpose of automatically configuring large heterogeneous networked unix environments, based on class-oriented abstractions. The configuration engine is portable and easily extensible.

Preface

The effective management of large unix clusters is an increasingly pertinent problem. A number of vendor specific tools are available for performing routine administrative tasks, but these still require an administrator to manually program a configuration. Few tools exist for defining global parameters at a high level. The configuration engine (*cfengine*) is a high level description language which defines actions to be performed on the basis of class membership. A single file or file-set is prepared, containing a description of an entire network; this file-set is then made available to every machine on the network. At predetermined times, *cfengine* extracts information relevant to each machine and configures the relevant parameters without the need for human intervention.

In its present form, cfengine deals with the following issues: (i) mounting and unmounting of network file systems (global and local resources), (ii) symbolic link management, (iii) file checking, ownership, permissions with warning or correction, (iv) network interface configuration, (v) tidying of junk files from system disk areas and user areas, (vi) controlled script execution and (vii) simple file editing functions.

The language itself is a standard left-right algebra implemented using the yacc parser and the lex lexer. The syntax is based on a list structure. The design of the language part is essentially class based. A given system in a network is characterized by a number of attributes: its hostname, its operating system type, its membership in user defined groups, the day of the week etc. The sum of these attributes answers a yes or no question for the configuration engine. A statement has the form

```
<compound-class> = ( list of actions )
```

If the compound class evaluates to TRUE, the list of actions is stored for later execution. The compound class is TRUE if each of the classes in the compound class is true. For example,

```
sun4.mygroup.Monday = ( ... )
```

evaluates to TRUE if the host parsing the instruction is of type sun4, belongs to 'mygroup' and today is Monday.

A number of variables are additionally maintained by the configuration engine. These are described in more detail in the following documentation. These variables such as,

```
<host>
<timezone>
<binserver>
<site>
<domain>
<nfstype>
```

to name a few, characterize a given host and can be used to construct abstract commands which encompass many machines in a single statement. This is the principal advantage of the configuration engine.

The configuration engine has been in use at the University of Oslo for about one and a half years at the time of writing and has proved to be an invaluable tool. Its development has not ceased and will no doubt continue until such a time as it is surpassed by a rival utility.

cfengine is available by anonymous `ftp` from `aurora.uio.no` and is self-configuring on all supported architectures.

CONFIGURATION ENGINE V2.2.15

A network unix configuration tool

Mark Burgess

June 1, 1994

Theory Group
Institute of Physics, University of Oslo
P.O. BOX 1048, Blindern, 0316 OSLO 3, NORWAY

SYNOPSIS

cfengine -sw1 -sw2 ... [-f filename]

SUMMARY

A very high level description language for the configuration of UNIX machines on a tcp/ip network. The setup of all machines is defined centrally from one file, or a single set of files in a class oriented fashion. Designed for portability. Run uid root.
Valid switches are:

-h Help information. Display version banner and options summary.

-l Normally *cfengine* does not follow symbolic links when recursively parsing directories. This option will force it to do so.

-v Verbose mode. Prints detailed information about actions and state.

-n No action. Only print what has to be done without actually doing it.

-i Do not attempt to configure the local area network interface.

-f Parse filename after this switch By default *cfengine* looks for a file called *cfengine.conf* in the current directory

-d Enable debugging output. Normally you will want to send this to a file! -d1 shows only parsing output. -d2 shows only action output. -d shows both levels.

-p Parse file and then stop. Used for checking the syntax of a program. You do not have to be superuser to use this option.

-m Do not attempt to mount file systems or edit the filesystem table.

-c Do not check file systems for o /nership / permissions etc.

-C Check mount points for consistency. If this option is specified then directories which lie in the "mount point" area are checked to see whether there is anything mounted on them. Normally this is *off* since not all machines use mounted file systems in the same way. e.g. HPUX does not generally operate with partitions, but nevertheless one might wish to mimick a partition-like environment there, but it would he irritating to be informed that nothing was mounted on the mount point.

-t Do not tidy file systems.

-s Do not execute scripts.

-a Print only the name of the system administrator then quit.

-V Print only the version string and then quit.

-D Define a compound class symbol of the form *alpha.beta.gamma.*

-N Cancel a compound class, or defined with value *false* a compound class of the form *alpha.beta.gamma.*

-L Delete links which do not point to existing files (except in user home directories, which are not touched).

Known Bugs

The only known bugs are in tbe parser. The absence of a space before or after parentheses will sometimes cause the parser to complain of a syntax error. Under links: if there is no space between the arrow and the first pathname, no error will be given but the arrow is assumed to be part of the pathname. Caution with spaces!

Obtaining cfengine

A compressed tar file containing the complete package may be obtained by anonymous ftp from `aurora.uio.no`.

Contents

1 Introduction

To use *cfengine* you create a single file[1] which describes the setup of all machines in a machine-park. The interpreter program "*cfengine*" is compiled on all machines. The program file is distributed to all machines and each one executes the same file. The relevant information is extracted from the file by the interpreter and used to configure each machine individually.

The operation of *cfengine* is essentially class orientated. A program or configuration file consists of the specification of a number of classes together with a number of actions which are to be carried out for each class. For instance, we might want to make a link from /usr/spool to /var/spool on all HP machines. The easiest way to do that would be to define the symbolic link for all machines which own the class "hpux". Any HP machine *running* the program would satisfy the requirements and the link would be made.

In general, actions specified in the program file are performed if the machine which is executing the program file is determined to be a member of the class for which an action is specified. A number of basic system variables are required such as the domain and and the subnet mask on the local area network segment. You choose what types of action are carried out by *cfengine* within the scope of the permitted functionality. You have control over what actions will take place and for which files, machines, groups and the order in which each class of event takes place.

cfengine is intended first and foremost to be run as a batch or shell job, perhaps daily though there is no limit as to how often it can be run. If run in silent mode (the default) then no output is generated if there is nothing wrong. It is therefore natural to pipe any output to some kind of wrapper program which mails it to the system administrator. See appendix C for an example. It is also possible to use the switches $-D$ and $-N$ to select only certain subsets of commands for execution. Thus a *cfengine* program can be adapted to many different situations.

2 Hints and Tips

It is useful to adopt some standard conventions and practices when using it.

- It is assumed that the *system possesses* a working domain name service which is configured by a file called /etc/resolv.conf.
- Although it is *not* an assumption made by *cfengine*, it makes sense to to mount file systems according to the convention

  ```
  /<site>/<machine>/filesystem
  ```

  This has serveral advantages, not least of which is for the sake of being systematic The actual path may be specified in the two variables *mountpath* and *homepat* *See the section 'localinfo' for more remarks about this*

[1] It is possible to break up this file into smaller logical pieces by using the *import* function

- In a heterogeneous networked environment, there will be many different conventions for placing the mail directory. *Cfengine* stores the operating system's assumed location for mail internally and will attempt to mount mail there, but it is probably a good idea to make links on all machines so that the user (who is caught in the middle of the bizarre conventions used by UNIX manufacturers) has some idea of where he/she is. For instance /usr/mail, /usr/spool/mail, /var/mail, /var/spool/mail are all possibilities. Perhaps all or some of these should be linked together.

3 An overview of the possibilities

Here is an overview of some tasks which can easily be administered by *cfengine*.

- Management of protection and ownership of files on any filesystem.
- Monitoring of basic security issues: hosts.equiv, shells, setuid root programs.
- Autoconfiguration of the local area network device interface, netmask and broadcast addresses, as in ifconfig.
- A symbolic link manager for installing and maintaining symbolic links.
- Shared filesystem (NFS mount) manager for installing home partitions and binary services from a server.
- Automatic update of /etc/fstab (or equivalent) with predefined filesystems.
- Class-controlled execution of user scripts.

The more advanced features include:

- Checking and optional fixing of permissions and ownership in file systems, according to lists of allowed users/groups. The lists may include netgroups.
- Linking all of the files in one directory to another directory in such a way that a mirror image of the directory is maintained in new location. For example: the command `/local/bin +> /local/perl/bin` would link all of the children of the directory /local/perl/bin so that they would appear to be in the directory /local/bin. As new files are added to /local/perl/bin, new links will be automatically made.
- Tidying up garbage files such as "core" files which are older than a certain number of days

4 Log Files

cfengine stores two types of logs. The first of these is a system log which is stored in /etc/*cfengine*.log. This log contains a list of all known setuid root and set gid root programs and it used to detect the appearence of new programs which may be a security risk. It also stores a separate log for each user on the system (/.*cfengine*.rm) which contains a list of all the files which were deleted by the tidy function during the last pass. The log is stored in the home directory of each user and gets overwritten each time *cfengine* runs to completion. A lock file is stored in /etc/cfengine.pid to ensure that two copies of *cfengine* will not be invoked simultaneously.

