HP-UX for MPE Users

A Robelle Tutorial

Neil Armstrong, August 1997

robelle solutions technology

1

For Techies

Welcome to *HP-UX for MPE Users*. This tutorial is given by Neil Armstrong, programmer for the Robelle R&D team.

This paper is intended to relate the MPE commands and concepts that you are familiar with to similar commands and concepts on HP-UX. This will not be a debate on the merits of MPE/iX versus HP-UX. It is intended to assist the MPE user, programmer, and operator to navigate and use commands on the HP-UX operating system.

Today we will discuss the various shells within UNIX and shell programming, the most basic features of HP-UX, manipulating files, and understanding the file system.

References

Robelle Solutions Technology Inc.
Suite 201, 15399 - 102A Avenue
Surrey, B.C. Canada V3R 7K1

Toll-free: 1.888.ROBELLE
1.888.762.3553
Telephone: 604.582.1700
Fax: 604.582.1799
E-mail: support@robelle.com
WWW: www.robelle.com

What's Inside

- Help from HP-UX
- Shells in HP-UX
- Logging on
- Variables
- File system
- Simple commands
- Searching files
- Regular expressions
- I/O redirection
- Pipes
- Hereis documents
- Shell programming
- Cron/Job streams
- Command summary

2

This paper is designed to give you an overview of how to navigate your way around HP-UX, and how to accomplish some of the everyday tasks that you may perform on MPE.

Questions are encouraged from the audience at any time.

Neil Armstrong has been a member of the R&D team for two of his four years at Robelle. His focus is on Suprtool, in which he not only tracks down and eradicates bugs, but also implements new features. Some of the enhancements Neil has added to Suprtool include progress messages, packed data-type date support, and the new Allbase interface for both the MPE and HP-UX versions of Suprtool. In his spare time Neil plays with his two young sons, rides his mountain bike, and coaches and plays soccer in the small town of Blackfalds, Alberta.

For Techies
All examples are from HP-UX 9.04.

References

Help me man

- Help = man
  - man [-] [section[subsection]] entry_name
- “Manual Pages” are divided into these sections:

1	User Commands
1M	System Maintenance
2	System Calls
3	Functions and Function Libraries
4	File Format
5	Miscellaneous
7	Device Files
9	Glossary

3

On MPE we had the Help command to find out the meaning of and syntax for various commands. On UNIX we have the man command. This is used to display on-line documents, often referred to as the “man pages”.

The man pages are divided into logical sections. Sections 6 and 8 are not included on HP-UX. Section 6 is reserved for Games and Section 8 is sometimes classified as Administrative or Maintenance Commands.

For Techies

References

Help me man — keyword search

- Keyword searches
  - **`man -k stty`**
- Man page index file
  - **`/usr/lib/whatis`**
- Create by doing a catman -w command
  - **`/etc/catman -w &`**
- Where does man search?
  - MANPATH variable
- Navigating man pages
  - Spacebar to see next page
  - Q to stop

4

To determine what section you want to read or to get a summary of all the entries with a particular keyword, you can use the -k option of the man command.

This option only works if you have a man page index file, which can be found in /usr/lib/whatis. If your system does not have this file, you can create it easily by logging on as root (superuser) and then doing a catman -w command.

This process takes a long time to run, so we suggest that you run it in the background by putting an "&" at the end of the command.

The man command searches the directories for the man page entries in the order described by the MANPATH variable. This is just like the HPPATH variable on MPE, but it specifically describes the order in which the man command should search for specific man pages.

You can see the next page of a man page listing by pressing the spacebar. You can stop a man page listing by pressing "q".

For Techies

References

Man page examples

```
$man -k stty
    stty(1)       {set the options for a terminal port}
    stty(7)       {terminal interface for Version 6/PWB compatibility}
    stty, gtty(2)    {control device}
```

5

For Techies

References

CI see a shell

- MPE has the CI and POSIX shells
- HP-UX has four shells:
  - Korn shell (ksh)
  - POSIX shell
  - Bourne shell (sh)
  - C shell (csh)
- Korn shell most MPE-like

6

The Command Interpreter on MPE is built-in, but on HP-UX it's just another program. As a result, UNIX systems have many different command interpreters, commonly referred to as shells.

sh
The Bourne shell, whose default prompt is "$", is the oldest of the shells and is on every UNIX system. It has no job control, is the default shell assigned to root, and is commonly used for writing command files or "scripts". On HP-UX 10.X, sh runs the POSIX shell.

ksh
My personal choice for shells because it is compatible with the Bourne shell and has many features of the "C" shell. Features of the Korn shell include command history editing, line editing, filename completion, command aliasing, and job control.

csh
The C shell has a default prompt of "%". It was developed at Berkeley and has lots of tricky features.

