

Presentation #: 5060
An introduction to System Maintenance Functions
for HP9000 Servers
F. John Kluth
Kluth Company
1060 DeLeone Dr.
Kent OH 44240-2026

UNIX began its long and glorious history as an operating system that was designed for technically literate users. Though documentation is plentiful, and comparatively well organized; still, it lacks convenient entry points for beginning users. This paper attempts to fill in some gaps, especially for those persons who become system administrators, perhaps unintentionally. These persons will want some basic rules to help organize an HP9000 server for convenient and efficient use.

UNIX has a fairly daunting scripting language which is surprisingly complex and cryptic. The tendency seems to be for users to try to avoid this and set up a windowing interface to give the user something more familiar and easy to use. The SAM system administrator interface is an example of this. SAM is often a convenience, but it is not perfect. And it helps to know how SAM works and verify what it has done. An example of this in HPUNIX 9.0 that I have found is that SAM manages the group file, but not perfectly. When a new user is created, the appropriate group file entry is created, but when that user is deleted, the entry in the group file is not deleted. These entries can be verified and corrected with standard shell tools. Typically one must apply one of the UNIX editors, vi, ex, ed, or even emacs to the etc/group file to check the data and correct any errors.

The simplest approach to assisting the user is to provide a log in with a menu that allows choice amongst the various programs that can be run. There are even windowing shells that can be provided to help satisfy this

An Intro to System Maintenance Functions for HP9000
Servers

5060-

craving for a windowing interface. But from another point of view this attempt to ease computer anxiety, may, in the long run, lead to a dependence on mechanisms that are cumbersome and inefficient. Menus are convenient the first time round, but after the thousandth use, they just get in the way. The Bourne shell case command seems ideally constructed to develop menus, and it is often so used. But a straightforward use of the case statement locks the user in a closed loop and isolates the user from the power of the shell.

A study of the Bourne shell will reveal a number of insights that will be helpful here. Of the various available shells it is the most cryptic. What you get for being cryptic is speed. An alert programmer will be able to write powerful programs that run faster on the system.

This is of value to the systems administrator because the user needs routines at log in that will produce quick and effective results. Time lost at log in is always a problem because it is repeated thousands of times as the user continues to log in. This problem is aggravated by vended programs which charge by number of concurrent users. If the log in process is cumbersome then users will log in and not log out during lapses in activity. The concurrent licenses will then keep track of users that are doing nothing.

The basic command structure seems to be the fastest for getting applications up and running and this is the method that should be preferred. For each program to be run the user is provided with a short command that executes the desired program. For example to run Lotus 123 on the HP9000 the user is provided with the command 123. This is done by providing the following file with execute permission for the user:

```
1  #!/bin/sh
2  /usr/lotus/123.v12/hp700/bin/123
```

This file can be placed in the users directory if it needs to be personalized for the user, or it can be placed in a

An Intro to System Maintenance Functions for HP9000
Servers

directory such as /usr/contrib/bin which is available to all users. The first line of the file indicates to the system that the file is indeed a Bourne shell executable file. The second line causes the Lotus 123 program to execute. The convenience is that the user now types many fewer keystrokes. The next consideration has to do with what lotus uses as the default path. If the user logs in and executes 123 then Lotus 123 will assume the default path is the home path of the user. If the user also uses other programs, such as WordPerfect, then it might be better not to mix the file types of the applications. In this case a line can be added before line 2 in the previous file to set the default path to a user subdirectory such as lotus.d:

```
1  #!/bin/sh
2  cd $HOME/lotus.d
3  /usr/lotus/123.v12/hp700/bin/123
```

This file assumes that the directory lotus.d exists. The system administrator may have to create the file for the appropriate users, or include additional programming that does this automatically.

A number of such applications may be available to the user. It would be helpful to remind the user what options are available. A menu would be useful here. Menus can be provided that can also be executed by simple commands.

