2003－DSM－29 (2)
2003／4／25

UNIX運用支援システムについて

瀧山 真貴、平山 善一、藤村 直美
九州芸術工科大学

概要

近年の情報処理センターなどではPCにWindows系OSとUNIX系OSの両方を導入してサービスを行う例が増えている。Windows系OSでは日常的な運用を効率よくまとめて支援するシステムがある。一方、UNIX系OSでは日常的な運用を効率よくまとめて支援する効果的な仕組みがない。そのために一台当りはそれほど手間がかからない作業も、対象のPCが多いと容易には実施できない作業になる。ここでは多数台のPCのUNIX系OSにおける運用を支援する仕組みを実現し、センターで運用を試みたので報告する。

Implementation of Management Support System for UNIX

Masataka Takiyama, Zen'ichi Hirayama, and Naomi Fujimura
Kyushu Institute of Design

Abstract

Almost computer centers are now providing services for users with Windows and UNIX operating systems. In Windows system, some convenient tools are available for daily management. However no similar tools are available in UNIX systems. We implemented new tools to support daily management in UNIX systems. This is the report of the management support systems and the experience with them.

1 はじめに

現在、九州芸術工科大学情報処理センター（以後、本センターと略す）には120台のPCが設置されており、それぞれがWindows2000とRed Hat Linux 7.3のデュアルブート環境になっている。これらのシステムを管理するために、Windows環境では富士通から提供されている通称「セルフメインテナンスシステム」、商品名「瞬快」[1]というソフトウェアを導入し、OSのインストール、ソフトウェアのインストール、システムファイルの復旧などの作業を自動化している。他にも、例えばSymantecよりSymantec Ghost [2]のように同様の機能を有するソフトウェアが提供されている。

これに対してUNIX系OSとしてのRed Hat Linux環境（以後、Linux環境と略す）では、同様の作業を自動的に行うソフトウェアは実現されていない。そのため、各種の設定変更、ソフトウェアのインストール作業などを、本センターでは管理者が手作業で一台ごとに行っている。実際の作業に際しては、本センターの実習室の一般利用を休止し、1台ずつPCの電源を投入し、Windowsではなく、Linuxを選択して起動し、それぞれのPCで必要な作業を行うことになる。これには多くの時間と手間がかかり、作業ミスによるエラーの可能性もある。

センターなどで多数台のPCでサービスを提供していくためには、WindowsやUNIXなどをインストールする作業を行った後も、日常的なこまごました管理、運用が絶え間なく続く。これらは一台づつは簡単な作業でも、台数がまとまると簡単には実施できない作業量となる。そこでLinux環境におけるファイルの更新や追加、ソフトウェアのインストールなどを可能な限り自動化することで、管理者の負担を軽減し、作業時間の短縮を図るとともに、作業の確実性を高めるための仕組みを提案し、本センターで運用を試みたので報告する。

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

if test -f /home/center/MAINT/com/exec.csh
then
        /bin/csh /home/center/MAINT/com/exec.csh >&/dev/null
fi
```

図 1: rc.local

2 運用支援システム

センターの Linux 環境における日常的な運用は、作業内容の性質から二つの管理、運営作業に分けられる。一つは、各種ソフトウェアのパラメータを変更するためのファイルの追加や更新、ソフトウェアを追加するために rpm パッケージを展開するといった管理者が作業内容と手順を明確に把握している作業である。一方、管理者や利用者が意識せずに何らかの設定ファイルやシステムファイルを変更、削除してシステムを不具合にしてしまった場合、あるいは外部からの攻撃でファイルを削除、改竄された場合などの対応は管理者が何をどうしてよいか事前には明確ではない作業である。本節では作業内容が明確で日常的な前者の作業を支援する仕組みを提案する。

2.1 処理コマンドの自動起動

まず実行すべきコマンドを個別の PC に配布すること自体に手間がかかることから、ここでは NFS マウントで、すべての PC から見える場所にコマンドや必要なファイルを置く。そして各 PC は起動時にこのファイルに基づいて自動的に処理に必要なコマンドを実行する。そのために図 1 に示すように個々の PC の rc.local に運用支援を行うためのコマンド (exec.csh) を実行する行を追加しておく。この作業はすべての PC の Linux 環境で一回は行わなければならない。

2.2 処理コマンドの準備

図 1 で起動している exec.csh は例えば図 2 に示すような内容になっている。このコマンドでは、所定の作業場所に移動して、さらに詳細な処理を記述してある p*.csh を起動する。ここで p*.csh の * に対応する通し番号は複数の処理をする時に順番を保証するための通し番号として考えている。

