OMNICONF – Making OS Upgrades and Disk Crash Recovery Easier

Imazu Hideyo – Matsushita Electric

ABSTRACT

OS upgrades are a headache because after installing a new OS, many files and directories need to be modified or created by hand, to restore the host's previous (pre-upgrade) configuration.

On the other hand, saving entire / and /usr file systems for crash recovery is redundant because most files are unchanged, and copies exist on distribution media. In addition, restoring from backups after a disk crash is not as easy as an OS installation from distribution media because OS installation software does not necessarily include utilities to aid in doing so.

Difficulties in performing OS upgrades and disk crash recoveries are dramatically reduced if a complete set of ''changes'' (a set of changes is called a ''configuration'' in this paper) which have occurred throughout / and /usr can be observed and saved. ''Change'' means: 1) addition and deletion of files and directories; 2) modification of the content and status of files and directories. Dealing with changes is non-trivial because conventional commands such as tar, cpio, and dump cannot handle deletion and cannot alter the permissions of a file without restoring its contents.

If configurations can be stored under a single directory, OS upgrades become easier because the configuration can be restored by a simple operation after the upgrade. Instead of saving all files in / and /usr, one only needs to save changes to those file systems. One can easily perceive what the entire configuration is and modify merely a part of it.

In this paper, the author introduces a tool called ''OMNICONF'', which stores and restores ''configurations'' to and from a specified directory. OMNICONF is implemented in 2400 lines of Perl[1] code, under the concept shown above.

Motivation

Computers must be configured to be used practically. With UNIX, configuration means (for the most part) modifying or creating files and directories in the /, /usr, and /var file systems.

OS upgrades can be a nightmare if the administrator has changed and created many files in /etc, /usr/lib, /usr/etc, /usr/sbin, etc. In addition, he may have made directories for mount points, device files for additional devices, and symbolic links in the course of system configuration. He might change the mode, owner, or group of files and directories. A typical UNIX system has between 20 and 100 modified or created files and directories in the /, /usr, and /var file systems. After an OS upgrade, these operation need to be done again, usually by hand. In most cases, there is no complete list of such changes. Ordinary administrators, therefore, must rely solely on remembering these past operations and must repeat them precisely after an OS upgrade.

As well as OS upgrades, recovering from a disk crash can also be difficult. Information on how to reinstall the system from backup may be hard to find or such an operation may be difficult to accomplish correctly. In the past, there was no such thing as an automated OS installation program, thus administrators did installations manually: set disk partitions, make file systems with the ''newfs'' or ''mkfs'' command, read OS tapes with the ''tar'' or ''restore'' command, and so on. Today, convenient installation programs do this work for the administrator. Therefore, information and tools for manual installation become hard to find today and are sometimes buggy, creating a complicated situation. For example, can you correctly install a boot block by hand when BSD/386 is in the second FDISK partition of an PC-AT compatible machine? Since recovery from backup is similar to performing a manual installation, it is harder to do correctly today.

Concept

To deal with problems described above, the author has developed the OMNICONF system. This section describes the design concept of OMNICONF.

Assume that the changes to files and directories since the original OS installation has been stored. The author defines the set of changes (or ''configuration'' hereafter) as the following items:

1. The contents of modified and created files
2. A list of removed files and directories

If a configuration can be stored and restored, an OS upgrade is much easier: install the new version of the OS, then apply the configuration. In reality, the administrator may also need to modify the configuration if he applies it to a new version of the OS, because the format of certain file may have changed, or some files made redundant. However, the stored configuration aids greatly in the upgrade task because it is a complete list of changes. There is no longer a need to remember changes and implement them manually.

The benefit of the stored configuration is more dramatic with disk crash recovery. All the administrator needs to do is to reinstall the OS normally and then apply the configuration. There is no hassle for OS installation from backup media, and the time needed for recovery is reduced. (because in many cases, OS software is distributed with CD-ROM and installation from CD-ROM is faster than from tape, which is a typical backup medium). Requirements

Under this concept, what is required for OMNICONF to be a reasonable aid for system administration?

First, the system should handle any type of file: ordinary files, directories, symbolic links, and device files.

Second, modified files must be stored hierarchically. Assuming modified files are stored under the /config directory, a modified /etc/sendmail.cf would be stored as /config/etc/sendmail.cf and a modified /usr/lib/sendmail would be stored as /config/usr/lib/sendmail. If a configuration is stored as monolithic data, it becomes much more difficult to manipulate only part of a configuration before applying it to an upgraded OS.