Since help is a UNIX command, it should not be used to name a help menu. A command like ahelp could be used. In the case of primary help screens a single letter or number might be desirable such as h or H or 1. The following is an example of file h that when executed generates a simple help screen:

```
1  #!/bin/sh
2  echo To run a program type the command at the prompt
   and press enter.
3  echo
4  echo Command | Program
5  echo -----|-----
```

An Intro to System Maintenance Functions for HP9000
Servers

5060-

```
6 echo 123          | Lotus 123
7 echo wp           | Wordperfect 5.0
8 echo sym          | SYMIX 4.0
9 echo
10 echo To log out type the command quit at the prompt and
    press enter.
```

After the file executes the prompt appears and the user can enter one of the commands that appears on the screen.

More complex help screens can be developed which refer to other screens for more advanced help. Typically UNIX users now desire help that is more basic than the man command. They appreciate having lists of basic commands that they can use. Once the suitable command is identified, then the man command would be useful for most situations.

Setting of the terminal variable is quite basic to the log in and is sometimes relegated to a menu. Unfortunately, the method just described cannot be used to set the TERM variable because a subshell cannot change the parameters of the parent shell. The menu would have to be a part of the .profile file that sets the parameters for the particular user. Bourne shell code for such a menu follows:

```
1  #!/bin/sh
2  # Set up terminal
3  clear
4  nd=n
5  until test $nd = y
6  do
7      echo "Indicate which terminal you are using:"
8      echo
9      echo "1 - hp 700/96"
10     echo "2 - reflections"
11     echo "3 - kermit"
12     echo "4 - wyse 60"
13     echo "5 - vt100"
14     echo
15     echo "menu > \c"
```

An Intro to System Maintenance Functions for HP9000
Servers

```

16  read input
17  case $input in
18      1 )  nd=y;TERM=2392;export TERM
19           stty erase "" kill "^U" intr "^C" eof "^D"
20           ;;
21      2 )  nd=y;TERM=2392;export TERM
22           stty erase "" kill "^U" intr "^C" eof "^D"
23           ;;
24      3 )  nd=y;TERM=vt320;export TERM
25           stty erase "" kill "^U" intr "^C" eof "^D"
26           ;;
27      4 )  nd=y;TERM=wy60;export TERM
28           stty erase "" kill "^U" intr "^C" eof "^D"
29           ;;
30      5 )  nd=y;TERM=vt100;export TERM
31           stty erase "" kill "^U" intr "^C" eof "^D"
32           ;;
33      * )  clear;echo Bad entry! ;;
34  esac
35  done

```

The erase control character in the case of 1, 2, and 3 is ^H while the erase control character in the case of 3 and 5 is ^?. This code is appended to the .profile code.

While this menu approach works better than the method of entering the TERM variable manually, there are still problems. If the users hits the wrong choice, then the user locks the terminal in a way that is quite confounding. Typically the user's processes must be killed before another log in can be attempted. A better approach is to use the ttytype command in the .profile file so the TERM variable can be set automatically. The default HP-UX profile contains an example of this. Care must be exercised with this approach because the ttytype command often returns terminal names which are abbreviated. Vended applications sometimes come with their own terminfo file that must be edited so the abbreviation is recognized.

In addition to help files, data bases can be set up

An Intro to System Maintenance Functions for HP9000
Servers

5060-

to manage various types of information that are useful to the user. Shell script data bases are surprisingly versatile. Flat file data bases can be set up with a dozen fields and 50,000 records that can be quickly accessed with the grep command. In a shell script data base field lengths are variable with a total maximum record length of about 1500 characters. Since the data is searched serially by grep, data bases that are too large will simply take too long to search. Though hashing techniques are not strictly available to shell scripts, larger data bases can be accommodated with techniques involving string sorting, file division, and path searching. Shell scripts are quite convenient and powerful for small data bases, especially where data structures are indeterminant. Shell scripting is a good prototyping tool. As the data structure becomes more fixed, and as the data base gets larger, more formal data base engines should be considered.