```
cd /home/center/MAINT/com
csh p002.csh
```

図 2: exec.csh

複数のコマンドをまとめて図 2 に示したように作成するためのシェルコマンド listup.csh を図 3 に示す。このシェルコマンドは p*.csh と x*.cmd が組になっているものとして、p*.csh の一覧からコマンド列を作成する。その時に図 4 に示す sed サブコマンドを使用して、図 2 に示すようにコマンドの先頭に cd コマンドを追加している。

```
#!/bin/csh
unalias rm
cd /home/center/MAINT/com
ls -1 p* |sed -e 's/^/csh /' >temp.csh
sed -f head.sed temp.csh >exec.csh
chmod 755 *
rm temp.csh
```

図 3: listup.csh

```
1i\
cd /home/center/MAINT/com
```

図 4: 編集用 sed サブコマンド (head.sed)

2.3 具体的な処理の記述

p*.csh の 具体的な中身は図 5 に示すようになっており、ロックファイルを作成するためのディレクトリ、ログファイルを作成するためのディレクトリを作成し、具体的な処理を実行するために x*.cmd を最後に起動している。

具体的な作業を行うコマンドファイルの例を図 6 に示す。ここでは作業場所に移動して、date コマンドを実行しているだけである。実行結果を図 7 に示す。

```
cd /home/center/MAINT/com
date
```

図 6: 作業コマンドの例 (x002.cmd)

2.4 ロックファイルとログファイル

一つの処理 (p*.csh) を行うと対応したロックファイルを作成するので、一度処理を行ったら、ロックファイルを消去しない限り、二度と同じコマンドは起動されない。また処理結果は図 7 に示すように、ログファイルに記録されるので、処理が正常に終了したかどうかを容易に確認することができる。

```
csh x002.cmd -x
2003 年  4 月  3 日 木曜日 11:45:32 JST
```

図 7: ログ見本

2.5 管理者から見た作業手順

管理者は NFS マウントした所定の場所で、システムを更新するために必要なコマンド文をコマンドファイル（例えば x001.cmd）に記述する。そして実行ファイル（例えば p001.csh）にコマンドファイル名を記述し、listup.csh を実行する。これによって実行すべきコマンドリストが他に必要な処理手続きと併せて exec.csh として作成される。管理者が更新のために直接行う作業はこれだけで、後の作業は Linux 起動時に自動的に実行される。

ただし、実際にはセンターの PC に電源を投入するとマルチブートになっており、デフォルトでは Windows が起動する。したがって、現時点では電源を投入して Linux を選択して起動する部分は人手の介入が必要である。しかしながら、その後は自動的に必要な作業を行うことができる。したがって、作業を早急に行いたい時は管理者が全 PC の電源を投入し、Linux を起動することで、処理を行うことができる。一方、問題が軽微であり、処理を急がない場合には利用者が Linux を起動するまで待つことも可能である。

2.6 ログファイルとロックファイル

個々の PC が自動的に指定されたコマンドを実行する際に、何らかのエラーが発生して作業が正常に行われなかった場合、その PC を特定して問題を解決し、作業をやり直す必要がある。このシステムでは図 8 に示すように、ログファイルとロックファイルは実行ファイル名に基づいて作成されるディレクトリ内（例えば p002）に、作業を実行した PC ごとに"ホスト名.log" や"ホスト名.lock" というファイル名で作成される。

作業が正常に終了した場合と異常が発生した場合はログファイルの内容が異なるため、管理者は実行ファイルごとに蓄積されたログファイルの大きさを例えば 図 8 に示すように ls -l コマンドを実行するなどして、一見することで、異常のあったクライアントを簡単に特定できる。また、すべてのクライアントが作業を正常終了したことを確認した後、対応する実行ファイル、コマンドファイル、ロックファイル、およびログファイルを削除することで後日の不要な動作を省くことができる。

