Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin

Dr. Magnus Harlander – GeNUA mbH, München, Germany

ABSTRACT

GeNUAdmin is an automatic system administration tool to control the configuration and administration of Unix workstations of different vendors from a central management host.

All relevant data about the system are kept in a central data repository. The configuration for each machine is extracted from these databases. Configuration files and installation methods are generated for many different architectures.

Before calculating and updating the configuration of each machine many consistency checks using pattern matching or inherent data correlations are performed. This assures data consistency as well as configuration consistency for a large number of machines and users.

The system is written in perl and creates Bourne shell scripts to accomplish its tasks. The prerequisites for using it are minimal.

GeNUAdmin is available without charge.

Motivation

Wouldn't it be nice to have an *OBSAT* – the ''One Button System Administration Tool''? Push the button and it does what you want. However, a cluster of Unix workstations is not a triviality, especially since nature has supplied us with a wide variety of operating systems, different command line options and file formats. So the real world does not allow us an *OBSAT*!

On the other hand, it is possible to represent all relevant data of a system configuration in an abstract notation. Using the knowledge about different file formats and installation methods it should be possible to deal with the maintenance of this abstract data repository and let the real work be done by an automatic system.

By modifying one database it should be possible to add a new user, printer, disk, or modify your existing configuration.

Introduction

GeNUAdmin's main goal is to automate as many system administration tasks as possible. It enables the system administrator to manage large numbers (hundreds) of machines and users (thousands) from a central server without the need to log in to any of these machines.

By automating many tasks we get the chance to perform consistency checking at many different levels of the configuration process. Using perl [1] as the programming language allows us to perform many powerful and efficient tests at different levels of the configuration process. Using perl's powerful pattern matching methods and associative arrays, we can check, e.g., for correct format of variable values or for the uniqueness of variables (e.g., for IP addresses or user names).

Consistency checks are performed

- when creating or modifying the databases,
- when parsing the databases,
- when creating the configuration and
- when installing the configuration on the target host.

GeNUAdmin is a tool for automating many tasks but it should not be used by inexperienced system administrators. It is essential to understand the structure and internals of a Unix cluster in order to avoid catastrophic errors.

GeNUAdmin is executed in several steps.

- Parsing the databases
- Creating a new configuration
- Installing the new configuration

Each run level can be executed by hand. So you can run GeNUAdmin without touching the existing configuration. The system administrator has the opportunity to check that the new configuration really looks as expected.

The installation process of a machine's configuration is split into so-called targets. A target might be, e.g., the configuration of `/etc/fstab` or the setup of a name server. It is possible to address specific targets on specific hosts for installation. This enables us to test the effect of a modification to the databases on a single machine before reconfiguring all machines at once.

The Databases

All information relevant to the configuration of GeNUAdmin targets is collected in a few ascii database files on the central server.

The Structure of the Databases

Some databases (e.g., the 1`netgroup.db` and `group.db` files) are laid out in a rather simple tabular format since the related information and information structure is more or less static. Figure 1 shows an example for `netgroup.db`.

```
####################################
# netgroup.db: defines all netgroups
####################################
host:all-hosts:sun4-hosts pmax-hosts:
host:sun4-hosts::
host:pmax-hosts::
user:all-users:cs-users ee-users:
user:cs-users::
user:ee-users::
```

Figure 1: A simple table to define all available netgroups. This table contains mostly redundant information. We use the entries mainly for consistency checks with respect to user and host netgroups from the other databases.

Entities like users, hosts, printers or filesystems however are treated as complex objects with many properties. The objects are stored in a stanza format, which avoids a static information structure and allows us to add properties on demand. All stanza keys however are cross checked with a static list of known keys to avoid typos and inconsistencies in the database itself. Figure 2 shows the format of a stanza entry for the `hosts.db` database.

```
BEGIN_HOST
name      = EntryName
key       = Value
key       = Value
....
default   = DefaultName
END_HOST
```

Figure 2: Complex data structures are stored in stanza format. We use it to describe hosts, users, printers and filesystems.

For all objects, it is possible to inherit properties from default entries which can again inherit information from other default entries. The levels of inheritance are unlimited. Figure 3 illustrates a useful constellation of default entries.