5 Invocation

cfengine may be invoked in a number of ways. Here are some examples:

```
% cfengine
% cfengine -f myfile
% cfengine -f myfile -v -n
% cfengine -d
```

The first (default) command looks for a file called `cfengine.conf` in the current directory and executes the commands silently. The second command reads the file `myfile` and works silently. The third works in verbose mode and the -n option means that no actions should actually be carried out, only warnings should be printed.

It is advisable to check all programs with the -n option before trusting them to the system, at least until you are familar with the behaviour of *cfengine*.

There is, in addition to the above, a -l mode which forces *cfengine* to follow symbolic link references. The default is not to follow symbolic links.

The complete list of options is listed in the summary at the beginning of this manual, or you can see it by giving the -h option.

6 Compiling

To compile cfengine you will need: cc, yacc and lex on your system. Edit the Makefile giving the correct options for your system. gcc -traditional may be substituted for cc for versions after 2.2.2 (which are bug ridden and ill-advised). bison and flex may be used instead of yacc and lex with a suitable modification of the Makefile. (yy.tab.c is replaced by prog.tab.c) etc. Note that the solaris include files arriving with the gcc distribution contain an error in sys/dirent.h in the definition of d_name, so this may cause problems I haven't seen yet

It is assumed that a working resolver exists on the system The package *resolv+* works well for sun workstations

7 Program Structure

7.1 Classes

It is easy to write programs for *cfengine*. Look at the example program in Appendix A for more help. A program consists of a number of declarations of the form.

```
action:

class = ( bracketlist... )
class1.class2.class3 = ( list2 ... )
```

The form of the program is free. Use of space is unrestricted, though sometimes the parser will be confused if there is not a space before or after a bracket. A *class* is defined to be one of the following:

- The name of an architecture (a h rd class) *e.g.* ds, solaris
- The hostname of a particular host *e.g.* boson, gollum, terminator
- The name of a group, as defined in groups (soft class), *e.g.* sysadm, servers
- An arbitrary string specified using the $-D$ or $-N$ options.

Classes must always have the syntax given above: a *list of strings connected by dots*. A so-called 'compound' class has the value of *true* if *all* of its members are defined classes. i.e. It is the logical AND of its members. If a single member fails to match for a particular host, then the compound class has the value *false*.

The command option $-N$ may be used to force certain symbols to be false, and thus there is a lot of freedom to define classes and subclasses which apply in different situations.

7.2 Comments

Comments in a *cfengine* program are specified by the number symbol # as in shell programming and apply for the rest of the current line.

7.3 Actions, Reserved words and Hard Classes

Actions are reserved words from the following list:

```
localdefs
broadcast
groups
import
shells
```

```
resolve
defaultroute
makepath
misc_mounts
files
ignore
tidy
homeservers
binservers
mailserver
required
mountables
links
disable
shellcommands
editfiles
```

Any hard class is also a reserved word.

```
ultrix
sun4
sun3
hpux
aix
solaris
osf
irix
linux
```

Two hard classes are always defined by the language itself, namely the name of the current host and its operating system architecture. These are determined internally by system calls to *uname(2)*. Thus a declaration of the form

```
mymachine = ( list )
```

will always be carried out on the machine "mymachine", and

```
solaris = ( list )
```

will always apply to machines which run solaris. In addition the day of the week may also be given as a class, provided it is capitalized in the following way.

```
Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

For instance

```
shellcommands:

   any.Sunday = ( weeklyscript )
```

7.4 Generic class " any"

The generic wildcard "any" may be used to replace any group or machine. Thus instead of assigning actions for the class sun4 only, one might define actions for any architecture by specifying:

```
any      = ( list ... ) # any machine
any.sun3 = ( list ... ) # any sun3 machine
```

7.5 Special class "exclude"

The special class exclude is also always defined by default. It's purpose is to allow certain 'excludable actions' to be defined. Specifically the syntax

```
any.exclude = ( ... )
```

will normally be carried out, unless *cfengine* is run in a restricted mode. By defining the symbol 'exclude' to be false one can therefore exclude all of the actions which include 'exclude' as a member.

```
cfengine -Nexclude          # Run cfengine in restricted mode
```

This would normally be used to omit certain time-consuming actions, such as tidying all home directories.

7.6 Variables with special meaning

Certain listed items may contain references to the following variables

`<faculty>` The faculty as defined in localdefs

`<site>` This variable is identical to `<faculty>` and may be used interchangeably.

`<binserver>` The default server for binary data.

`<host>` The hostname of the machine running the program

<sysadm> The name or mail address of the system administrator.

<timezone> The current timezone.

<domain> The currently defined domain.

These variables are provided in order to encourage the definition of fully generalized pathnames. The judicious use of these variables can reduced many definitions to a single one, with some thought.

You can use these variables in the following places:

* In any pathname. The <binserver> variable is not always appropriate in this context. For instance (See under binservers for more information)...

```
links:

osf = ( /<site>/<host>/directory -> somefile )
```

* In any quoted string. (See "Shell Commands")

```
shellcommands:

any = ( "/bin/echo <timezone> | /bin/mail <sysadm>" )
```

The latter possibility enables *cfengines* variables to be passed on the user-defined scripts.

The redefinition of any of these variables usually leads to an error. The exception is the <sysadm> variable which may be redefined. The means that the mail address for the system administrator can be specified for each group individually. Note that the -a option can be used to print out the mail address of the system administrator for any wrapper scripts.

7.7 Anulling entries when debugging

A useful trick when debugging is to eliminate unwanted actions by changing their class name. Since *cfengine* assumes that any class it does not understand is the name of some host, it will simply ignore entries it does not recognize. For example:

```
myclass = ( ... )
```

can be changed to

```
Xmyclass = ( ... )
```

Since Xmyclass no longer matches any defined classes, and is not the name of any host it will simply be ignored. The *N* option can also be used to the same effect.

8 Actions

The actions performed by *cfengine* are defined using the following contructions. (Look at the example in the appendix for more examples.) A detailed description of the what is done at run-time and how it may be customized is given in the next section.

8.1 Localdefs

Here we define some general information for the subnet in which this file applies. Here is an example

```
localdefs:

   faculty = ( mn )                    # or: site = ( mn )
   domain    = ( uio.no )
   sysadm    = ( drift@fys.uio.no )
   netmask   = ( 255.255.254.0 )
   timezone  = ( MET )
   nfstype   = ( afs )

   sensiblesize  = ( 1000 )
   sensiblecount = ( 2 )
   editfilesize  = ( 100 )

   actionsequence =
      (
      mountrescources
      makepaths
      links
      editfiles
      mailcheck
      mountall
      required
      tidy
      disable
      files
      shellcommands
      )

   mountpath = ( /<faculty>/<host> )
   homepat   = ( u? )                  # e.g. <faculty>-home-<host>? u1 u2 )
```

The faculty variable[2] is used in the construction of generic filenames. This information is copied directly into the variable <faculty> or <site> which may be used later in the definition of symbolic links and required file systems. The domain information is used in the resolver configuration. sysadm is intended to be the mail address to which mail will be sent. *cfengine* does not currently use this however. The netmask and broadcast addresses must be specified in the decimal network byte order form shown. The timezone should be in upper case, provisionally.

mountpath is the pathname under which it is assumed that all mounted file systems belonging to the host lie. (It is strongly advisable to mount nfs files systems in one place and link them if necessary.) In this case, if the hostname were "zaphod" it would expand to /mn/zaphod.

homepat is a list of names for home directories. Usually this will just be something like u1, u2 etc. This name is added to mountpath in order to obtain the absolute path to u er home directories. Wildcards may be specified as well as generic variables with the exception of binserver. The last example above would expand to mn-home-zaphod?, a perverse choice though, of course, possible. The question mark matches a single character.

Users mounting homedirectories like /home/machine are advised to group together all file systems under some logical scheme like the one above (/faculty/machine/filesystem) and to make links to /home/machine if they really must, rather than setting mountpath to /home/machine since cfengine checks for all mounted filesystems under mountpath. This feature would then be useless.

The nfstype variable defines the string to be sent to the mount command for mounting network filesystems. The default value is "nfs", but this could be "afs" for the Andrew file system or anything else which appears in the future.

It is illegal to define more than one element in all the above lists with the exception of *homepat*.

8.2 Groups

Groups are soft classes. There is no limit to the number of groups which may be defined. Groups must be defined before they are referred to. An idenifier which has not been defined as a group is assumed to be a hostname.

A given host may appear in as many groups as desired. Remember that groups are classes and should therefore be used to group machines according to their function. For example: you might define the following:

```
servers     = ( anyon fidibus hope )
suncluster = ( anyon boson fermion semion gluon gluino)
ypslaves    = ( anyon fidibus )
```

[2]The word 'site' may be used synonymously with 'faculty'

The machine anyon appears in all three groups, but this is not a problem. This may be used to advantage. It simply means that the machine anyon inherits the soft classes "servers", "suncluster" and "ypslaves", and thus any actions which pertain to those classes will be carried out for host "anyon".

Groups are defined using the list syntax. The list members may be the names of machines or may refer to netgroups. The NIS + or +@ notation is used to signify a netgroup. For example

```
physics = ( +@physics-sun4-hosts roger bambi thumper +NISothers )
usering = ( bilbo mordor gollum )
```

Note that there is no penalty for using netgroups here, regardless of the size of the netgroups. The members of the group are never stored, for it is sufficient to know that the host name is one of the members in the list. When the host is found to belong to the list on the right hand side of a group declaration, it inherits the class on the left hand side which results in the storage of only one item.

The '-' symbol may be used to make exceptions from a list. This is useful if you have defined a netgroup of many machines but a particular rule should not apply to one or two members of the group.

```
exception = ( +@my-net-group -somemachine -@sub-group )
```

This syntax means that the group 'exception' consists of all the machines in the netgroup *my-net-group* minus the host 'somemachine' and minus all the members contained in the netgroup 'sub-group'. Note that unlike the case of the '+' symbol, the '@' symbol here changes the meaning from a single entry to a netgroup.

Be warned that if an action is specified for a group which is not defined, *cfengine* will assume that the name refers directly to the name of a host, so that misspellt words will simply be ignored since they will never match either a group or a hostname.

See also the sections on `homeservers` and `binservers`. Note: *There is no need to separate machines into different architectures at this stage; this is done later by defining* `binservers`. cfengine *will only match the correct architecture, in architecture dependent actions. The machines in the groups do not even have to belong to the current domain, but may be machines which the current domain will receive services from, perhaps in the form of NFS file systems.*

8.3 Import

Further files can be imported into the root file using the import facility. This might be used to group together a global file for say all ultrix machines. Programs can therefore 'inherit' the definitions made in other files The root file is special however

```
import:
```

```
sun4 = ( sun4.globaldefs sun4.localdefs )
any  = ( extra_stuff )
```

and cannot be replaced, only supplemented by the additional files. Imported files are read in after the main file.

8.4 Broadcast

Every cfengine.conf file must specify the local convention for forming broadcast addresses. There are two choices: either with "ones" or with "zeroes". If zeroes is specified

```
any = ( zeroes )  # or zeros, if you spell it like that
```

then the broadcast address is formed according to the rule

```
broadcast = IP address & netmask
```

If ones is specified, it is formed by

```
broadcast = IP address | ~netmask
```

The broadcast address is specfied per machine/group/class.

8.5 Defaultroute

Dynamical routing is not configurable in *cfengine*, but for machines which have static routing tables, it is useful to check that the default (wildcard INADDR_ANY) is configured so as to point to the nearest router or gateway. The syntax for this is simply:

```
class = (
        mygateway
        )
```

or

```
most = (
       129.240.22.1 # address for my gateway
       )

rest = (
       small_gw      # hostname small_gw
       )
```

This checking is optional. The effects of this command may be seen by looking at the static routing tables using *netstat -r*. This command does the same as the BSD shell command *route(1)*. It is equivalent to

```
route add default mygateway 1
```

There is a problem with the *ioctl(2)* commands which manipulate the routing tables, namely that the default route may be assigned more than once so any number of different addresses. Moreover, once an address has been assigned, it is not easily removed again. *cfengine* therefore checks before adding anything that the present setup is okay. Only if the default route is missing will it be added. It is not possible to fix badly assigned routing since it requires "routing" in the kernel tables – something which I am not willing to do (at least for the time being). The command *route -f* will flush the gateway references from the routing tables if need be and then you can either fix things manually or run *cfengine* one more time to add the default route.

8.6 Shells

The file `/etc/shells` is generated from the lists given here. Note that since the issue of allowed shells is, at least in principle, a security issue, this file gets generated from scratch each time the program runs. It simply copies the list of shells for a given host machine into the file. If no shells are defined the file will be erased! The generic class "any" is probably appropriate here. Example:

```
any = (
      /bin/csh
      /bin/sh
      /local/bin/tcsh
      )
```

The file is set to mode 0644.

8.7 Resolve

The /etc/resolv.conf file specifies the default nameserver and the default domain. These are currently the only two types of information that *cfengine* cares about. Specifying a list of nameservers (by decimal IP address), for instance

```
any = (
      129.240.22.35
      129.240.22.18
      129.240.2.3
      )
```

results in a file which looks like

```
domain uio.no
nameserver 129.240.22.35
nameserver 129.240.22.18
```

```
nameserver 129.240.2.3
...
```

That is to say, any line starting with "domain..." gets overwritten by the default domain. If any of the addresses in the list exist they are moved to the head of the list according to the ordering specified. Any other lines are left alone. The editing algorithm will delete lines which begin or end in spaces or blank characters since the resolver cannot understand them anyway, but does not complain. The file is set to 0644.

Different zones in the network could easily be programmed to use a different default nameserver by defining appropriate groups for different zones.

8.8 Mailserver

This is used to look for a mail spool area such as /usr/spool/mail. Only one mailserver may be defined per group. For example:

```
most = ( hphost:/usr/mail )
some = ( ulrik:/usr/spool/mail )
```

cfengine 'knows' the official name of the mail spool directory under different operating systems and tries to mount a remote mail directory at that location. So if we were on a hpux machine which was a member of the group 'some', *cfengine* would try to mount /usr/spool/mail on host 'ulrik' onto /usr/mail locally. If the mail directory is mounted somewhere else, a warning is generated in verbose mode (with the -v flag) but otherwise nothing is done to change the situation.

8.9 Homeservers

A list of machines which have disk space for users' home directories.

```
group = ( machine1 machine2 machine3 ... )
```

The hosts do not necessarily belong to the users domain. *cfengine* uses this list to find out what file systems a given machine needs to mount. Private groups can therefore be defined as desired. A file system is deemed mountable if it comes from a machine which is in a list which matches the acceptable classes for the current host. That is, in the example above. If host is a member of 'group' it will mount all the homedirectories on machine1, machine2, machine3 etc

8.10 Binservers

A list of machines which have disk space containing binary directories. Binary directories include substitutes and additions to /usr, /usr/local, or any architecture specific information. For example

```
group.sun4 = ( mysun )
```

The hosts need not necessarily belong to the users domain. *cfengine* uses the binserver list to find out what file systems (other than homedirs) a given machine should mount. The list is also used in the definition of the special variable <binserver>.

8.11 Mountables

This specifies a straightforward list of all the mountable rescources which exist, including the server it is found on. The items do not belong to any particular class. Mountables are a global commodity. *cfengine* only mounts directories which are specified in this list. Home-directories are automatically identified as long as the 'homepat' variable is defined. Example:

```
mountables:

   (
   anyon:/mn/anyon/u1
   anyon:/mn/anyon/u2
   anyon:/mn/anyon/u3
   anyon:/mn/anyon/local
   anyon:/mn/anyon/fys
   gluino:/mn/gluino/pc
   fidibus:/mn/fidibus/u1
   fidibus:/mn/fidibus/u2
   fidibus:/mn/fidibus/local
   fidibus:/mn/fidibus/fys
   tema:/mn/tema/u1
   )
```

Note that since mountables are parsed in order, you need to place default binservers with highest priority first. If *cfengine* has been instructed to link a particular binary file to some default binary server (for example by some path which includes the variable <binserver>) it will pick the first one that matches. If you want to give priority to a particular binary server, make sure it comes earlier in the list than any other binary servers. If there is only one binary server for each mountable then this matter of priority should never be a problem. If however there were two sun4 machines in the list which both exported a file system called /faculty/<hostname>/local, then only the first of these would be used

The mountables list is used in conjunction with the homeserver and binserver lists in order to determine what file systems should be mounted on a host.

8.12 Misc mounts: Miscellaneous mounts

Quite apart from filesystems associated with home directories or binary services, there might be directories which you simply want to mount even though they don't fit into

those categories, for instance a central library server, or an information database. The `misc_mounts:` directive may be used for this. For example:

```
misc_mounts:

  any = (
        infohost:/path/info /local/info ro
        )
```

This command would mount the directory /path/info on the host infohost onto the local directory /local/info, read only. If /local/info doesn't exist it is created. The `rw` option clearly specifies read-write mounting.

8.13 Unmount: unmount old filesystems

This function removes entries from */etc/fstab* etc, attempts to unmount the filesystem and removes the directory on which the filesystem is mounted.