```
#!/bin/csh

set COMMAND='x002.cmd'
set LOCKDIR='/home/center/MAINT/lock'
set LOGDIR='/home/center/MAINT/log'
set TMPLOCKDIR='/tmp/MAINT_lock'
set TMPLOGDIR='/tmp/MAINT_log'

if ( -f $LOCKDIR/$0:r/`hostname -s`.lock ) exit 9
if ( ! -d $TMPLOCKDIR ) mkdir $TMPLOCKDIR
if ( ! -d $TMPLOCKDIR/$0:r ) mkdir $TMPLOCKDIR/$0:r
if ( ! -d $TMPLOGDIR ) mkdir $TMPLOGDIR
if ( ! -d $TMPLOGDIR/$0:r ) mkdir $TMPLOGDIR/$0:r

csh $COMMAND -x >& $TMPLOGDIR/$0:r/`hostname -s`.log
touch $TMPLOCKDIR/$0:r/`hostname -s`.lock

su - maint -c "cp -r $TMPLOCKDIR/* $LOCKDIR"
su - maint -c "cp -r $TMPLOGDIR/* $LOGDIR"
```

図 5: 作業コマンド起動手続き例 (p002.csh)

2.7 NFSに関する問題

現在、本センターのNFSサーバにNR1000という専用のファイルサーバ [3] を使用している。本センターで今回の仕組みを実装するにあたって、処理を実行するコマンド exec.csh は root 権限で動作させないとシステムの更新操作ができない。しかしながらデフォルトでは root 権限で NFS サーバ上のファイルに書き込むことができないために、ロックファイルやログファイルをそのままでは書き込めない。そこで、個々の PC ごとに root でも書き込めるように設定を変更することも検討したが、設定上の問題、セキュリティ上の不安などがあり、図 5 に示しているように、一旦それぞれの PC の /tmp の下にロックファイルとログファイルを作成し、最後に通常の利用者 (maint) としてコピーするようにした。

3 変更ファイル検出

管理者や利用者が意図していなくても、システムファイルを変更、削除する場合などがある。あるいは知らない間に外部からの攻撃によってファイルを改竄、削除されたりする。こうした何がおかしいか不明で、動作が異常なシステムを復旧する場合に、その原因を特定することは一般的に言って容易ではない。基本的にはファイルシステム全体を正常な状態に復元する必要があるが、そのためにはどのファイルがどのように変更されたかを漏れなく検索し特定する仕組が必要である。

管理者が変更されたことを把握していないファイルを正確に特定するためには、正常時のファイルシステムの情報を予め保存しておいて、異常発生時のファイルシステムの情報と比較することで、異常なファイルを特定することが合理的である。そのための方法として、いくつか考えられるが、処理時間、手軽さ、確実さなどを考慮すると次に述べる find コマンドを使用する方法が良いと考えている。この方法で異常になったファイルを特定できれば通常の方法で修復することが可能である。

最初にシステムが正常なときに次に示すコマンドを実行する。

```
find / |xargs ls -ld |awk '{print $5,$6,
    $7,$8,$9,$10,$11}' |sort >listfile-1
```

その結果、図 9 に示すようにシステム内の全ファイル及びディレクトリの"ファイルサイズ"、"更新日時"、"フルパスのファイル名"のリストを作業用ファイル listfile-1 に保存する。

次に異常が見つかったときに同じコマンドを実行し listfile-2 など別のファイルにリストを保存する。こうして得られた 2 つのリストを