One confusing part of UNIX systems is that some commands may be specific to a shell while other commands, such as rm and cp, are programs unto themselves. One way of identifying a built-in command is by checking whether it has its own man pages, so you have to read the man page of your shell to find the built-in commands. For example,

For Techies

References

`man ksh` {list out ksh info and

Logging on

- getty program displays /etc/issue and displays login prompt
- Login validates username and password
- Login then runs the shell you have defined
- Korn, Bourne, and POSIX shells execute commands in these files:

  `/etc/profile` {global commands to all users}

  `.profile` {your own version}
- C shell

  `/etc/csh.login` {global commands to all users}

  `.login` {your own version}

7

What happens when you log in to an HP-UX system?

When you press Return, the getty program wakes up and first displays the contents of the /etc/issue file followed by the login prompt, and then runs the login process.

Not only is login responsible for validating the username and password, but it places the user in the home directory and starts the default shell defined in the /etc/passwd file.

At this point, what happens depends on what shell you have configured. For the Bourne, Korn and POSIX shells, the contents of the file /etc/profile is executed globally for each user.

The shells also have the ability to execute the commands from local files in your home login directory. All the shells have the ability to execute the .profile file. The POSIX and Korn shells can also execute a second local file called .kshrc.

The C shell also has both a global and a local file which it executes. The names are /etc/csh.login for the global commands and .login as the local filename.

For Techies

References

Sample /etc/profile

```
# Default system-wide profile (/bin/sh initialization)

PATH=/bin/posix:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bi
n
MANPATH=/usr/man:/usr/contrib/man:/usr/local/man

if [  "$TERM" = "" ] # if term is not set then set
then
     TERM=hp
fi
EDITOR="qedit -s -cvi"
export PATH MANPATH TERM EDITOR
stty erase "^H" intr "^Y" eof "^E" kill "^U" swtch "^Z"
```

8

"#" usually indicates the beginning of a comment.

For Techies

References

Sample .profile file

My personal .profile file

Set the Qedit RcrtModel variable to detect my terminal

```
export RCRTMODEL=2
```

Setup my prompt to be my current working directory

```
PS1='$PWD'
```

9

For Techies

References

Variables

- System variables = Environment variables
- Typical MPE variables
  - HPPATH, HPGROUP, HPACCOUNT
- Typical HP-UX variables
  - PATH, TERM, MANPATH
- Setting variables in MPE
  - setvar HPPATH "cmd.sys,cmd.util,cmd.test"
- Setting variables in Korn shell
  - PATH=/bin/posix:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin
  - export PATH

10

For Techies

We are used to the concept of system variables on MPE. On HP-UX we have a similar concept called environment variables. As the name implies, these variables describe your environment. There are some common variables that many tools rely upon in order to function properly.

Some typical MPE variables are HPPATH, HPACCOUNT, and HPGROUP. Some very common environment variables on HP-UX are the PATH, TERM, and MANPATH variables.

The PATH variable is the same as the HPPATH variable on MPE.

The TERM variable describes the type of terminal that you are using. This variable is often used by programs that use the Curses library, such as vi.

The MANPATH variable is similar to the PATH variable, but is used as a search order for the man utility to find the various pages.

These variables must be "export"-ed in order for any child process to reference them.

References

Variables continued

- Referencing variables
- MPE
  - **`echo !hppath`**
- Bourne/Korn/C shells
  - **`echo $PATH`**
- Inheritance

11

On MPE we can reference a variable by preceding the name of the variable with an exclamation point. On HP-UX you must precede the name of the variable with a dollar sign ($).

Environment variables can be inherited by a child process from the parent. The parent process, however, cannot reference variables created or changed by the child process once the parent process becomes active again.

For Techies

References

Variables in the real world

- Set system-wide variable via global Profile files
  - **`NTDEV="beano.robelle.com"`**
  - export NTDEV
- Reference that system-wide variable in all scripts
  - **`FTP -n -v $NTDEV <<!EOD`**

12

In our ever changing development environment at Robelle, we wanted a way of moving files around to our various servers via FTP.