It is clear that with this method managing many similar hosts becomes very easy. The entry for a single host becomes very short containing only key/value pairs for the host name, IP address and ethernet address. Figure 4 shows an example for a server entry and two cloned clients.

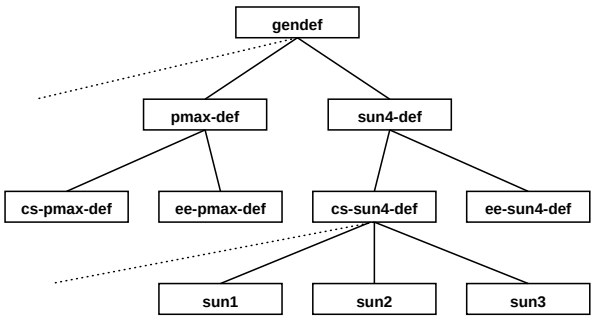

Figure 3: Example for the hierarchical structure of the object oriented databases. *sun4* and *pmax* defaults inherit information from the most generic default *gendef*. Department defaults like *cs-sun4-def* extend this information which finally is used by the hosts *sun1*, *sun2*, ..., etc.

```
BEGIN_HOST
name        =  server1
address     =  135.33.44.1 135.33.45.1
cname       =  loghost mailhost
ypmaster    =  YoUR.NiS.dOMain
....        =  .....
default     =  some-default
END_HOST

BEGIN_HOST
name        =  client1
address     =  135.33.44.2
default     =  some-default
END_HOST

BEGIN_HOST
name        =  client2
address     =  135.33.44.3
default     =  some-default
END_HOST

BEGIN_DEFAULT
name        =  some-default
ostype      =  bsdi
....        =  .....
default     =  basic-default
END_DEFAULT

BEGIN_DEFAULT
name        =  basic-default
domain      =  your.ns.domain.com
....        =   .....
END_DEFAULT
```

Figure 4: Example for entries in `hosts.db`. Server entries may be longish while client entries tend to be minimal.

List of Databases

The information about the different types of objects is split into several object specific files. The following listing gives an overview over the most important databases and related variables. The **defines.db** file contains a list of variables for

controlling the general behavior of GeNUAdmin. Next to things like path names and mail addresses this file contains also switches for enabling or disabling the execution of certain targets.

The **hosts.db** file contains all data relevant to a host. These are:

- host name, domain name and a list of additional domain names,
- IP and ethernet addresses, subnet masks,
- name server information like CNAMES and MX records,
- architecture, operating system type and version,
- a list of local filesystems,
- amd related variables,
- a list of hosts for root's `.rhosts` -file,
- a list of host netgroups the host should belong to,
- lists of domains for configuring a primary or secondary name server,
- flags for configuring master or slave ypservers,
- variables for a set of files to keep up-to-date:
  - crontab
  - services
  - inetd.conf
  - syslog.conf
  - resolv.conf
- a flag whether this host is to be managed by GeNUAdmin.

The **disks.db** file contains all relevant data for network filesystems. The data are used for exporting and mounting filesystems via `fstab` or `amd`. The most important filesystem variables are:

- the location of the disk. For replicated filesytems this might be a list of file servers. GeNUAdmin will determine the closest – in terms of network metric – for each host,
- the mount point,
- the original directory name on the file server,
- the mount options,
- an access list based on hosts and netgroups,
- architecture and os-type dependencies, since binaries for a SUN should not be mounted on a HP,
- the preferred mount method: amd or fstab.

The **user.db** file contains all users for the cluster with the following user properties:

- first name,
- last name,
- user name,
- user id,
- a list of group ids (the first will be the primary group),
- a list of netgroups the user should belong to,
- home directory,
- login shell,
- gecos field data,
- quotas for a list of filesystems,
- expire date for the account. GeNUAdmin will disable the users login shell after this date but the user will get several warnings before getting disabled.

In the next section GeNUAdmin's concept of user management will be explained in detail.

The **printer.db** file contains all available printers.

The **links.db** file is for maintaining a list of soft links on all systems. The entries in this database have tags that may be used to decide which links should be made on which hosts. The decision may depend on the architecture, operating system or a match against a list of host names.