Typical UNIX shell script data bases involve variable length fields with character delimiters which are by default either a space character or a tab character. Within fields an underscore character can be used to separate words. Shell scripts which manipulate data base information must be carefully constructed so that string variables are appropriately quoted. Shell control characters such as backslash, apostrophe, and quote can cause difficulty if included in fields. In spite of these limitations the search features of the Bourne shell are quite powerful and useful, even if cryptic. Even using the grep command relational data base constructs can be achieved. The relational constructs are easiest if the fields related are first or last, but there are other techniques for intermediate fields. The awk command can be used, as well as reading all fields into variables.

Shell script data bases have the advantage of having the availability of many tools to work with them. Data can be entered or patched with vi or sed. Data can be reorganized with cut, paste, and sort. Commands such as tr and sed are also useful. In addition there are other

structures which can be used for data that fails to fit the standard mold. For example text data can be stored in a file with a keyword filename. The ls command can then search a directory for a keyword such as "bottle" with a structure such as "ls b*". Once the keyword is found it can be listed with the more or pg command. In this same directory the command structure "grep bottle *" will search every file for an instance of the word "bottle" and will list all files that contain it. UNIX contains powerful data base features which are often overlooked.

Shell script data bases should be used by system administrators to construct data bases of data that assists users. Phone numbers for company employees, vendor and customer names, equipment catalogs, and LAN addresses are all examples of suitable data. Maintenance information for company computer equipment is especially relevant. Service calls can be minimized if a record is made of these calls that can be searched by problem topic.

Artificial intelligence is the name applied to such a data base constructed of frequently asked help desk questions.

In fact one of the most important of such data bases is a list of maintenance procedures for the system. This data base is the concern of the system administrator and not the user. This file should include daily, weekly, and monthly routines, as well as system descriptive information, and who to call in case of various types of emergencies. This data is then available to any remote user, as well as in a position to be saved to tape, so that a disaster that destroys the hard drive will not destroy the data. Maintenance routines can often be automated, so this data will describe how to start the automatic routines at a regular interval.

Monthly routines involve administrative information relating to disk usage, user information, data size. The purpose of this information is to provide information relating to system usage to other managers. Application of the bdf command provides disk usage. A user count can

provided by applying `wc -l` to the file `/etc/passwd`. The number of administrative log ins must be subtracted from the count to give a true user estimate. Important data bases need to be identified and treated separately. Shell script data bases can be monitored with `wc -l` command. Data bases maintained by data base engines require the use of the `du` command. A shell script can be written which automatically executes the command and prints a report to send to the other managers.

Weekly routines involve information relating to disaster recovery, backup, and security. An example script file follows:

```
1  #! /bin/sh
2  # file monmor created 19940516 by F. John Kluth.
3  # Checks system parameters.
4  echo Printing files changed since Friday morning.
5  find / -depth -mtime 3 -type f | pr -o5 -h "Files
Changed since Friday" > monmor.tmm
6  cat monmor.tmm | lp -o -nb -o -10
7  echo Printing computer usage report.
8  who /etc/wtmp | sort | pr -w140 -3 -o5 -h "Weekly
Computer Usage Report" > monmor.tmn
9  cat monmor.tmn | lp -o -nb -o -c
10 rm /etc/wtmp
11 touch /etc/wtmp
12 chmod o-rw /etc/wtmp
13 echo Working on bad login report
14 strings /etc/btmp | sort | pr -w140 -5 -o5 -h "Bad
Login Report" > monmor.tmo
15 cat monmor.tmo | lp -o -nb -o -c
16 rm /etc/btmp
17 touch /etc/btmp
18 chmod o-rw /etc/btmp
19 echo "Setuid Programs-programs that do not match
programs from last week" > monmor.tmp
20 cp suid.lst suid.old
21 find / -hidden -perm -4000 -exec ls -ld {} \; >
suid.lst
22 diff suid.lst suid.old >> monmor.tmp
```