Every time we build one of our products, we generate new WinHelp files on our NT server called beano.robelle.com. Although we had hardcoded the name of the machine (beano.robelle.com) in many of our scripts, we wanted to be able to change to a new NT server name in case the server we currently use became unavailable.

By setting the name of our NT development machine to NTDEV via the use of the global files, we can reference that machine by its logical name instead of its real name.

Now, we only need to change the real name of the NT server in the global files so that our scripts will continue to behave as they did before the change.

For Techies

References

Using system-wide variables

```
#!/bin/sh
#
#  Get the latest Suprtool WinHelp files from NTDEV
#
      ftp -n -v $NTDEV <<!EOD
      user username password
      binary
      cd /d/windev/winhelp/zip
      get suprux.exe /usr/robelle/winhelp/suprhelp.exe
      quit
      !EOD
```

13

The above Hereis document (more on that topic soon) shows how we use a logically named environment variable as opposed to the actual name of our NT development server, which will change names over time.

For Techies

References

HP-UX file system

- Tree structure consisting of directories and files
- MPE: filename.group.account
- HP-UX: /ACCOUNT/GROUP/filename
- /ACCOUNT/GROUP/mydir/mysubdir/myfile
- Mapping HP-UX to MPE
- HFS (Hierarchical File System)

14

When I first started with MPE, I took the introductory MPE course. We were given the analogy that the file system is like a file cabinet. We can apply this analogy to HP-UX, in which the filing cabinet can be the entire file system, a drawer is a directory, a folder within the drawer is a subdirectory, and a report within a folder can be thought of as a file.

The difference on HP-UX is that you can have folders within folders.

On MPE we refer to a file by the filename, the group the file resides in, and then finally the account:

filename.group.account

On HP-UX or in the HFS on MPE, the file would map to

/ACCOUNT/GROUP/filename

HP-UX and the HFS on MPE allow subdirectories within directories, giving multiple levels to the file system.

For Techies

References

File system and filenames

- Absolute path name
  - /users/me/directory/file
- Relative path name
  - directory/file
- ./ means current directory
- ../ means one directory above current directory

15

On HP-UX and subdirectories (within the HFS space), you can refer to a file in one of two ways: the absolute name of the file or the "full" name of the file. Absolute filenames start with "/", or from the root directory.

The relative filename is just the name of the file "relative" to where you are in the current directory structure. For example, if the full name of a script file is

`/users/me/cmds/scriptfile`

and you are currently in the /users/me directory, the relative filename of scriptfile is cmds/scriptfile.

There are several short forms which you can use to reference a file or program. For example,

`./suprtool` {run Suprtool program in the current directory}

`../suprtool` {run Suprtool program one directory above current}

You can find out your current directory by issuing the following command:

`pwd`

For Techies

References

File access and permissions

- File access and permissions are divided into these three categories:
  - Owner, Group, Other
- Three types of access:
  - Read, Write, eXecute

Owner	Group	Other
rwx	rwx	rwx

- **ls -l output**

```
-rw-rw-r--  1 root   sys    0 Jan 31 14:50 robeject
-rwxr-xr-x  1 root   sys  696 Jan 31 14:50 tape.license
-rwxr-xr-x  1 root   sys  692 Jan 31 14:50 tape.pre
```

16

The UNIX file system divides users into these three categories as it relates to a file: the user who creates or owns the file, the group that may have access to it, and everyone else.

Within each of these three categories, users can perform three different operations on a file: Read, Write, and eXecute.

You can see the permissions of a particular file by using the -l option of the ls command (sometimes aliased to ll). The description of the access and permissions is always in Owner, Group, and Account order, and Read, Write, and eXecute within each classification.

For Techies

References

Changing permissions

- Change the permissions of a file with the chmod command
- **`chmod +x jtest`** {give execute access to all}
- **`chmod u+x g-x o-x jtest`**
  {add x for owner, remove from group/other}
- Numeric representation of Read, Write, and eXecute:
  - Read = 4
  - Write = 2
  - eXecute = 1
- **`chmod 640 jtest`** {numeric equivalent 640 is rw-r-----}

17

You can change the permissions of a given file, if allowed, via the chmod command. The syntax is

chmod permissions filename

Permissions can be represented in any number of ways by a slightly confusing numeric representation.

Given that Read is worth 4, Write is worth 2, and eXecute is worth 1, we know that full access to a file is 7 for each category of Owner, Group, and Other.