The **net.db** file contains a description of the net topology. With this information about nets, subnets and gateways it is possible to create static routing tables and determine the distance in terms of gateway hops between two hosts. When creating `fstab` entries for multi homed hosts or replicated filesystems, this information is used in order to find the closest file server.

User Management

All databases are modified using a regular editor, however, a tk/tcl [2] based graphical interface is under development.

The user database `user.db`, however, can be accessed and modified by an home brew RPC server (*usermgrd*). For fast access the data used by *usermgrd* are stored in a dbm file. Therefore the data from the dbm file must be converted to an ascii file in stanza format before using it with GeNUAdmin. Actually GeNUAdmin does this for you.

User management – adding, modifying and deleting users – can all be done by simply modifying the `user.db` database. The modification is performed by communicating with the *usermgrd* through one of two RPC based client programs: *usermgr* and *rpasswd.*

usermgr

Usermgr is intended for user management by junior (and senior) system administrators.

With optionally restricted access to a subset of users they have the possibility to

- add,
- delete,
- enable or disable,
- modify, and
- set the password

of their users.

The access rights are based on

- the location of the home directory,
- group and netgroup membership,
- the maximum quota value,
- the maximum lifetime of the account, and
- a range of user ids.

Usermgrd reads the file `auth.db` to learn about the access rights.

To grant free access for senior system administrators wild cards (''*'') can be used in some or all of the fields. Figure 5 shows an example of a small authentication database.

```
#####################################
# auth.db: granting access to user.db
#####################################
# user:dir:groups:ngs:quotas:exp:uids
#####################################
bigmac:/:*:*:0:0:0-32000
ee:/home:ee:*:50000:30000:0-10,50-199
cs:/home/cs:cs:ng:20000:30000:200-299
```

Figure 5: An example for the authentication database `auth.db`. Three users allowed to modify the database. `Bigmac` has unrestricted access while access for the users `ee` and `cs` is restricted to users of their department only.

rpasswd

The second program, *rpasswd*, is intended for use by all users and allows them to change their own password or login shell by modifying the central password file.

The server – *usermgrd* – can be told which password file to change and which command to execute after the update of the password file. Therefore one can use NIS master files, local password files and even shadow password files.

If you don't like NIS, this actually might be the method to maintain a central password file. You could rdist this file once in a while to all your machines or use GeNUAdmin to update the local password files (See the section ''Local password files'').

Consistency

The client asks the operator for a couple of user attributes and checks with the server whether the attributes are acceptable. It asks the server also for an unique user id. The server checks user name, user id, first and last name for uniqueness. It also checks the format of all known keys (e.g., uid must be numeric). If first and last name already exist, you might add initials of additional names with a period (e.g., John Kennedy -> John.F Kennedy).

After completing the form, a final check for consistency is performed by the server before updating the database.

Access Control and Security

The user is asked for her password before any data are accepted or delivered by the server. The authentication is valid through the entire TCP session.

If there are repeated attempts to get authorization with wrong passwords, *usermgrd* closes the connection after three of these attempts. If there are more than 20 attacks within 10 minutes the server quits operation and sends mail with notice about its termination to root.

Passwords are never sent clear text over the net. The passwords are encrypted using a des binary (based on *libdes* [3]). If the des program could not be found, the client uses a very simple XOR modification to make it at least unreadable. We provide compiled binaries of the des program for:

- BSD/386
- SPARC
- Pmax
- HP/PA
- RS6000
- SGI

Whenever a new user is added the server will ask for a password for this user and do a lot of security checks on this password. It can even use programs like `checkpasswd` [4] to cross check the password against dictionaries. Only if it is a good password it will be accepted. The same procedure is used for changing passwords with *usermgr* or *rpasswd*. If you have good dictionaries, you might be well prepared for crack [5] attacks!

In case the perl library files `syslog.pl` and `syslog.ph` are available, *usermgrd* will log any interesting event with the syslog facility. In addition it keeps a log of events in a private log file.

Batch Processing

Many sites have the need to add many users in a bulk process to their user name space. Instead of entering hundreds of users by hand, you can process the content of a file in batch mode using *usermgr*.

The entries for such a file can be generated by the perl script `account` which can be used as a login shell for a login granting account.

Customizing

It is possible to extend the list of properties for a user. The server *usermgrd* uses a separate configuration file to learn about additional site specific user tags and their descriptions. This enables you to add as many features to a user as you like to.