```
unmount:

  any = (
        infohost:/old/filesystem
        )
```

If the device is busy, the actual unmount will not take place until it becomes available, or the machine is rebooted. Despite the peculiarities of the AIX operating system, this should feature still work there!

8.14 Makepath

Makepath declarations consists of a number of directories to be created. An entire path may be created in a single item.

Read the section about links carefully: Unix exhibits some peculiarities concerning the creation of links which makes the order in which directories are created important

Directories may also be created using the "touch" option to the "files" action by specifying a directory using the notation "/pathname/ " to indicate that the file is a directory and not a plain file. See files.

The makepath list is a straightforward list of absolute pathnames

```
mymachine = (
            /arfle/barfle/gloop
            )
```

Directories are created with the default permission 0755 with ownership root. If necessary this may be altered later using the "files" action. The special variables <host> and <faculty> may be used here.

Note: the creation of a path will fail if *cfengine* finds that one of the links in the pathname was a plain file.

8.15 Links

Read this section carefully. The order in which links are defined will affect the way in which directories are created. There are peculiarities in the way in which Unix makes symbolic links.

The link manager is a case where the special variable <binservar> may be used. When a link is defined it is first checked and made if necessary. When referring to symbolic links the format is

```
class = ( from_link -> to_file)
```

which creates a pointer from the file "from" to the already existing file "to".

If the "from" link lies in a non-existent directory, then the directory will be created. That is, if we want a link from */a/b/c* to a file *d* then the directories *a* and *b* will be created, if they do not already exist. Directories are created with the default mode of 0755 and ownership root. An error may occur if part of path to the link is blocked by the presence of a real file i.e. not a directory. *NOTE: this feature requires some caution to be exercised. See the note below.*

The "to" path may contain special variables. For example:

```
any = ( /local -> /<faculty>/<binserver>/local )
```

in which case *cfengine* looks for a file system which matches from the list of binservers, starting with the name of the current host. If the link already exists but points somewhere else and warning is issued. If the link exists and is correct, nothing is done.

There is a peculiarity associated with symbolic links. Consider the following shell commands

```
mkdir /ant
ln -s /mydir /ant
```

These commands do not fail with an error but result in a link being made between /ant/ant -> /mydir This is a feature/bug which is built into the unix call symlink() and thus it also applies to *cfengine*. This has some implications for link making. *Because cfengine makes the directories leading up to the* from *link* the following two definitions result in different link structures

```
mygroup = (
          /local/math -> /mn/gluon/local/math
          /local -> /mn/anyon/local
          )
```

and

```
  mygroup = (
            /local -> /mn/anyon/local
            /local/math -> /mn/gluon/local/math
            )
```

In the former case, the directory /local does not exist and is therefore created by *cfengine* in order to be able to make the link. Since /local then exists and is a directory, the second link results in /local/local->/mn/anyon/local and not what was actually specified. The second example yields the correct result.

8.16 Extended Links

cfengine enables the definition of multiple links from a single command. Consider the notation:

```
  /path +> /varpath
```

This is interpreted to mean: link all the children (files and directories) of /varpath to files with corresponding names in /path. /varpath may contain the special variables <binserver>, <host> and <faculty>. The left hand side of the assignment may contain all the variables except <binserver>.

8.17 Links: linkchildren

The linkchildren feature is an extension of multiple links. The linkchildren construction may be specified in two places: (i) as a "link" action or (ii) as a "files" action. If specified as a link action it takes the form:

```
   /my/file/path +> linkchildren
```

This syntax has the following meaning: make links of all of the children in a corresponding file system on a binary server. That is: first look for a binary server in the list of mountables which matches the path name in some way. If such a server is found, link all the files on that server to images in the directory /my/file/path

For example: we might like to have

```
   /local -> /mn/<binserver>/local
```

but actually have a real directory /local with some but not all objects required already filled in. The following

```
/local/priss
/local/lib
```

are real directories, but we want to fill in a full /local filesystem from some server, i.e. we want to link any files that do not exist to mirror images on the server.

Once a suitable server has been found, *cfengine* descends into the directory (one level only) and links any files that don't exist on the host. In order for a server file system to match an item in "mountables", the following conditions must be met.

(i) The potential file system must belong to an allowed binserver

(ii) The directory name (the last element in the path on the host) must exist as an element in the path of the potential item in "mountables".

so that

```
/local
```

on a sun would match

```
/mn/anyon/local      (sun)
```

but not

```
/mn/fidibus/local    (dec)
/mn/anyon/fys        (sun)
```

`linkchildren` does *not* take any notice of home directories in the list of mountables. The process of locating a server works by reverse parsing the name declared in files. For instance

```
group = (
        /local/lib/emacs +> linkchildren
```

would start a search for a mounted filesystem ending in emacs, then for one ending in lib, then for one ending in local until a match was found. A match is only made for filesystems which reside on valid "binservers" . Binservers are chosen sequentially from the list defined above (not including the current host). Once a valid file system has been identified, *cfengine* checks that the files or required directories exist on the file system and tries to link them. If it turns out that it cannot link the files for some reason (files don't exist) then it continues looking for other alternatives and eventually gives up when there are no more binservers left, generating the message:

```
cfengine: Couldn't link the children of <file> to anything because no
          file system was found to mirror in the defined binservers list.
```

This feature has proven useful on client machines which have their own /local file system, but which have a natural server in addition. Normally one would simply mount the server onto the client. If the client has disk space and is host to some special software, then it is convenient to define a real directory /local on the client, containing the special software and link the remaining files. Example: server anyon has a file system /mn/anyon/local which it exports for all clients. Host gluon has its own filesystem /mn/gluon/local which contains the Mathematica program, because it is the only machine which is lisenced for that package – also it has some spare disk space. We therefore define:

```
gluon = ( /mn/gluon/local +> linkchildren)
```

The file system /mn/gluon/local then contains the package mathematica and links to all of the packages on the server machine anyon. (It is assumed that anyon is correctly defined as a binserver for gluon.)

8.18 Disable: renaming dangerous files

Plain files (not directories) may be disabled by placing them in a list here. The files are moved (renamed) by adding the suffix "*.cfengine*-disabled".