Therefore, complete access to a file for all categories is 777. If we want to change the permissions of a file in the current directory, we would issue the following chmod command:

```
chmod 777 filea
```

For Techies

References

Simple commands

- Listf = ls -l

```
$ll proc*
-rw-rw-r--   1  neil  users  968  Feb 1 14:46  proctime.c
-rw-rw-r--   1  neil  users  740  Feb 1 14:46  proctime.o
```

- Fcopy, MPEX = cp

```
cp source.file destination.file
cp /dev/src/*.c  *.c
cp -R /users/me .
```

18

Listf = ls

On MPE we commonly use the Listf or Listfile command to look at the attributes of MPE files. On HP-UX we can look at the various attributes of a file with the ls command.

By default, the ls command does not list all of the files. To list the files that begin with a period (.), you must use the -a option.

Fcopy, MPEX, and cp
On MPE we could copy files using Fcopy, Copy or the MPEX Copy command. The MPEX Copy command allows the user to specify filesets when selecting file(s). The cp command in HP-UX also allows this by default.

More on filesets and wildcards (regular expressions) in a few minutes.

For Techies

References

Simple commands continued

- Purge = rm
  ```
  rm file1
  rm file*
  ```
- Rename = mv
  ```
  mv file1 file2
  ```
- Overwriting files on UNIX does exactly what you asked
  ```
  alias rm='rm -i'   # make the cp default to interactive
  ```
- File command, not the same as MPE
  ```
  file proctime.*
  proctime.c:  {C program text}
  proctime.o   {s800 relocatable object}
  ```

19

For Techies

Purge and rm
Beginning with MPE/iX 5.0, we can purge files on MPE with wildcard characters. On HP-UX the command for purging files is rm, which stands for remove files.

WARNING: Never use rm -r * because this command will remove all the files from the file system, starting from the root directory.

Rename and mv
On HP-UX the equivalent to the MPE Rename command is mv, which stands for move.

Overwriting Files
One problem with the cp, mv, and rm commands is that, by default, they do not prompt the user when they are going to overwrite an existing file. To prompt the user before overwriting an existing file, you must use the -i option with the commands. I make the -i option the default for some commands by putting an alias command in my .profile file.

References

File Command
The file command on HP-UX is not the same as on MPE. On MPE the file command is used to define the attributes of a file when it is built, or on what device it is built.

On HP-UX the file command attempts to determine the file type. There are no file labels or file codes to indicate the file type on HP-UX. The file command prints out the type of file you have.

Searching through files

- Print = more
- More has “more” functionality
  - **`more file1`** {display the contents of file1}
  - **`more files*`** {display all files beginning with the word "files"}
  - **`more +/rm /etc/profile`** {display all the lines with the letters "rm"}
- Magnet versus grep
  - **`grep searchstring filelist`**
  - **`grep -i -w blkpt *`**

20

More

On MPE we can list files with the Print command. On HP-UX we have the more command, which can display file(s) on a page by page basis. It also allows for intelligent searching of a list of files for strings matching a particular pattern.

Searching Through Files

One of the most common operations that we perform on a daily basis as part of software development is searching through a set of files. I use the Magnet Tool on MPE to search through filesets. On HP-UX we have grep.

Grep is a standard program found on most UNIX systems. It accepts some command options. For example, a searchstring expression followed by a fileset expression.

By default, grep is case sensitive (use -i to ignore case), and it ignores the context of a string (use -w to match only words).

The expressions used for the searchstring are known as Regular Expressions.

For Techies

Magnet is a product of Lund Performance Solutions.

References

Regular expressions

- Regular expressions are a symbolic method of describing a pattern
- MPE has fileset definitions
  - **`listf  npa@.group.account,2`**
- HP-UX regular expression summary:

^ (caret) Match expression at the start of a line, as in ^A
$ (dollar sign) Match expression at the end of a line, as in A$
\ (back slash) Turn off the special meaning of the next character, as in \^
[ ] (brackets) Match any one of the enclosed characters, as in [aeiou]
Use a hyphen (-) for a range, as in [0-9]
[^] Match any one character except those enclosed in [ ], as in [^0-9]
. (period) Match a single character of any value, except end of line
* (asterisk) Match zero or more of the preceding character or expression
\{x,y\} Match x to y occurrences of the preceding characters or expression
\{x\} Match exactly x occurrences of the preceding characters or expression
\{x,\} Match x or more occurrences of the preceding characters or expression

21

Regular expressions are a symbolic method of describing a pattern to match text within a line. For me, this is the most difficult function to remember in my everyday life. Once you understand regular expressions however, you won't forget.