OMNICONF can deal with changes in file type. For example, /tmp is originally a directory, but it may be changed by the administrator to become a symbolic link pointing to a new location, e.g., /var/root.tmp‖.

Administrators may change the mode, ownership, or group (the combination of this information will be hereafter referred to as an ''attribute'') of files and directories while keeping their contents intact. Attribute changes should be observed and saved without storing the content of a file. If the contents of a file is stored in such a case, the contents may be needlessly or harmfully applied during an OS upgrade.

System files may be removed for administration purpose. For example, to disable the routed daemon without modifying /etc/rc.local, one may instead remove /usr/etc/in.routed. The removal of files and directories must be observed.

When restoring the configuration, original files should not be overwritten, but should be stored elsewhere, since one may need to refer to the original versions of files such as sendmail.cf, inetd.conf, syslog.conf, etc. at a later date.

For some files, a certain command needs to be invoked after the file is modified. For example, the newaliases command should be invoked after the /etc/aliases file is modified. This sort of binding should be handled.

Even if a program that performs the above tasks works correctly, no one will want to use it if they need to maintain the complete list of changed files and directories by hand. Such a list should be generated automatically and maintained dynamically.

Conventional Commands

Commands such as dump, tar, and cpio can store only files newer than a certain date and time, and these conventional commands lack some essential functionality. For example, they overwrite files and do not save original files, they fail to create a symbolic link if a directory with the same path name already exists, they cannot restore attributes without restoring content, and they do not correctly handle file deletion.

Features

Now that the requirements have been stated, the author will describe the features of OMNICONF system.

List Changes

The key issue in storing a configuration is to list all modified/created/removed files and directories automatically. For this purpose, OMNICONF uses a list of all files and directories, which is created by a special command when the OS is installed or a machine is unpacked. This list is called the original ''profile'' of the OS. Here is an example of a portion of a profile file:

```
/:40755:0:0:767587342
/.cshrc:100644:0:10:711924842
/.login:100644:0:10:711924842
/.profile:100644:0:10:711924842
/.rhosts:100644:0:10:711924842
/bin:120777:0:0:767585902:usr/bin
/boot:100444:0:3:767586993
```

Each line denotes a file or directory and consists of colon-separated fields. The first field is the path name of the file. The second through fifth fields are pieces of data that are returned by the stat system call. The second field is the mode of the file (in octal), the third field is the UID of the file's owner, the fourth field is the GID, and the fifth field is mtime (when the file was last modified). Symbolic links have a sixth field, which is the contents of the link. The profile file of a SunOS 4.1.3 installation without any configuration contains 8,871 lines and is 440,867 bytes in size. The size varies, depending on which software sets are installed.

The original profile of an OS is stored in /etc/omniconf/profile. When a configuration is stored, the current profile and the original profile are compared. OMNICONF determines whether the contents of a file has been changed by comparing mtime's. Change of attribute is determined in a similar fashion. A file that is not listed in the original profile but is listed in the current one is determined to have been created. Deleted files can be determined similarly.

What portion of the entire file space should be handled? The administrator should specify this information in order for OMNICONF to work correctly. First, the file systems to be managed should be specified. Second, files and directories to be excluded should also be specified because saving certain files can be redundant or even harmful. For example, files that store system status, such as /etc/utmp, /etc/mtab, etc., should not be manipulated by OMNICONF. Files generated from other files such as /etc/aliases.dir, also should not be saved.

OMNICONF refers to /etc/omniconf/area to determine what portion of an entire file space is handled. Here is an example of area file for SunOS 4.1.3.

```
filesystem:
      /
      /usr
      /var
excluded:
      .pid$
      .lock$
      /etc/aliases.dir
      /etc/aliases.pag
      /etc/dumpdates
      /etc/ld.so.cache
      /etc/mtab
      /etc/psdatabase
      /etc/state
      /etc/ttys
      /etc/utmp
      /dev/console
      /dev/null
      ^/dev/tty
      ^/dev/pty
      /var/adm
      /var/log
      /var/tmp
```

The file consists of two portions: the file system portion and excluded portion. The file system portion is straight forward. The excluded portion contains path names and regular expressions. Entries that begin with a slash are taken as path names and otherwise are regular expressions. Files and directories that match excluded entries are ignored. A path name entry in the excluded portion matches the path itself and any descendant files and directories (children).