```
group = (
        /etc/hosts.equiv
        )

disable:

  some = ( /var/spool/cron/at.allow )

files:

  some = ( /var/spool/cron/at.allow =0644 N [root] [wheel] touch )
```

8.19 Files: *permissions and ownership*

The file checking utility is one of the more powerful features of *cfengine*. The idea is to specify a filesystem or simple file which is to be checked for permission, ownership and group ownership An action is also specified which is then carried out if the file(s) do(es) not fit the specified criterea. This is most easily seen in an example

```
group = (
        /local/lib -2002 R [root,daemon] [daemon,sys] fixall
        )
```

This means that for machines in "group", we should check the files lying in /local/lib. The R means that the checking should extend recursively into all subdirectories. The first square bracket is a list of the allowed owners of files. This list may either contain textual names or numerical user IDs. The second square bracket represents group ownership, where the same rules apply. The final token is a reserved word which means that *cfengine* should fix the problem without reporting an error.

What happens is the following. The UID and GID of the file are determined and compared to the owner lists. If there is a mismatch, the UID or GID or both are set to the first name appearing in the list. In addition the file's access modes are checked against the template -2002. The minus sign means that none of the marked flags (specified in the normal chmod octal notation) should exist. *cfengine* unsets them. If we had used a plus + sign, then *cfengine* would ensure that those flags were set. An equals sign = means set to the absolute mode. Thus in the example, we are checking that no files are writable to the world and that no files are setuid root. If you want to make sure that some flags are present AND that some are not, then the +- notation can be used. For instance

```
/fys/drift                    -0002+0770 R [root] [andrift] fixall
```

Directories: because directories must be executable in order to be readable, cfengine will always add an execute flag to a permission if the corresonding read flag is set. Thus setting a whole file system to mode = 644 would set all directories to = 755 and all plain files to = 644.

8.20 Recursion

The recursion specifier may have the following values: R=<number>, R or N. The optional number may be used to specify a maximum number of recursion levels. *cfengine* will not descend more levels than the given number. The option N is equivalent to R=0, thus if the file to be checked is a directory and the specifier is N or R=0, then only the directory itself will be checked. If R=1 then the children in that directory will also be checked, but not the children of any subdirectories. (*Note: the same recursive specifiers are also available for the* `tidy:` *function.*)

8.21 Home directories

The special directive "home" may be used instead of an absolute file path. If this variable is used, *cfengine* will automatically iterate over all home directories on the current machine. If none are found then no error is generated; it is thus straightforward to make a generic check of all home directories.

```
any =
   (
   home        -2002 R [*] [an] fixall
   )
```

8.22 files: setuid and setgid root

Files which are setuid or setgid root are logged in the system log file /usr/spool/*cfengine*.log and the appearence of any new files results in a warning message. The exception is if the files are marked "fix" then a warning if issued but the name is not stored.

8.23 files: options

The syntax for files is as follows:

```
group = (
        /pathname [+-=]0000 [N|R=<number>] '[' UID list ']' '[' GID list ']'
                                                (fix*|warn|touch|linkchildren)
        )
```

0000 is a four digit octal number.

A uid/gid list is specified inside square brackets with commas between the members

e.g. `[root,mark,8245,10987]`

e.g. `[*]`

`+` add flags

`-` remove flags

`=` absolute permission

The wildcard * may be used instead to match any UID or GID. This is useful for parsing user areas for instance. Note that even though no specific UID or GID is specified, this does not compromise the checking of setuid root files. If "fix" is selected and the first member of the file list is a wildcard the ownership of the file is not changed, but the file is touched. The uidlist may contain references to netgroups defined in the Network Information Service, using the + and +@ notation.

`[root,+admin]`

Netgroups are not allowed in the gidlist. Allowed file actions are

```
warnall
warndirs
warnplain
fixall
fixplain
fixdirs
touch
linkchildren
```

The types offer the possibility to single out either directories or plain files. If warn* is specified, only a warning is issued and nothing more is done to files. If fixall is specified any problem is immediately corrected regardless of the file type. Plain (regular) files or directories may be singled out by using fixplain and fixdirs respectively (see also the note below concerning directories). If the target is a file or directory and the action is touch then the object will be created and the permission and ownership set to the specified values. If only + or - are specified then objects are created with the default mode of 755 before adding or removing the named flags. If an object exists its access times will be updated. The special option linkchildren is covered below.

Directories: because directories must be executable in order to be readable, cfengine will always add an execute flag to a permission if the corresonding read flag is set. Thus setting a whole file system to mode = 644 would set all directories to = 755 and all plain files to = 644.

8.24 touching directories

A directory or path name may be specifiifed by the following line

```
/pathname/. [+-=]xxxx [N|R] [UID] [GID] touch
```

The trailing dot tells *cfengine* that you want to create a directory. All directories up to the final one are created using the standard modes and then the final link in the path is changed to the mode/owner as specified in the UID and GID parts. The recursive function has no meaning here since a newly created directory has no children to parse, but syntactically something must be specified.

8.25 uid, gid = −1

When *cfengine* says it is changing the uid or gid of a file to −1 it means that it is not changing it at all. −1 is used as an internal representation of a wildcard. This enables either the uid or gid but not both to be changed easily within the context of the program.

8.26 Files: linkchildren

The linkchildren feature is discussed in detail under "links" above. The syntax here is

```
group = (
        /local/lib/emacs +0000 N [blueeyes] [cataract]  linkchildren
        )
```

The links are made in the usual way, but now the ownership is changed directly to the values given here. In this case the owner of the links would be *blueeyes* and the group would be *cataract*. In most circumstances the ownership root will be appropriate

8.27 Ignore: omitting filesystems from recursive file checks

Ignore items are used to explicitly exempt files from checking when recursively parsing directories for files. Files which are specified by direct pathname are not checked against this list of files, thus particular files may be checked even though they reside in a directory which is marked "ignore". The items in this list may be either floating wildcards or pathnames. For example:

```
any = (
      !*
      /local/lib/gnu/emacs/lock
      )
```

8.28 Tidy: removing files

The tidy function is used to delete (remove permanently) unwanted files from a system. The form of an entry is

```
tidy:

machine = ( /<path>  <wildcard> <recursion-spec> <age> )
```

Path is the directory point at which *cfengine* should start searching for files. Wildcard is a pattern which should match a particular file or set of files (i.e. a filename in the simplest instance) The recursion specifier is the same as that described under the `files:` action. The age of the file is the minimum age after which the file is to be deleted. Here are some examples:

```
tidy:

   allhservers = (
                 home                    core            R       >0
                 home                    *%              R       >3
                 home                    \#*             R       >1
                 /tmp/                   *               R       >1
                 /etc                    printcap.old    N       >0
                 /                       core            R=2     >0
                 )
```

In the first example thus searches for all instances of (plain) files named "core" in home-directories which are older than zero days i e core files will always be deleted

The special variable home causes cfengine to search through all home directories and subdirectories R is the only option allowed if home is specified It is not possible to restrict the search to a finite number of levels (The search ends when it reaches either the deepest subdirectory, a symbolic link or an NFS mounted filesystem) When searching a home

directory (which is assumed to belong to the user area <mountpath>/<homepat>) cfengine will match even files that begin with a dot. Normally dotted directories are not searched on recursive descent. This is to allow certain programs such as window managers to hide files in /tmp/.X11-like directories. The removal of such files could result in loss of window data about the open display.

The backslash character is ignored if it is the first character of a wildcard. This enables you to write:

```
home \#* R >1
```

The hash symbol would otherwise be ignored as a comment line.

Note: elements of a pathname may NOT be wildcards. cfengine will not delete directories.

8.29 Required: filesystems with special importance

This check is performed after *cfengine* has mounted all the filesystems it thinks are missing on a given host. Files marked here are deemed to be important for the operation of the host. If they do not exist then a warning is generated. The special variables <binserver>, <host> and <faculty> may be used in these items. The latter two are substituted directly from from variables defined in localinfo and from the information determined from the host. The variable <binserver> is expanded according the list of defined binary servers, starting with the host name. This allows a kind of wildcard notation. If the destination does not exist, *cfengine* chooses the next binserver in the list until it finds a match. If no match is found there is an error. For example:

```
required:

   any =     (
             /<faculty>/<binserver>/local
             )

   teorfys = ( /mn/<binserver>/pc )
```

Some checking is also performed on the files here to see if they are sensible. These checks are somewhat arbitrary and are controlled by the constants

```
sensiblecount
sensiblesize
```

If a required directory exists but has fewer than *sensiblecount* files in it, a warning is issued Similarly, if a required file exists but is smaller than *sensiblesize* bytes, a warning is issued These variables may be set in *localdefs*

8.30 Shell commands

The scripts or shell commands listed in this section are executed at the end of the program execution. Any shell commands may be given here. Tbe quoted lines are passed directly to the shell (normally */bin/sh*). The syntax is as follows:

```
shellcommands:

  class = ( list )
```

For example:

```
  any =    (
           "/local/etc/config/use.daily"
           "/fys/mutils/bin/noseyparker /mn/anyon/u1 nomail"
           )

  sun4 = (  "/usr/lib/find/updatedb" )
```

8.31 Editfiles

This function is experimental and may change in a later release. The idea is to allow simple editing actions on text files. Since this is a potentially dangerous function, some checks are made. First of all, it is only permitted to edit regular files. Textfiles which are bigger than the user-defined variable 'editfilesize' are not be edited.

The syntax is

```
editfiles:

class =
   ( /path/filename

   actionname  "string....."
   actionname  "string....."
   )
```

The list of actions is

```
DeleteLinesStarting
DeleteLinesContaining
AppendIfNoSuchLine
PrependIfNoSuchLine
```

These are self explanatory?

9 Runtime Behaviour: actionsequence

The special list 'actionsequence' is used to *define the order and the number of times in* which the different actions are executed. The list is defined under localdefs and consists of the following keywords.