The Targets

The configuration process as performed by GeNUAdmin is divided into different targets. If a target should be processed, i.e., create configuration files and installation methods, it must be activated by setting a variable.

Here is a list of all currently available targets:

- Netgroup
- Passwd
- Ethers
- Ypservers
- Group
- Etchosts
- Dotrhosts
- Krb_Hosts
- Krb_Users
- Exports
- Fstabs
- Amd
- Links
- Pseudonyms

- Xaliases
- Bootptab
- Nameserver
- Crontabs
- Distfile
- Resolv_conf
- Inetd_conf
- Syslog_conf
- Services
- Dottwmrc
- Lpdhosts
- Printcaps
- Lhosts_db
- Quota_check
- Routing
- Ifconfig

Some of the targets are already described in the chapter about the databases (''The Databases''). Where it is useful an attempt will be made to group the descriptions for logically related targets.

NIS

We can generate NIS maps [6] for the following files:

- passwd,
- hosts,
- group,
- netgroup,
- ethers,
- services, and
- ypservers.

As an alternative, local versions of these files can be updated if NIS is not used on a host.

Local password files

There are two cases were it might be interesting to create local password files instead of using NIS.

1. You have machines that are not able to talk NIS.
2. You don't want NIS because of its security or reliability problems.

Each host can select – based on netgroups – which users are to be included in the local password file. There is the possibility to specify a minimal and maximal user id. Only users within that range will be taken from the central password pool. That is why local users like root, uucp, ... will not be touched.

Name server

If selected, primary and secondary name server zone files and boot files are created including address, cname and mx records. Configurations for forward and reverse mappings can be generated. At some sites primary and secondary name servers with more than 20 forward and reverse maps are administered by GeNUAdmin.

Sendmail

GeNUAdmin enables keeping up-to-date both IDA and V8 sendmail [7] configurations for a central mail server. The targets related to the mail system are:

- pseudonyms
- xaliases

The pseudonyms file is used by the central mail server to identify hosts for which it should accept mail.

The xaliases target installs an xaliases database for an IDA sendmail with incoming and outgoing aliases for all users. The aliases file contains user names and the users' full names.

Printing

The contents of **printer.db** are evaluated to configure the printing system via the targets **printcaps** and **lpdhosts**. At the current point printer configuration is well supported for BSD-Systems. GeNUAdmin creates printcaps for servers and clients, creates spool directories, accounting files and logfiles.

The `/etc/hosts.lpd` files are created for all printer servers.

For some SYSV systems (hpux, riscos) setting up printing is also implemented. It might be easy to copy these methods to other SYSV systems.

Filesystems

There are five targets concerning filesystem administration:

fstab and exports

With the targets fstab and exports both files are created and installed and the current export and mount configuration is updated.

When creating the fstab, every effort is made to find the closest server – in terms of the network metric – of a set of replicated filesystems (e.g., /usr/local) or to find the closest interface name of multi homed servers for the simple reason that NFS and NIS always use the first address they get from the name services (NIS or BIND). This would cause all your NFS clients to use the same interface unless you explicitly tell them to use another interface.

amd

It is possible to create amd maps for the Berkeley auto mounter `amd` [8]. The maps will contain filesystems of type nfs, link and auto. Replicated filesystems and multi homed hosts are also taken into account when creating nfs entries.

lngroups

This is not really a target but included in the amd and fstab targets. Its purpose is to hide the real location of a users home directory from the users view and use something like `/homes/username` in the password file. The entry in `/homes` will be a soft link to the real location. This target maintains these soft links for all users on all machines.

It can be achieved by either using amd's link filesystem in an amd map or by creating a script that makes the soft links for all users.

links

This target evaluates the contents of the `links.db` database and maintains a set of soft links on all machines. The selection of soft links can be controlled using architecture, operating system type or host specifications.

quota_check

This creates quota lists for all local filesystems that use quotas. The lists are scanned by a utility perl script that will adjust the quotas of all users.

Installing Users

If a user appears to be new in the user database, a script for installing a new user will be called. It is also possible to move the home directory of existing users from one disk to another by simply changing the home directory in the user database using *usermgr*. GeNUAdmin will recognize the new location and run the copy process.