Preserve Originals

OMNICONF assumes that the original version of a file whose content may be modified is saved as *.orig. For example, the original /etc/sendmail.cf can be saved as /etc/sendmail.cf.orig. But one doesn't have to preserve original files if they are not needed.

When the administrator want to disable the routed daemon by deleting /usr/lib/in.routed, he has two alternatives: rename the file or unlink the file. OMNICONF recognizes deleted files and directories by renaming them *.orig. For example, an administrator may rename /usr/etc/in.routed /usr/etc/in.routed.orig. In this case, OMNICONF considers /usr/etc/in.routed as having been removed. OMNICONF also handles unlinked files and directories.

Modified or removed files and directories whose originals are preserved can be reverted to their original versions by a certain OMNICONF operation.

Store Configuration

A configuration is stored under the directory (referred to hereafter as the ''repository'') specified by /etc/omniconf/reposit. Assume that the repository is /config for example. The repository contains one directory named 'cont' (in this case /config/cont), and two files named 'remove' and 'chstat.'

Files and directories that are modified or created are stored hierarchically under the cont directory preserving mode, owner, group, and mtime. For instance, /etc/sendmail.cf is stored as /config/cont/etc/sendmail.cf.

In the /config/remove file, removed files and directories are listed line by line. An example of remove file is as follows:

```
!/etc/hosts.equiv
/usr/etc/in.routed
```

Unlinked files and directories are prepended with a '!.' Files and directories renamed *.orig and not unlinked are listed as just their path names.

/etc/omniconf/chstat contains attributes of files and directories whose attributes have changed but whose contents have not. Here is an example of chstat file:

```
/var/spool/uucppublic:644:4:8
/var/tmp:1777:3:10
```

Each line consists of a path name, permissions, UID of owner, and GID of group. Although all mode bits are stored in profile files, only permission bits, which are mode bits masked by 07777 (octal), are stored in the chstat file.

Command Binding

The binding of a file to a command is specified by /etc/omniconf/exec. Here is an example:

```
/etc/aliases
   newaliases
/etc/named.boot
   if [ -f /etc/named.boot ]; then \
      named.restart \
   else \
      kill `cat /var/run/named.pid` \
   fi
/vmunix
   fastboot
```

File names begin in column one, and bound commands have spaces or tabs at the beginning of lines. Much like the ''make'' command, command bindings are evaluated by /bin/sh. Order in the exec file is significant: entries in the file are examined in the order in which they appear.

Restore Configuration

The process of restoring a saved configuration consists of following steps.

1. Remove files and directories according to the remove file.
2. Change the attribute of files and directories listed in the chstat file.
3. Rename files whose new version exists in the repository. The original files are renamed as *.orig.
4. Copy files and directories with the cpio command.
5. Examine the exec file and invoke commands as specified.

Elements of OMNICONF

The current OMNICONF system is written in about 2400 lines of Perl code. It consists of following commands:

- mkoprof (Make OMNICONF profile), which is used to create an original profile in /etc/omniconf/profile when an OS is installed.
- putconf, which calculates difference between the current profile and the original profile, then stores the configuration under the repository directory.
- getconf, which reads the repository and restores the saved configuration.

These commands should run without a Perl interpreter since there may be no Perl interpreter available when they are invoked. Using the undump feature of Perl, the commands are made into pure executables.

Putconf and getconf use GNU cpio to write and read the repository. Standard cpio is inadequate since it cannot handle the Berkeley Fast File System properly.

Real Operations

After one has installed an OS or unpacked a UNIX machine, copy the mkoprof command to local disk (such as to /tmp) and invoke it to create /etc/omniconf/profile. On a SPARCstation 2 with SunOS 4.1.3, mkoprof takes about 2 minutes to complete, and the resulting file consists of 8,871 lines (440,867 bytes).

After the machine has been configured, install putconf, then prepare a repository directory, and create /etc/omniconf/area and /etc/omniconf/reposit.

When the administrator wants to store a configuration, he invokes putconf. Putconf take about 2 minutes to calculate the configuration in the example above, then putconf takes up to an additional 30 seconds to store data. A configuration typically amounts to between one and several megabytes. The administrator may want to save the file hierarchy under the reposit directory onto backup media. Compared to an entire OS area of between 50 and 150 megabytes in size, saving only a configuration consumes much less storage (and thus backup time).