```
      makepaths  # Make directories.
          links  # Check and build all links (simple and child).
    simplelinks  # Check and build simple links.
     childlinks  # Check and build multiple links.
      mailcheck  # Check for mail directory and edit filesystem list
       required  # Same as required:
           tidy  # Same as tidy:
 shellcommands   # Same as shellcommands:
          files  # Same as files:
        disable  # Same as disable:
mountrescources  # Edit the file system list to add new mounts
       mountall  # Mount all previously defined file systems
      editfiles  # Edit a simple text file
        unmount  # unmount mounted filesystems
```

and the list is specified with the syntax:

```
   actionsequence =
      (
      unmount
      mountrescources
      makepaths
      links
      editfiles
      mailcheck
      mountall
      required
      tidy
      disable
      files
      shellcommands
      )
```

Any keyword may appear any number of times in the list

Before this list is executed *cfengine* checks the network interface and the timezone for consistency, configuring the interface if necessary It then scans the mount list for a list of mounted filesystems

10 Bootstrap script: cf.preconf

Certain systems can reach a kind of deadlock situation whereby cfengine cannot parse its configuration file because it does not have a properly configured network interface, and can therefore never get far enough to fix the problem itself. To solve this theoretical situtation, a shell script "cf.preconf" is searched for prior to parsing the configuration file. This enables any prerequisites to be dealt with. A typical use of this would be to ensure that a completely fresh machine had access to NIS databases which get referred to in 'groups:'. 'cf.preconf' may be any script in any script langauge. It is fed one argument which is the *system hardclass pertaining* to the machine (e.g. `ultrix`) so that this can be tested for in the script if necessary.

11 Portability

cfengine was designed to combat the problems of writing scripts which would work on a variety of systems. It is designed to be portable. A minimal amount of OS information is stored in the program itself in order to deal with system specific matters. In a later version it may be worth reading this in from configuration files to make it more easily extensible . This storage of information should be completely transparent to the end user, but will be *of interest in porting* the language to other platforms. The currently supported platforms are

soft class	Op. Sys.	Architecture	Release
sun4	sunos	sun 4?	4.1.*
sun3	sunos	sun3	4.1.*
ultrix	ultrix	risc	4*
hpux	hp-ux	9000*	*
aix	aix	*	2
linux	linux	i?86	4*
irix	irix	ip22	5*
osf	osf1	alpha	*

The interpreter has not been tested on *osf* or *hnux* platforms (no machines are available!)

A Example program

Here is a substantial example program to show the principles.

```
#######################################################################################
#
#
#  CFENGINE CONFIGURATION FOR site = mn
#
#  For V 2.2 or later
#
#
#  Mark 31. Jan. 1994
#
#
#######################################################################################

localdefs:

   faculty    = ( mn )
   domain     = ( uio.no )
   sysadm     = ( drift@fys.uio.no,meh@usit.uio.no )
   netmask    = ( 255.255.254.0 )
   timezone   = ( MET )
   nfstype    = ( nfs )

   sensiblesize  = ( 1000 )
   sensiblecount = ( 2 )
   editfilesize  = ( 4000 )

   actionsequence =
      (
      unmount
      shellcommands
      editfiles
      mountresources
      makepaths
      links
      mailcheck
      mountall
      required
      tidy
      disable
      files
      )
```

```
    mountpath = ( /<faculty>/<host> )
    homepat   = ( u? )

#########################################################################################

groups:

    fys_teori           = ( +@fysikk-sun4-hosts -tema -aurora -mephisto -arsen )
    fys_strukt          = ( tema )
    fys_plasma          = ( aurora bjoern )
    fys_fast            = ( gran sir hassel rogn alm linn selje
                                    vidje einer misteltein )
    fys_elg             = ( wolfram tin indium silver arsen )
    fys_kjerne          = ( isak hope )

    AllHomeServers      = ( anyon fidibus hope tema aurora hassel isak linn )
    AllBinaryServers    = ( anyon mephisto fidibus hope gran alm )

    packages            = ( anyon hope gran alm fidibus )

    fys_supersparc      = ( mephisto vakuum )
    fys_felles          = ( hope fidibus mephisto )
    fys_alle            = ( +fysikk-hosts -tema )
    fys_regnemaskiner   = ( elektron hope isak mephisto vakuum )
    fys_west            = ( hope fidibus wolfram mephisto indium tin isak silver )
    fys_east            = ( +fysikk-sun4-hosts -mephisto -silver -tema )

    HP_bin_clients      = ( wolfram indium tin isak )
    HP_bin_servers      = ( hope gran )

    fys_NIS_servers     = ( anyon fidibus gran )
    fys_PCNFSD_servers  = ( anyon fidibus gran mephisto )
    fys_timeclients     = ( arsen boson fermion gluino gluon gran hope indium
                            linn rogn selje semion sir tema tin vidje wolfram )
    fys_cern_hosts      = ( isak anyon )
    fys_ftp_hosts       = ( aurora )

#########################################################################################

import                                                  # inherit global set-up

   any +
        (
        <f global_classes
        )
```

```
   fys_fast =
      (
      cf.faststoff
      )

##############################################################################

broadcast:

  any = (
        zeroes
        )

##############################################################################

defaultroute:

   fys_alle =
      (
      fys-gw
      )

   tema =
      (
      inno-gw
      )

##############################################################################

resolve:

   fys_east =
      (
      129.240.22.222  # elektron
      129.240.22.230  # mephisto
      129.240.2.3     # nissen
      )

   fys_west =
      (
      129.240.22.201
      129.240.22.230
      129.240.22.222
      )
```

```
    fys_fast =
        (
        129.240.22.201
        129.240.22.230
        129.240.22.222
        )

    tema =
        (
        129.240.2.3
        129.240.64.2
        129.240.2.40
        )

##########################################################################################

homeservers:

  fys_teori   = ( anyon fidibus )
  fys_fast    = ( hassel linn fidibus )
  fys_felles  = ( fidibus hope anyon aurora tema linn hassel mephisto
                  isak wolfram )
  fys_strukt  = ( tema fidibus )
  fys_plasma  = ( aurora fidibus )
  fys_kjerne  = ( isak fidibus hope )
  fys_elg     = ( fidibus hope wolfram )

##########################################################################################

binservers:

    fys_teori.sun4   = ( gluino )
    fys_fast.hpux    = ( gran )
    fys_fast.aix     = ( alm )

  # fys_felles.sun4    = ( mephisto )       # mephisto mirrors anyon

    fys_supersparc = ( mephisto )
    any sun4       = ( anyon )
    any ultrix     = ( fidibus )
    any hpux       = ( hope )

##########################################################################################
```

```
mailserver:

    fys_alle    = ( fidibus:/usr/spool/mail )            # /usr/spool/mail
    fys_strukt  = ( ulrik:/usr/spool/mail )

################################################################################

mountables:

    (
    anyon:/mn/anyon/u1              # check these against the hosts
    anyon:/mn/anyon/u2              # groups listed above in binservers/
    anyon:/mn/anyon/u3              # homservers and match a hostname in
    anyon:/mn/anyon/u4
    anyon:/mn/anyon/local           # an arbitrary field to figure out which
    anyon:/mn/anyon/fys             # host it belongs to.
    anyon:/mn/anyon/particle
    anyon:/mn/anyon/epf-pp
    gluino:/mn/gluino/pc
    mephisto:/mn/mephisto/local
    mephisto:/mn/mephisto/fys
    fidibus:/mn/fidibus/u1
    fidibus:/mn/fidibus/u2
    fidibus:/mn/fidibus/local
    fidibus:/mn/fidibus/fys
    hope:/mn/hope/fys
    hope:/mn/hope/local
    hope:/mn/hope/u1
    wolfram:/mn/wolfram/u1
    tema:/rn/tema/u1
    aurora:/mn/aurora/u1
    aurora:/mn/aurora/u2
    aurora:/mn/aurora/u3
    gran:/mn/gran/fys
    gran:/mn/gran/local
    hassel:/mn/hassel/u1
    linn:/mn/linn/u1
    alm:/mn/alm/fys
    alm:/mn/alm/local
    isak:/mn/isak/u1
    isak:/mn/isak/cern
    )