Kerberos

In the current version of GeNUAdmin only the administration of the host principals in the kerberos master database [9] is implemented.

For all hosts, the realm file called `/etc/krb.realms` is generated and populated with all domains of the generic hosts. US export laws restrict us to Kerberos Version 4.

Network Configuration

It is possible to generate a file that contains commands for creating static routes. This file can be executed at boot time if `routed` is not used to get the proper routing information.

It is also possible to create a file that contains commands to configure all available network interfaces. This, too, can be executed at boot time to configure all network interfaces.

Daemon Configuration Files

The targets

- Resolv
- Syslog_conf
- Inetd_conf
- Services
- Crontabs

maintain consistent versions of a set of configuration files for system and network services. After installing updated versions, the corresponding daemons are notified about the new version.

With these targets a consistent central repository for the essential configuration files can be easily maintained.

Miscellaneous

The following targets are intended to make your life as system administrator easier. Their main intent is to keep lists of hosts up-to-date with the current stock of machines.

lhosts_db: This is a file which contains a list of all generic hosts with attributes like domain name, os-type and architecture. Using a little awk script, host name lists matching special properties can be retrieved from this database.

distfile: This creates a distfile for use with rdist. Host lists are provided for all hosts, all architectures, os-types and netgroups.

dottwmrc: This enables you to create *twmrc*, *fvwmrc* and *mwmrc* files which contain menus with commands to open a window on a host. The menus can be grouped based on host netgroups.

rhosts: Entries for root's `.rhosts` file are created. For each host that should be included in the `.rhosts` file all interface names and all IP addresses are added since some systems – especially if they use NIS for host name lookups – will not recognize hosts if the connection does not come from the primary network interface.

bootptab: Creates bootptab files with entries for X terminals or PC's which use the bootp protocol for network booting.

Technical Aspects

Running GeNUAdmin

There are several well separated steps in running a configuration process by GeNUAdmin.

- Dumping system data (password file contents, kerberos database, ...) into the GeNUAdmin workspace.
- Reading and consistency checking of the databases.
- Creating the configuration into a separate directory on the local disk without modifying the existing configuration.
- Installing selected targets on selected hosts or
- Installing the complete new configuration on all hosts.

It is possible to execute these different levels manually step by step and even redo them several times if needed. This enables checking of the intermediate results (e.g., the format and content of the new configuration files before installing them). There is also a command for saving an old configuration on the local disk and comparing a complete new configuration against a complete old configuration. This gives you an overview of the changes in all files of the configuration.

Usually, however, all steps will be executed with one command. This is normally done in regular intervals using the cron facility.

Implementation

GeNUAdmin's core part is written in `perl` and the utility programs are simple shell scripts. Hence porting to any flavor of a Unix operating system should be no issue as long as perl is available.

Operating system independent targets are included in the main driver script but all os-dependent parts are kept in separate files for each operating system type and target. The files are loaded at run time if they are available. A default method is provided for each target which can be used by operating system dependent routines. They might however implement a different method if the

default method is not applicable. If a method for an operating system does not exist, the target for machines of this type will be just ignored.

There is no problem in including non supported machine types into GeNUAdmin's framework. The os-type independent targets will be processed immediately and support for the machine dependent targets for the new machine may be added later.

Figure 6 shows an example of an operating system module for managing `inetd.conf` on an aix system.

```
#
# $Log: inetd_conf_aix.pl,v $
#
# $Revision: 1.2 $
# at $Date: 1994/05/02 $
#

sub inetd_conf_aix {
    local($h) = @_;
    &inetd_conf_default($h,
        "/etc/inetd.conf","inetimp");
}
1;
```

Figure 6: An example module file for an operating system type (aix) depended method (inetd_conf_aix) which calls a default method (inetd_conf_default) with os-specific arguments. The specialty in this case is the third argument – the command `inetimp` – which exists and must be executed only on aix systems. Other systems will call `inetd_conf_default` with only two arguments and no command will be executed.

Prerequisites

The prerequisites for using GeNUAdmin are minimal. You need two programs on each client for identifying the architecture (`arch`) and the operating system type (`ostype`) of the machine. The GeNUAdmin server needs `perl` (Version 4.036). On the client systems, perl is useful but not necessary. Some targets use lower quality shell scripts as replacement if perl is not available. Using perl, however, makes life much easier on the clients.