When the system disk of a machine crashes and its OS is reinstalled, the administrator must restore the configuration previously saved onto backup media. In some cases, the repository may not have been affected by the failure. In that case, simply copy getconf to local disk and invoke getconf, specifying the repository directory as an argument.

An upgrade operation is not as simple as disk crash recovery. The administrator has to examine files and directories stored as the configuration of the previous version, and make some changes. For example, a kernel image (/vmunix, /unix, /bsd) should be removed. Then he can install the configuration by invoking getconf. OMNICONF helps OS upgrade procedures mainly for minor OS upgrades such as from SunOS4.1 to SunOS4.1.3. and from BSD/386 1.0 to BSD/386 1.1. Major OS changes such as from SunOS4.1 to Solaris 2.3 cannot be simplified with OMNICONF, because formats and locations of configuration files may be changed.

Repository on a Different Machine

The Feature

The repository can be placed on a different machine, which means OMNICONF can save and restore a configuration of a machine to and from another machine. In this case, /etc/omniconf/reposit contains the name of the machine that has its repository. Another perl script named "omniconfsrv" should be installed on the machine that will store the repository. When a configuration is stored on another machine, putconf invokes omniconfsrv via rsh instead of storing configuration files with cpio. Similarly, getconf will restore a configuration by invoking omniconfsrv via rsh.

Remote Configuration

Files and directories in a repository can be manipulated with file manipulation commands such as vi, chmod, chown, mkdir, etc. Getconf will transfer the manipulation to the machine whose repository was manipulated. This allows one to manage the configuration of a machine by manipulating its repository. For example, assume that the machine ''aries'' has its repository in /config/aries on the machine ''taurus''. If the administrator creates the directory /config/aries/mnt1 on taurus, then invokes getconf on aries, the directory /mnt1 is made on aries. Since OMNICONF handles all aspects of a configuration, anything concerning configuration can be done remotely.

Comparison with Other Systems

The Track[2] System propagates a configuration to uniformly configured machines on a large scale. If the same repository is shared among several machines, OMNICONF can also do such a thing. But since the repository machine's load may be rather high, it may be inadequate to use OMNICONF on a large scale under the current implementation.

Conclusions

OMNICONF is very useful for crash recovery and OS upgrades, but it has a few shortcomings: the contents of /etc/omniconf/area should be determined by a ''cut and try'' manner, and the putconf command takes a relatively long time to execute, but this system is worth using it. Using OMNICONF, an administrator can concentrate configurations of several machines on a single repository machine. He can then manipulate the configuration of any machine by making changes on the repository machine.

The author is using OMNICONF on SunOS4.1 and BSD/386. By the virtue of OMNICONF, the configuration procedure of a machine running BSD/386 is 1.1 completed almost instantly.

OMNICONF is still a premature system. It should be used in more systems and by more administrators to develop into a reliable and capable tool.

Availability

OMNICONF is a available from Information and Communications Lab., Matsushita Electric under a license agreement. Please contact the author of this paper for details.

Acknowledgements

The author would like to thank Yoshida Jun, manager, and Kushiki Yoshiaki, director of our lab for giving me the chance to develop OMNICONF. Special thanks are given to Utashiro Kazumasa of SRA for valuable suggestions on this scheme. The author also thank Ohtsu Takashi of our lab, who has patiently brushed up the English in this paper. Finally, the author really appreciates Kennedy LEMKE of Panasonic Technology Inc. for his proof reading.

Author Information

Imazu Hideyo (Imazu is his family name) earned a Masters degree in Computer Science from the Tokyo Institute of Technology in 1988. Since then, he has been working for Information and Communications Lab., Matsushita Electric as a network administrator. He can be reached via snail mail at Information and Communications Lab., Matsushita Electric, Osaka-hu Kadoma-si Kadoma 1006, Japan or electronically at himazu@isl.mei.co.jp .

References

[1] Larry Wall, Landal L. Schwartz, *Programming Perl*, O'Reilly and Associates, 1990.

[2] Daniel Nachbar, *When Network File Systems Aren't Enough: Automatic Software Distribution Revisited*, Proceedings of the Summer USENIX, Atlanta, GA., June 16-18, 1986.