An Intro to System Maintenance Functions for HP9000
Servers


```

23 echo >> monmor.tmp
24 echo "Empty Password Fields:" >> monmor.tmp
25 awk -F: '$2 == "" || $2 == ",..." < /etc/passwd >>
monmor.tmp
26 echo >> monmor.tmp
27 echo Copying files to /users/sysadm/sys.d
28 cp /etc/passwd /users/sysadm/sys.d
29 cp /etc/group /users/sysadm/sys.d
30 cp /etc/lvmtab /users/sysadm/sys.d
31 cp /etc/lvmconf/* /users/sysadm/sys.d
32 cp /etc/checklist /users/sysadm/sys.d
33 echo "\nPreparing backup information. \n"
34 echo "\n\n\nBackup Information for use in System
Recovery. `date`\n\n\n" >> monmor.tmp
35 echo "\nOutput of ioscan *** \n" >> monmor.tmp
36 /etc/ioscan -fk >> monmor.tmp
37 echo "\nOutput of vgdisplay *** \n" >> monmor.tmp
38 /etc/vgdisplay >> monmor.tmp
39 echo "\nOutput of lvlnboot *** \n" >> monmor.tmp
40 /etc/lvlnboot -v >> monmor.tmp
41 echo "\nOutput of lvdisplay *** \n" >> monmor.tmp
42 /etc/lvdisplay -v /dev/vg00/lvol1 >> monmor.tmp
43 /etc/lvdisplay -v /dev/vg00/lvol2 >> monmor.tmp
44 /etc/lvdisplay -v /dev/vg00/lvol3 >> monmor.tmp
45 /etc/lvdisplay -v /dev/vg00/lvol4 >> monmor.tmp
46 echo "\nOutput of pvdisplay *** \n" >> monmor.tmp
47 /etc/pvdisplay -v /dev/dsk/c0d0s2 >> monmor.tmp
48 /etc/pvdisplay -v /dev/dsk/c3d0s2 >> monmor.tmp
49 echo "\nOutput of file checklist *** \n" >> monmor.tmp
50 cat /etc/checklist >> monmor.tmp
51 echo "**** end of checklist ****" >> monmor.tmp
52 echo "\nBackup the LVM configuration. *** \n"
53 /etc/vgcfgbackup /dev/vg00
54 echo "\nUpdate the boot structures. *** \n"
55 /etc/lvlnboot -R
56 echo "\n *** End of Configuration printout ****" >>
monmor.tmp
57 echo End of monmor program
58 lp -o -nb -o -c monmor.tmp

```

The find command in line five is useful for finding

An Intro to System Maintenance Functions for HP9000
Servers

5060-

which files are being used and which are not. This command needs to be modified for the particular system used. The find -atime could be useful in a system where only a few files are accessed and the find command itself is not often used, since find resets the -atime value. The -mtime value is more useful for finding files which are growing without bound. The -ctime value is more useful for finding which files have had their permissions changed. Reviews of changes to system files can help boost security.

Lines 9 through 12 print out a listing of all log ins during the past week. This report can be used to find out if users have logged in at unauthorized times. It does not report the length of time the user logged in. The bad login report is used to determine if a port receives unwarranted attention. Excessive log ins suggest passwords may be being guessed. Ordinarily setuid programs do not change. Any change in these programs suggests a security breach. Lines 19 through 23 cause a display of any changes in these programs. Lines 24 and 25 check for bank password fields that could give the user extraordinary access. Lines 27 through 32 cause system file information to be saved outside of the file system reserved for the operating system. This facilitates recovery in the event of a disaster. The lines following 34 are recommended by HP for the benefit of the CE (computer engineer) should there be a system disaster.

In addition to running the above script, weekly maintenance routines include cleaning the tape drive and recovering a file off the tape. To recover a file off the tape provide a dummy file with some regular pattern that can be visually checked for errors. Once the file is saved to tape in a regular backup, the dummy file is erased from the hard drive. The recovery routine is executed and the recovered file is visually inspected for errors. This procedure does not guarantee that a full backup will restore the system, but it does indicate that the tape drive is functioning properly.