################################################################################
```

```
misc_mounts:

    any =
        (
        fidibus:/mn/fidibus/fys       /mn/fidibus/fys         rw
        anyon:/mn/anyon/u2            /mn/anyon/u2            rw
        granitt:/mn/granitt/u1/mnfoto /mn/granitt/u1/mnfoto   rw
        )

    hpux =
        (
        priss:/use/priss/texfonts /use/priss/texfonts ro
        isak:/mn/isak/cern        /mn/isak/cern       ro
        )

    aix =
        (
        priss:/use/priss/texfonts /use/priss/texfonts ro
        )

    sun4 =
        (
        anyon:/mn/anyon/particle /mn/anyon/particle  rw
        )

    bjoern =
        (
        aurora:/mn/aurora/data   /mn/aurora/data     rw
        )

    hope =
        (
        anyon:/mn/anyon/epf-pp  /mn/anyon/epf-pp   rw
        wolfram:/mn/wolfram/fys/VIEWlogic /mn/wolfram/fys/VIEWlogic rw
        )

#########################################################################################

makepath:

    AllBinaryServers =
        (
        /mn/<host>/fys/public
        /local/lib/tex/fonts/pk
        )
```

```
   any =
      (
      /usr/spool/palantir
      )

   hpux.fys_regnemaskiner =
      (
      /mn/<host>/scratch
      )

################################################################################

links:

   any =
      (
      /fys               -> /<faculty>/<binserver>/fys
      /etc/moduser.site -> /fys/mutils/bin/fys-reg       # used by ureg
      )

   sun4 =
      (
      /usr/lib/X11    -> /fys/mutils/X11
      /local/bin/form -> /fys/lib/form/form
      /cern           -> /mn/anyon/particle/cern
      )

   AllBinaryServers =
      (
      /public                -> /fys/public
      /local/bin/X11         -> /local/X11R5/bin
      /local/lib/GS/data     -> /local/GS/lib/data/fonts
      /local/lib/GS/fonts    -> /local/GS/lib/data/fonts
      )

   aim =
      (
      /local/GS/lib/data/fonts -> /local/gnu/lib/ghostscript/fonts
      )

   anyon =
      (
      /etc/bootptab   -> /local/tftpboot/vxt/bootptab
      )
```

```
arsen =
   (
   /mn/arsen/fys      +> linkchildren
   )

aurora =
   (
   /mn/aurora/fys     +> linkchildren
   )

elektron =
   (
   /mn/elektron/local +> linkchildren
   )

fidibus =
   (
   /usr/mail  -> /usr/spool/mail
   /fys/info  -> /mn/fidibus/fys/lib/gopherd/fysinfo
   )

gluon =
   (
   /mn/gluon/local +> linkchildren
   /scratch        -> /mn/gluon/local/scratch
   /work           -> /mn/gluon/local/scratch
   )

hassel =
   (
   /scratch         -> /mn/hassel/u1/tmp
   /work            -> /mn/hassel/u1/tmp
   )

hope =
   (
   /fys/modsim      -> /fys/modsim1.8
   /cern            -> /mn/isak/cern
   /fys/cern        -> /mn/isak/cern
   )

HP_bin_servers =
   (
   /fys/common      -> /mn/fidibus/fys/common
```

```
   /fys/info       -> /mn/fidibus/fys/info
   /fys/doc        -> /mn/fidibus/fys/doc
   /fys/src        -> /mn/fidibus/fys/src
   /fys/include    -> /mn/fidibus/fys/include
   /pc             -> /mn/fidibus/fys/pc
   /fys/proj/qgsm  -> /mn/fidibus/fys/proj/qgsm
   )

hpux =
   (
   /usr/ucb/rdist  -> /fys/mutils/contrib/rdist/rdist
   /fys/man        -> /mn/fidibus/fys/man
   /usr/bin/elm    -> /local/bin/elm
   )

mephisto =
   (
   /local/TEX/lib -> /mn/anyon/local/TEX/lib
   /var/yp        -> /mn/mephisto/scratch/.nisfiles
   )

fys_regnemaskiner =
   (
   /scratch  ->  /mn/<host>/scratch
   /work     ->  /mn/<host>/scratch
   )

silver =
   (
   /mn/silver/fys  +> linkchildren
   /fys/cadence/4.2.1adoc -> /fys/misc/cadencedoc
   )


fys_teori =
   (
   /pc             -> /<faculty>/<binserver>/pc
   /public         -> /fys/public
   )

fys_cern_hosts =
   (
   /cern/pro -> /cern/93d    # This should point to the current version
   )
```

```
#
# Next line is to link in USIT's packages.
#

  packages =
           (
           /local/bin        +> /local/perl/bin
           /local/bin        +> /local/TEX/bin
           /local/bin        +> /local/MF/bin
           /local/bin        +> /local/elm/bin
           )

##############################################################################

tidy:

   AllHomeServers.exclude =
       (
       home                      core          R      >0
       home                      *%            R      >3
       home                      \#*           R      >1
       home                      *.dvi         R      >14
       home                      *~            R      >3
       home                      XtermLog.*    R      >3
       home                      *.nfs         R      >0
       home                      *.CKP         R      >1
       home                      *.BAK         R      >1
       home                      *.log         R      >14
       home                      *.aux         R      >3
       home                      .deleted      R      >2
       )

   AllBinaryServers =
      (
      /public                    *%            R      >1
      /public                    *~            R      >1
      /local/lib/tex/fonts/pk/   *             R      >14
      )

   fys_PCNFSD_servers =
      (
      /var/spool/pcnfs           *             R      >1
      )
```

```
any =
   (
   /tmp/                        *              R     >1
   /usr/tmp                     *              R     >1
   /etc                         printcap.old   N     >0
   /                            core           R=2   >0
   )

fys_regnemaskiner.exclude =
   (
   /scratch                     *              R     >2
   )

hope.exclude =
   (
   /mn/hope/u1/anneho           *.o            R     >7
   )
```

****

```
files

   AllBinaryServers.exclude =
      (
      /fys                      -0002 R [root] [0,1,3,mndrift,fysadmin]   fixall
      /local                    -0002 R [root] [0,1,2,3,4,5,6,tex,palsrc] fixall
      /local/lib/tex/fonts/pk   =0777 R [root] [0]                        fixall
      )

   any =
      (
      /etc/unit/palantir client =0644 N [root] [*]          touch
      /usr/spool/palantir       =0644 R [palantir] [palsrc] fixall
      )

   anyon =
      (
      /public                   =0666      R [*]             [fysteori]   fixall
      /mn/anyon/epf-pp          -2002=0660 R [root,=fys-epf] [fysepf]     fixall
      )

   fidibus exclude =
      (
```

```
    /fys/drift                  -0002+0770 R [*]     [mndrift]  fixall
    /fys/man                    =0664      R [root]  [mndrift]  fixall
    /fys/linux                  -0002      R [*]     [linux]    fixall
    /fys/lib/gopherd/fysinfo    =0664      R [*]     [mndrift]  fixall
    /fys/ftp/incoming           =0777      R [root]  [*]        fixall
    /var/spool/mail             =1777      N [root]  [*]        warnall
    /fys/bin/pubreg             +4022+0011 N [root]  [fysadmin] fixplain
    )

fys_felles =
    (
    /fys/proj                   -2002      R [*]     [*] fixall
    )

gran =
    (
    /local/lib/idl              -0002      R [root]  [*] fixall
    )