```
ipccl120 $ ls -l lock
合計 8
drwxr-xr-x    2 maint maint     4096  3月 31 17:26 p001
drwxr-xr-x    2 maint maint     4096  4月  3 11:45 p002
ipccl120 $ ls -l lock/p002
合計 0
-rw-r--r--    1 maint maint        0  4月  3 11:45 ipccl120.lock
-rw-r--r--    1 maint maint        0  4月  3 11:35 ipccl121.lock
ipccl120 $ ls -l log
合計 8
drwxr-xr-x    2 maint maint     4096  3月 31 17:26 p001
drwxr-xr-x    2 maint maint     4096  4月  3 11:45 p002
ipccl120 $ ls -l log/p002
合計 8
-rw-r--r--    1 maint maint       53  4月  3 11:45 ipccl120.log
-rw-r--r--    1 maint maint       53  4月  3 11:35 ipccl121.log
ipccl120 $
```

図 8: ログファイルとロックファイルの例

```
0 Apr 1 17:51 /dev/initctl
0 Apr 11 2001 /home/src/DONE/auctex-10.0g/style/.nosearch
0 Apr 17 2002 /usr/share/doc/libmrproject-0.6/NEWS
0 Apr 2 08:04 /.autofsck
0 Apr 2 08:04 /dev/log
0 Apr 2 08:04 /dev/shm
```

図 9: listfile–1 の先頭部分

```
diff --suppress-common-lines
                    listfile-1 listfile-2
```

として比較すると、図 10 に示すように変更、追加、削除されたファイル及びディレクトリをすべて一覧表示することができる。図 9 で示した例は日頃使用している Red Hat Linux 8.0 を導入しているノートパソコンにおいて実行した結果を示している。listfile–1 と listfile–2 を作成する間に、この研究会の tex の原稿を処理するために jlatex を実行している。listfile–1 と listfile–2 の大きさは約 157400 行で、800 MHz の P3 Mobile の CPU で listfile–1 の作成に約 2 分 23 秒、listfile–2 の作成に約 1 分 24 秒の経過時間、CPU 時間に両方とも約 11 秒を要している。ほぼ連続して同じコマンドを実行しているので、関連情報がメモリにキャッシュされ、経過時間が短縮されていると考えられる。いずれにしても、数分で全ファイルシステムの情報を収集することができる。

今回の listfile–1 と listfile–2 の diff の結果は 17 行が異なることを示している。この方法では、正常時のリストをこまめに更新しておかないと、膨大な量の変更点が発見され、逆に必要な情報を得るのが難しくなる場合があるので注意が必要である。

4 おわりに

本研究で作成したシステムで、センターの Linux 環境におけるファイルの更新、ソフトウェアのインストールなど、システムの運用作業を自動化する見通しが立った。これによってシステム管理者の負担を軽減し、作業時間の短縮を図るとともに、作業の確実性を高めるための仕組みが実現されたと考えられる。予め処理をコマンドとして準備しておくと、後は電源を投入し、Linux を選択して、処理が終われば電源を切断すれば良いということは、急がない修正の場合には利用者が Linux を起動するまで待つことも可能で、大幅な省力化を実現したと言える。

```
48752a48753
> 21219 Apr 2 14:37 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.tex
50179d50179
< 21811 Apr 2 14:32 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.tex
50245c50245
< 2185 Apr 2 14:32 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.aux
---
> 2185 Apr 2 14:37 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.aux
54518a54519
> 23144 Apr 2 14:37 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.dvi
56039d56039
< 23648 Apr 2 14:32 /home/fujimura/FILE/tex/20030315-DSM-Self/jgen.dvi
```

図 10: diffによって発見された更新ファイル（一部）

しかしながら、Windows 系 OS の同種の目的のソフトウェアに比較するとまだまだできることは限られている。電源の投入で Windows と Linux を自動的に選択して必要な処理を行う機能、まっさらな PC に Windows と Linux をインストールする機能の実現など、これから実現したい機能も多くある。さらにより広範囲の UNIX 系 OS に対応できるようにもしたい。これらについては今後の課題としたい。

参考文献

[1] 瞬快 : http://www1.infoeddy.ne.jp/ftk/shunkai/

[2] Symantec Ghost :
http://www.symantec.com/sabu/ghost/

[3] NR1000 : http://storage-system.fujitsu.com
/jp/products/nearstore/nr1000r100/