The following is an example of a shell script that can be run daily:

```
1  #! /bin/sh
2  # File sysck created 19940518 by F. John Kluth.
3  echo Machine info: > syscka$$
4  uname -a >> syscka$$
5  echo >> syscka$$
6  echo Capacity should not exceed 90% >> syscka$$
7  bdf >> syscka$$
8  echo >> syscka$$
9  echo System messages collected by dmsg since last run:
>> syscka$$
10 dmesg - >> syscka$$
11 echo >> syscka$$
12 echo boot time: >> syscka$$
13 who -b >> syscka$$
14 echo >> syscka$$
15 echo printer state: >> syscka$$
16 lpstat -t >> syscka$$
17 echo >> syscka$$
18 echo Location of core files: >> syscka$$
19 echo >> syscka$$
20 find / -name core -print > sysckb$$
21 if test -r sysckb$$
22 then
23     while read flnm
24     do
25         echo "For file $flnm:" >> syscka$$
26         ll $flnm >> syscka$$
27         strings $flnm | grep tty[01d][0p] >> syscka$$
28         echo >> syscka$$
29     done < sysckb$$
30     rm sysckb$$
31 fi
32 echo >> syscka$$
33 echo Network configuration >> syscka$$
34 ifconfig lan0 >> syscka$$
35 echo Note: PROMISC should not be evident >> syscka$$
36 echo Log file sizes >> syscka$$
37 wc /usr/adm/sulog >> syscka$$
38 wc /usr/sam/log/samlog >> syscka$$
```

An Intro to System Maintenance Functions for HP9000
Servers

```

39 wc /usr/sam/log/br_log >> syscka$$
40 wc /usr/sam/log/br_index.full >> syscka$$
41 wc /usr/lotus_log/lotus/log/123usage12.log >> syscka$$
42 wc /usr/spool/lp/log >> syscka$$
43 wc /usr/lib/cron/log >> syscka$$
44 wc /usr/lib/cron/OLDlog >> syscka$$
45 wc /tmp/index >> syscka$$
46 echo "Note: /tmp/index is used by tape to hold tape
index." >> syscka$$
47 wc /usr/adm/msgbuf >> syscka$$
48 echo >> syscka$$
49 echo Who is on the system: >> syscka$$
50 who >> syscka$$
51 echo >> syscka$$
52 echo Security on following files should be -r--r--r--
>> 53 syscka$$
54 ll /etc/passwd >> syscka$$
55 ll /etc/group >> syscka$$
56 ll /etc/hosts >> syscka$$
57 ll /etc/inetd.conf >> syscka$$
58 ll /usr/adm/inetd.sec >> syscka$$
59 ll /etc/ftpusers >> syscka$$
60 ll /etc/securetty >> syscka$$
61 echo Files modified since yesterday >> syscka$$
62 find / -depth -mtime 1 -type f >> syscka$$
63 cat syscka$$ | pr | lp -o -nb -o -10
64 rm syscka$$

```

The uname command in line 4 merely identifies the system you are using. This serves as a header for what follows. The bdf command in line 7 repeats information that is printed weekly. Daily monitoring of this information is more appropriate. Rapid changes in the results of this command are symptoms of many serious problems. Since the system will not run with a full disk, it is desirable to constantly monitor this situation. The dmesg command in line 10 is not as useful as it sounds. Certain system error messages are monitored by dmesg. These error messages always appear in another form, such as on the system monitor. Running the command prints what is in the error buffer and then empties it. Running dmesg

keeps the buffer from being too large. The error messages saved are not usually very interesting. If the system is rebooted from the console then dmesg contains an image of the boot messages. If the system is booted remotely, no error messages are saved by dmesg.

In line 13 the command "who -b" gives the time of the last reboot. Rebooting is a security issue and should be controlled carefully. An HP-UX machine can be rebooted remotely so care should be taken that each reboot be accounted for. The output of the who -b command should be matched with a verifiable boot time. The command "lpstat -t" in line 16 is useful for identifying hung printers or print jobs. If the command is run early in the morning, then printers having problems from the day before will be revealed. Printers with many files in the spool can be investigated. Most problems will be more quickly revealed by the pained cries of the users, but this is not always the case.