When you type an incorrect regular expression, you get the infamous error message, "Regular Expression botch".

The backslash character is also known as the "escape" character.

For Techies

References

Examples of regular expressions

- **`grep smug files`** {search for lines with “smug”}
- **`grep '^smug' files`** {“smug” at the start of a line}
- **`grep 'smug$' files`** {“smug” at the end of a line}
- **`grep '^smug$' files`** {lines containing only smug}
- **`grep '\^s' files`** {lines starting with “^s”, “\” escapes the ^}
- **`grep '[Ss]mug' files`** {search for “Smug” or “smug”}
- **`grep '[B[oO][bB]' files`**
  {search for “BOB”,”Bob”,”BOb”, or “BoB”}
- **`grep '^$' files`** {search for blank lines]
- **`grep '[0-9][0-9] ' file`**
  {search for pairs of digits}

22

For Techies

References

Input/Output redirection

- Stdin = 0
  - **`prog < filename`**
- Stdout and Stdlist = 1
  - **`prog > filename`**
  - **`prog>>filename`**
- Stderr = 2
- **`prog >> file 2>&1`**
- MPE/iX 5.5 treats Stdin, Stdout/Stdlist, and Stderr in a much more UNIX-like manner

23

For Techies

I/O redirection allows input from a file to be passed into a program and conversely, output from a program to be redirected to a file. This concept is very similar to what we have had on MPE, with a few exceptions.

On MPE redirected output, by default, goes to a temporary file, but because there are no “real” temporary files on HP-UX, this is not a major issue.

On HP-UX there is a third standard file, known as stderr, to which all error messages are written. You can redirect both stdout and stderr with the following command:

```
prog > file 2 >&1
```

Starting with MPE/iX 5.5, we will have the stderr file, and will handle the stdin and stdout files exactly the same as on HP-UX.

References

Pipes

- Transfer output from one program to the input of another
- Use the “|” symbol between the two commands
- **`ps -ef | grep suprtool`**
  {find out all users who are running Suprtool}
- **`man who | lp -o2`**
  {print out the man page for the who command}

24

Pipes are simply a method of taking the output of one program and feeding it in as input to another program.

For Techies

References

Hereis documents

- hereis = Info String
- Used to specify input to a program
- hereis symbol is "<<"

```
#!/bin/sh
#  Get the latest Suprtool WinHelp files from NTDEV
      ftp -n -v $NTDEV <<!EOD
      user username password
      binary
      cd /d/windev/winhelp/zip
      get suprux.exe
      /usr/robelle/winhelp/suprhelp.exe
      quit
      !EOD
```

25

Hereis documents can be used to specify input to a program within a shell script without having to deal with filenames and cleaning up files when using I/O redirection.

A Hereis document allows input data as in-line text after the program name, which eliminates the need for a separate file.

For Techies

References

Shell programming

- Command substitution
- Expression evaluation
- Testing conditions
- While loops
- For loops

26

Shell programs are exactly the same as command files on MPE. They are regular ASCII files that contain a series of commands, and have at least read and execute access. You can comment lines in shell scripts by preceding the line with a “#”.

You can reference environment variables, pass in command line arguments, and accept user input.

You can execute a shell script by typing the filename of the shell script.

For Techies

References

Command substitution

- Command substitution syntax:
  `'command'`
- Executes command and places the output within the quotes
  `tar cv 'cat filelist'` {analogous to MPE's store ^filelist}
- Limited to maximum length of the command line
- Can be used to simulate exporting variables from a program (JCWs)

27

In MPE we call setjcw() to set an environment variable (JCW) from a program. While a process cannot change its parent's environment in HP-UX, it can create files. For example, to simulate passing a JCW called "LinesChanged" with a value of 5 back to the shell, simply have the process create a file called "LinesChanged" that contains the line:

`echo '5'`

The shell can access this "variable" by using the following syntax:

'LinesChanged'

For Techies

References

Expression evaluator

- Syntax:
  - expr expression
- Evaluate the expression and print the result on stdout
- Shell special characters (e.g., *) must be in double quotes
- Basic math and string operations
  - **`expr length "Qedit"`** # result is 5
  - **`expr substr "Suprtool" 5 3`** # substring is "too"
  - **`expr index "Suprtool.docnew" "."`** # result is 9
  - **`count='expr $count + 1'`** # add 1 to count

28

Typically an expression is used with an if statement and *while* or *for* loops. The expression may also include arithmetic operations. (Standard order of operation rules apply.) The expression is usually the part that is inside an if statement.