AllHomeServers.exclude =
    (
    home                        -2002      R [*]     [*] fixall
    )

fys_PCNFSD_servers =
   (
   /usr/spool/pcnfs/.           =0777      N [root]  [0] touch
   )

fys_ftp_hosts.sun4 =
    (
    /fys/ftp                    =0555      N [ftp]   [daemon] warndirs
    /fys/ftp/bin                =0555      N [root]  [daemon] warndirs
    /fys/ftp/etc                =0555      N [root]  [daemon] warndirs
    /fys/ftp/usr                =0555      N [root]  [daemon] warndirs
    /fys/ftp/dev                =0555      N [root]  [daemon] warndirs
    /fys/ftp/pub                =0555      N [ftp]   [daemon] warndirs
    /fys/ftp/etc/passwd         =0444      N [root]  [daemon] warnplain
    /fys/ftp/etc/group          =0444      N [root]  [daemon] warnplain
    /fys/ftp/usr/lib/libc.so.1.8  =0555 R [root]  [daemon] warnplain
    /fys/ftp/usr/lib/libdl.so.1.0 =0555 R [root]  [daemon] warnplain
    )

fys.regnemaskiner =
    (
```

```
    /mn/<host>/scratch   =0777 N [root] [*] fixdirs
    )

################################################################################

ignore:

  any =
    (
    !*
    /local/lib/gnu/emacs/lock/
    /local/tmp
    /fys/pc
    /fys/public
    /fys/info
    /fys/drift
    /fys/lib/gopherd
    /fys/ftp
    /fys/src
    /fys/linux
    /fys/ftp
    /local/bin/top
    /fys/man
    /local/lib/tex/fonts
    /local/palantir
    /local/etc
    /local/ftp
    /fys/proj
    )

  NP_bin_servers =
    (
    /fys/common
    /fys/doc
    /fys/info
    /fys/include
    /fys/pc
    /fys/src
    /fys/proj/qgam
    )

  fys_files =
    (
    /fys/proj
```

```
    )

  gran =
    (
    /local/lib/idl
    )

  wolfram =
    (
    /local/elm
    )

#########################################################################################

required:

  any =
    (
    /<faculty>/<binserver>/fys
    /<faculty>/<binserver>/local
    )

  hpux =
    (
    /local/elm
    )

  fys_teori =
    (
    /<faculty>/<binserver>/pc
    )


#########################################################################################

editfiles

  sun4 =

    ( /etc/netmasks

    AppendIfNoSuchLine "129 240          255 255 254 0"
    )
```

```
##############################################################################

shellcommands:

   anyon.Sunday.exclude =
      (
      "/fys/mutils/bin/noseyparker /mn/anyon/u1 nomail"
      "/fys/mutils/bin/noseyparker /mn/anyon/u2 nomail"
      "/fys/mutils/bin/noseyparker /mn/anyon/u3 nomail"
      "/fys/mutils/bin/noseyparker /mn/anyon/u4 nomail"
      )

   anyon.exclude =
      (
      "/fys/mutils/bin/set-date"
      "/usr/ucb/rdist -f /fys/mutils/cron/Mephisto.distfile > /dev/null"
      )


   fidibus =
      (
      "/fys/mutils/cron/fidibus.daily"
      )

   fys_NIS_servers =
      (
      "/fys/mutils/bin/getnismaps > /dev/null"
      )


   AllBinaryServers.sun4.exclude =
      (
      "/fys/lib/locate/updatedb"                    # GNU find  testing
      "/usr/etc/catman -w -M /local/man"
      "/usr/etc/catman -w -M /fys/man"
      "/usr/etc/catman -w -M /local/X11R5/man"
      "/usr/etc/catman -w -M /usr/man"
      "/usr/etc/catman -w -M /local/gnu/man"
      )


   AllBinaryServers ultrix exclude =
      (
      "/fys/lib/locate/updatedb"                    # GNU find  testing
      "/usr/etc/catman -w /local/man"
```

```
    "/usr/etc/catman -w /fys/man"
    "/usr/etc/catman -w /local/X11R5/man"
    )

fys_timeclients.sun4 =
    (
    "/usr/ucb/rdate anyon >/dev/null"
    )

aurora.Sunday.exclude =
    (
    "/fys/mutils/bin/noseyparker /mn/aurora/u1 nomail"
    "/fys/mutils/bin/noseyparker /mn/aurora/u2 nomail"
    "/fys/mutils/bin/noseyparker /mn/aurora/u3 nomail"
    )
```

B Import file

```
#######################################################################
#
# GLOBAL class definitions for cfengine
#
# These configurations are inherited by all machines
#
#
# Mark S. Jan. 1994
#
#######################################################################

links:

       any = (
             /local              -> /<faculty>/<binserver>/local
             /usr/local          -> /local
             )

      hpux = (
             /var/spool      -> /usr/spool
             /usr/spool/mail -> /usr/mail
             /etc/logingroup -> /etc/group
             /bin/wall       -> /etc/wall
             )

   solaris = (
             /usr/spool      -> /var/spool
             /var/spool/mail -> /var/spool
             )

#######################################################################

files

   any =
      (
      /etc/motd                    =0644 S [root] [=] touch
      / rhosts                     =0600 S [root] [=] touch
      / hushlogin                  =0644 S [root] [=] touch
      )

   hpux =
      (
```

```
      /etc/passwd                =0644 N [root] [sys] fixplain
      /etc/group                 =0644 N [root] [sys] fixplain
      /dev/kmem                  =0640 N [bin]  [sys] warnall
      )

   sun4 =
      (
      /etc/passwd                =0644 N [root] [staff] fixplain
      /etc/group                 =0644 N [root] [staff] fixplain
      /lib/libc.so.1 8.3         =0644 N [root] [bin] warnall
      /dev/kmem                  =0640 N [root] [kmem] warnall
      )

   ultrix =
      (
      /etc/passwd                =0644 N [root] [system] fixplain
      /etc/group                 =0644 N [root] [system] fixplain
      /dev/kmem                  =0640 N [root] [kmem] warnall
      )

# osf
# solaris

#########################################################################################

disable

   any =
      (
      /etc/hosts.equiv
      /etc/nologin
      )

#########################################################################################

editfiles

   any =
      ( / rhosts
      AppendIfNoSuchLine  "anyon"
      AppendIfNoSuchLine  "anyon uio no"
      )

#########################################################################################
```

```
shells:

   any =   (
           /bin/sh
           /bin/csh
           /bin/ksh
           /local/bin/tcsh
           /local/bin/bash
           /local/bin/gnu/bash
           /local/gnu/bin/bash
           )

#########################################################################

shellcommands:

   sun4 =
      (
      "/bin/sh -c '[ -x /lib/libc.so.1.8.3 ] && /bin/echo order hosts,bind,nis > /etc/ho
      )

   hpux =
      (
      "/bin/echo setenv TZ MET-1METDST > /etc/csh.login"
      )
```

C Wrapper

Here is a simple wrapper program such as one might use to mail the system administrator. In this case the mail address is "admin@mydomain".

```
#!/local/bin/perl
#########################################################################
#
# Script wrapper, mails output if there is any
#
#########################################################################

$mail = 0;

$sysadm = 'cfengine -a';

$comm = join(" ",@ARGV);
```

```
@path = split(m;/;,$comm);
$suff = $path[$#path];
$tmpfile = "/tmp/mwrap.$$";
$subjectfile = "/tmp/mwrapsub.$$";

open (H,"/bin/hostname |") || die "mwrap: Can't get hostname";

 $hostname = <H>;

close(H);

open (SH,"$comm 2>&1 | ") || die "mwrap: Can't start shell\n";
open (OUT,">/tmp/mwrap.$$") || die "mwrap: Can't open workfile\n";

while (<SH>)
   {
   if ( /\S/ )
      {
      print OUT "$_";
      $mail = 1;
      }
   }


if ($mail)
   {
   open (SUB,">$subjectfile") || die "Cannot open $subjectfile\n";
   print SUB "Subject: $comm\n\n";
   print SUB "This message originates from host $hostname\n";
   print SUB "The full command issued was: $comm.\n\n";
   close(SUB);
   }

close(OUT);
close(SH);

if ($mail)
   {
   system ("/bin/cat $subjectfile $tmpfile | /bin/mail $sysadm");
   }

unlink ("$subjectfile");
unlink ("$tmpfile");
```

FYSISK INSTITUTT FORSKNINGS-GRUPPER	DEPARTMENT OF PHYSICS RESEARCH SECTIONS
Biofysikk	*Biophysics*
Elektronikk	Electronics
Elementærpartikkelfysikk	Experimental Elementary Particle Physics
Faste stoffers fysikk	Condensed Matter Physics
Kjerne- og energifysikk	Nuclear and Energy Physics
Plasma- og romfysikk	Plasma and Space Physics
Strukturfysikk	Structural Physics
Teoretisk fysikk	Theoretical Physics