On each client which should have access to *usermgrd* using *usermgr* or *rpasswd*, perl is also needed unless the perl builtin dump function is used to create executable client programs.

GeNUAdmin uses rsh for the installation of the configuration and gets the data and programs from a read-only mounted NFS partition. An RPC based client-server model for exchanging the data and methods is under development.

Related Work

GeNUAdmin borrows many ideas from systems like Moira from MIT [10]. By now, however, it has grown a lot by adding many targets and tricks that we found necessary or useful at the sites, where we (GeNUA) are responsible for administration.

It still has some common functionality with systems like Moira from MIT or Tivoli's Management Environment [11]. There is however a major difference in the software and hardware requirements and the implications for your system when using one of the tools.

- No modification of the running system is needed. GeNUAdmin exclusively uses existing operating system tools and programs.
- No information about the internals of the running system or kernel is needed or used.
- You don't need any additional programs, servers, daemons, etc. on the clients save optionally perl.
- It's written in a script language (perl) and can easily be modified, extended and ported to new unix platforms.

Future Plans

- The distribution and installation of the configuration should be done using a RPC based client/server method instead of rsh/NFS. All data transmission will then be done via a TCP socket. The need for transmission could be determined using checksums to minimize network traffic.
- Support of AFS and DCE environments will come, too.
- An interface to high performance databases for the storage of large amounts of data instead of using flat ASCII files should be implemented. We plan to support first the freely available postgres database [12].

Availability

GeNUAdmin is available without charges. Contact the author for further information.

Most of the targets from section ''The Targets'' are implemented on the following operating systems:

- SunOS,
- Ultrix,
- HP-UX,
- IRIX4/5,
- AIX-3.2.x,
- DEC-OSF1,
- BSD/386,
- Solaris, and
- RiscOS.

Author Information

Magnus Harlander received his diploma and doctoral degree in theoretical physics from the Technische Universität München (TUM), where he had been working in theoretical nuclear physics and in the field of neural networks. He has been

responsible for system administration at the TUM and at Lawrence Berkeley Lab for several years until he became a co-founder of GeNUA (Gesellschaft für Netzwerk- und Unix-Administration) in 1992. Within his company he is responsible for software development. Most of these projects are targeted towards GeNUA's main field of activity: Unix System Administration. He can be reached via e-mail at harlan@genua.de .

References

[1] Larry Wall, Randal L. Schwartz, *Programming perl*, O'Reilly and Associates, Sebastopol, CA, 1991.

[2] John K. Ousterhout, Tcl and the Tk Toolkit, Addison Wesley Publishing Company, Inc, ISBN 0-201-63337-X), 1994.

[3] Eric Young, libdes Version 3.00, available via anonymous ftp from e.g., from ftp.psy.uq.oz.au, 1993.

[4] Clyde Hoover, npasswd and checkpasswd from the University of Texas, available via anonymous ftp from ftp.cc.utexas.edu, 1990.

[5] Alec D. Muffet, Crack the Sensible Unix Password Cracker, available via anonymous ftp, e.g., from ftp.informatik.tu-muenchen.de.

[6] Sun Microsystems Inc. *Network Information Services* SunOS Reference Manual, Volume 1 Section 8: ypbind(8), ypserv(8).

[7] B. Costales, *sendmail*, O'Reilly and Associates, Sebastopol, CA, 1993.

[8] Jan-Simon Pendry and Nick Williams, The 4.4 BSD Automounter, Documentation to amd version 5.3, available via anonymous ftp from usc.edu. in the directory `/pub/amd`, 1991.

[9] J.G. Steiner, C. Neuman, J.I. Schiller, *Kerberos: An Authentication Service for Open System Networks*, Usenix Conference Proceedings, Winter 1988, pp 191-202.

[10] Mark A. Rosenstein, Daniel E. Geer, Peter J. Levine, The Athena Service Management System, Winter Usenix 1988.

[11] Tivoli Systems, *Tivoli Management Environment*, 1992.

[12] M. Stonebraker, *Documentation to Postgres 4.2*, available via anonymous ftp from postgres.cs.berkeley.edu.