The code about core files in lines 20 to 32 is more interesting. Files produced by a system core dump are the most mystical and cryptic of UNIX. Of course UNIX computers do not have cores anymore so the base meaning of the term has changed. In the past, the bigger computers had a special sort of permanent memory called core. This memory was intimately connected to the operation of the machine and central to its function, hence the term core.

Some of the function of core memory has been taken over by the hard drive, but most has been taken over by RAM. The core dump is a snapshot of what the computer was doing at the time of a special incident. The purpose of the core file is to provide a record of the incident for the computer engineer. In the past, these dumps were more significant, in terms of diagnosing difficulties with the computer system. Now the UNIX system is so well constructed that they are rarely used. But they can be used more than they are. The ordinary system administrator sees these files as an inconvenience, because they are often large, and need to be erased. To the alert administrator they contain important

information. The program in lines 20 to 32 presents that information in a more readable format. For each core file the program generates the time it was created, and the login of the user whose program created it. The label of the serial line used is also generated. Using this information, the system administrator can observe trends that suggest system problems such as persons who do not log in and log out properly, and programs that are poorly adjusted to the system. Unfortunately, the difficult question must sometimes be asked, "What were you doing at 11:53 am yesterday?"

The LAN inquiry command at line 34 is intended to check the LAN configuration. The PROMISC configuration checked for allows a hacker access to all messages on the LAN. The size of log files needs to be monitored so the size does not get too large. Active log file can be found with the daily application of the find command as used in lines 60 and 61. Once log files are identified their size should be monitored as is done in lines 36 to 47 above. Application of the who command as in line 50 identifies who is currently logged in. Log ins are dated so it should be obvious who logged in the day before and who never logged out. These situations need to be investigated. The lines 52 to 60 are for checking security on files important for system security. Finally the find command is used in lines 62 and 63 to find files that have been modified. Note that checking for -atime, -mtime, or -ctime may be more important depending upon the system.

AS was demonstrated, shell scripts can be used to increase the convenience of the user. They can also be used to print reports that monitor the state of the system. Shell scripts can also verify that the sam program has worked properly. They can run the backup, but the system administrator must manually recover a file to verify the functionality of a tape. An understanding of shell scripts allows system configuration files to be modified, but any modified system file must be carefully documented with a copy stored apart from the operating

system. Printer and other customized drivers should be copied and saved separately. In the event of a hard drive disaster a full backup might be used to restore the system, but in the event of a disaster where the cpu must be replaced, the operating system cannot be restored off the backup tape.

Bibliography:

1. Blinn, Bruce, PORTABLE SHELL PROGRAMMING, Prentice Hall PTR, 1996, 269 pages, diskette included.
2. Kluth, F. John, "Shell Script Data Base Modeling", HPWORLD '96 PROCEEDINGS, Interex, August 4-9, 1996, pp 3012-1-9.
3. Kluth, F. John, "System Disaster Recovery", INTERWORKS '97 PROCEEDINGS, Interex, April 12-17, 1997, pp 735-9.
4. Nemeth, Evi, Garth Snyder, Scott Seebas, Trent R. Hein, UNIX SYSTEM ADMINISTRATION HANDBOOK, Prentice Hall PTR, 1995, 779 pages, includes CD ROM.
5. Peek, Jerry, Tim O'Reilly, Mike Loukides, UNIX POWER TOOLS, O'Reilly & Associates/Bantam Book, March 1993, 1119 pages, includes CD ROM.
6. Poniatowski, Marty, "HP-UX Audit Overview", HP-UX/USR, march/april 1997, pp 28-34.
7. Poniatowski, Marty, HP-UX 10.X SYSTEM ADMINISTRATION, Prentice Hall PTR, 1996, 382 pages.