31

AIX update

May 1998

In this issue

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 550955
From USA: 01144 1635 33823
E-mail: HarryLewis@compuserve.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Contributions

If you have anything original to say about AIX, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you actively be helping the free exchange of information, which benefits all AIX users, but you will also gain professional recognition for your expertise and that of your colleagues, as well as being paid a publication fee – Xephon pays at the rate of £170 ($250) per 1000 words for original material published in AIX Update.

To find out more about contributing an article, see *Notes for contributors* on Xephon's Web site, where you can download *Notes for contributors* in either text form or as an Adobe Acrobat file.

Editor

Harold Lewis

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £15.00 ($22.50) each including postage.

AIX Update on-line

Code from *AIX Update* is available from Xephon's Web page at www.xephon.com (you'll need the user-id shown on your address label to access it).

Decreasing filesystem size

This article provides some hints and tips on methods of reducing the size of your filesystems. For the sake of this discussion, I've divided filesystems into two groups, as each requires a somewhat different treatment.

THE '/USR', '/VAR', '/TMP', AND '/' FILESYSTEMS

I've read many articles, including *Understanding performance inhibitors* in Issue 27 of *AIX Update* (page 28), that state that the size of a filesystem can't be decreased once it's set. I'm not sure whether the authors concerned mean that it's difficult to change the size of a filesystem once it's set (which it is) or that, in their view, it's actually impossible, but I've uncovered methods for doing just this that I'd like to share with other users.

The size of a filesystem in the *rootvg* volume group may be decreased by first changing */image.data* (in AIX Version 4) and then reinstalling the file system. The procedure for doing this is detailed below (it was tested on an RS/6000 43P Type 7248-133).

1 Start the system as normal. Login as *root*.

2 Mount all filesystems using **mount all**.

3 Ensure that */tmp* has at least 8 MB of free space (this is necessary for the step below).

4 Use the **mkszfile** command to create the file */image.data*.

5 In */image.data*, find the *lv_data* section that relates to the filesystem that you want to resize.

For example, if you want to reduce the size of the */var* filesystem, the relevant section of */image.data* is:

```
lv_data
        VOLUME_GROUP= rootvg
        LV_SOURCE_DISK_LIST= hdisk0
        LV_IDENTIFIER= 00751302d046b128.7
        LOGICAL_VOLUME= hd9var
```

```
VG_STAT= active/complete
TYPE= jfs
MAX_LPS= 128
COPIES= 1
LPs= 8
STALE_PPs= 0
INTER_POLICY= minimum
INTRA_POLICY= center
MOUNT_POINT= /var
MIRROR_WRITE_CONSISTENCY= on
LV_SEPARATE_PV= yes
PERMISSION= read/write
LV_STATE= opened/syncd
WRITE_VERIFY= off
PP_SIZE= 4
SCHED_POLICY= parallel
PP= 8
BB_POLICY= relocatable
RELOCATABLE= yes
UPPER_BOUND= 32
LABEL= /var
MAPFILE=
LV_MIN_LPS= 4
```

Now you can change the statement *LPs= xx* to the number of logical partitions that you want, where *xx* is the filesystem size you require (in megabytes) divided by *PP_SIZE*. This number must be greater than or equal to *LV_MIN_LPS*. In the above example we can change the line that sets *LPs* to *LPs= 4*. Note that this is the same as *LV_MIN_LPS*, meaning that the */var* filesystem is being set to its minimum size (this constraint is a result of the data stored in this filesystem).

Note that a physical partition (PP) is the actual unit of storage to which a logical partition (LP) maps. The size of a PP is defined when the volume group (VG) is created and cannot be changed later. The standard size of a PP is 4 MB for disks larger than 300 MB. The range of sizes available starts at 1 MB, doubling at each step (2, 4, 8, …) up to 256 MB.

6 In */image.data* find the section headed *fs_data* and change the line *FS_SIZE= yy* to the required filesystem size, where *yy* is *LPs* times *PP_SIZE* times 2048. The reason you multiply by 2048 is that filesystem size is by default in 512 KB blocks. *FS_SIZE* must be greater than *FS_MIN_SIZE*.

Let’s continue our example:

```
fs_data
        FS_NAME= /var
        FS_SIZE= 65536
        FS_MIN_SIZE= 25040
        FS_LV= /dev/hd9var
        FS_FS= 512
        FS_NBPI= 4096
        FS_COMPRESS= no
```

As we previously changed *LPs* to 4, so *FS_SIZE* must be changed to 32,768 (since 4 x 4 x 2048). Notice that *FS_SIZE* is greater than *FS_MIN_SIZE* (25,040), as required.