For Techies

References

Testing conditions

- [condition] {alternative; note the space around "[" and "]"}
- If condition returns error code 0, then the test has succeeded

[-f file] {file exists and is a regular file}
[-d file] {file is directory}
[-s file] {file exists and size > 0}
[str = str] {note the space around "="}
[str != str] {not equal to}
[num -eq -ne -gt -lt -ge -lt num]
[! condition] {not equal}
[cond -a cond] {and}
[cond -o cond] {or}

29

The if command/expression evaluation allows for branching in scripts. One tricky part is that the expression needs to have a space after and before each square bracket.

For Techies

References

Examples of testing conditions

```
if [ -r /usr/tmp/tapebeingmade ]
then
  echo “Tape being made currently. Please wait.”
  sleep 10
else
  echo “No one else is making a tape.”
  touch /usr/tmp/tapebeingmade
  x=1
fi
```

30

In English the above test means the following:

if the file /usr/tmp/tapebeingmade exists, then display a message and sleep for 10 seconds else display another message, create the file /usr/tmp/tapebeingmade and set x to one.

if…fi

case…esac

else if is be shortened to elif.

For example

```
if [ $susan_type -eq 2 ]
then
   echo “  $susan_number “ >> $temp_hpslist
elif [ $susan_type -eq 4 ]
then
   echo “ $susan_number”  >>$temp_hpslist
fi
```

For Techies

References

While loops

- While loops are the most commonly used looping construct in shell scripts
- The following example creates file0 to file99:

```
while [ $count -le 99 ]
do
    #create an empty file; can also use touch
    > file$count              #create the file
    count='expr $count + 1'   #increment count
done
```

31

While loops can be used for standard repetitive loops. The loop begins with a do statement and ends with a done. Other looping constructs are

until…do…done *or* for…do…done

Another example:

```
x=0
while [ $x != 1 ]
do
  mt -t /dev/rmt/0m rew
  if [ $? -ne 0 ]
  then
    echo “No tape was found please insert.”
    sleep 10
  else
    echo “A tape has been found.”
    x=1
  fi
done
```

For Techies

References

For loops

- A For loop iterates over a given set of values
- The following example creates qedit.list, suprtool.list, etc.
- Each of the .list files contains the contents of the corresponding directory

```
for var in Qedit Suprtool Xpress Howmessy.*
do
   ls -R /users/robdev/$var > ${var}List
done
```

32

For Techies

References

Job streams and cron

- MPE has job streams
- HP-UX has cron
- Cron executes commands at specified times
- Cron never exits and should be run through /etc/rc.
- Files are only checked during initialization or if a file changes
- Sample crontab file entry (/usr/sys/backup/backup.crontab)
  - 55 23 * * 1-5 /usr/sys/backup/backup -archive
- Run backup script at 23:55 every Monday thru Friday

33

For Techies

References

Starting crontab

- Must be Root
- crontab <filename>
- Script files
  - crontab.start

34

```
# !/bin/sh
# This script creates a new master crontab file by combining all of the
# various crontab files for root together and sending them off.
# See crontab.stop for a script to empty the crontab queue.
#
#  Create temporary files
temp_crontab=`mktemp - d /usr/tmp`
touch $temp_crontab
# Create master crontab list
cat /usr/sys/backup/backup.crontab  > $ temp_crontab
cat /users/scripts/pingrept.crontab     >> $temp_crontab
cat /users/scripts/weekly.crontab     >> $temp_crontab
# Submit the master list to crontab
crontab $temp_crontab
rm $temp_crontab
```

For Techies

References

Stopping crontab

- Submit an empty list
  - crontab /dev/null
- Crontab.stop

35

The crontab queue is emptied by submitting an empty file. /dev/null works just fine. Put it into a command file for consistency, called crontab.stop. For this to work, you must be logged on as root.

For Techies

References

Summary

- Help from HP-UX
- Shells in HP-UX
- Logging on
- Variables
- File system
- Simple commands
- Searching files
- Regular expressions
- I/O redirection
- Pipes
- Hereis documents
- Shell programming
- Cron/job streams
- Command summary

36

In summary, I hope this tutorial will help you as you integrate HP-UX into your HP 3000 environment.

For Techies

References