6 Use the command below to ascertain whether any user-defined paging spaces exist.

```
lsvg -l rootvg | grep paging
```

The output of this command is shown below. In this instance, it shows that a user-defined paging space has been created on a paging space other than *hd6* (it’s in *paging00*, a user-defined paging space).

```
            Physical  Volume
Page Space  Volume    Group       Size  %Used  Active Auto  Type
paging00    hdisk1    rootvg     288MB    13     yes    yes   lv
hd6         hdisk0    rootvg     260MB    16     yes    yes   lv
hd6         hdisk1    rootvg      28MB    16     yes    yes   lv
```

If any such paging spaces exist, deactivate them using the command:

```
chps -a n <paging_space>
```

<paging space> is the name of the paging space listed in the first column of the output.

7 Reboot the system. This is necessary to remove any user-defined paging spaces deactivated using the **chps** command.

8 Use **mount all** to mount all filesystems.

9 Make a full back-up using the command:

```
mksysb /dev/rmt0
```

The importance of using **mksysb** when backing up the system is that it uses the new version of */image.data* to restore the system.

10 Reboot the system again and activate all deactivated paging spaces using the command:

```
chps -a y <paging_space>
```

Reboot the system again, this time in order to re-activate user defined paging spaces.

11 Use the **mount all** command to mount all filesystems.

OTHER FILESYSTEMS

A much simpler procedure is required to reduce the size of filesystems other than those dealt with in the above section:

1 Back up the data to tape.

2 Unmount the filesystem.

3 Remove the filesystem.

4 Recreate the filesystem using the new size.

5 Mount the filesystem.

6 Restore the necessary data from the backup tape.

Despite warnings issued by IBM about possible damage to data, I see no reason why this procedure should result in any kind of damage if each step is applied carefully. Of course, taking a full system back-up before changing system parameters and variables is good practice, regardless of what changes you're making to your system, to ensure full recoverability.

Serdar Sahin
System Engineer (Turkey)

Understanding the grep command

This is the second and concluding part of this article, the first part of which was published in Issue 30 of *AIX Update* (April 1998).

Listing only the names of files containing matching lines (-l)

If you just want the names of files that contain lines that match the search string, use the **-l** flag. This displays a list of names of files (one per line) that contain matches. One benefit of using this method is that it returns only one entry per file, no matter how many matches the file contains.

This option is very useful when handling the large number of files that production applications and development projects often generate. For instance, suppose you have 300 project files, thirty of which contain strings that match the search pattern. Furthermore, suppose that each of those thirty files contains about ten lines matching the pattern. Using the **-l** flag causes **grep** to display just thirty file names; without the flag, **grep** would display every one of the hundreds of matches.

Displaying a count of matched lines (-c)

If you need a count of the matches in each file, use the **-c** flag. This produces a similar result to piping the output of a **grep** command to the **wc -l** filter, which counts and displays the number of lines that it receives as input.

Although the **-l** and **-c** flags both produce a list of file names, the **-c** flag displays all file names that match the file specification, whether or not they also contain lines that match the search pattern. Naturally, where a filename matches the file specification but no lines in the file match the search pattern, the file name is displayed followed by a zero.

The example below shows this version of the command in use along with a sample output.

```
grep -c xant /home/*/.login

/home/jones/.login:0

/home/ksmith/.login:2
```

```
/home/nsmith/.login:0

/home/paula/.login:1

/home/tbaker/.login:0

/home/walker/.login:0
```

The **grep** command above displays a list of all files that match the filename specifier along with a count of the number of lines in the file that match the string ‘xant’ in the login profiles of users in the */home* file system. Note that most of the entries contain no matches of the string ‘xant’.

To avoid the list being clogged-up by entries listing files with no matches, use another **grep** command to filter out lines that contain the string ‘:0’.

```
grep -c xant /home/*/.login | grep -v :0
```

This **grep** command does the following:

1 Searches all files matching the file specification */home/*/.login* for the search string ‘xant’.

2 Pipes the results to the second **grep** command.

3 The second **grep** command displays file names that don’t contain the string ‘:0’.

The result is:

```
/home/ksmith/.login:2

/home/paula/.login:1
```

Displaying the line number of the matched lines (-n)

Use the **-n** flag to display the line number of strings that match the search pattern. This is useful when you’re using a tool that takes line numbers as part of the input for some process, and is also useful when you need to identify lines by number. For instance, the command:

```
grep -n 'insert text here' master.doc
```

shows the line number in file *master.doc* of lines that contain the string ‘insert text here’.

A sample output of this command is:

```
124:insert text here

130:insert text here

215:insert text here

826:insert text here
```

Displaying the entire paragraph containing matched lines (-p)

Sometimes viewing just the lines that contain the search string may not provide you with all the information you need. Sometimes it's necessary to put lines in their proper context. In such instances it's helpful to have the whole paragraph at your disposal.

The **-p** flag does just this, displaying the entire paragraph that contains the matched text. You may specify a paragraph separator when using the **-p** flag by appending the separator directly to the flag. For example:

```
grep -pPARA 'specific text' myprog.doc
```

This displays paragraphs that contain the string 'specific text' in the file *myprog.doc*, using the string 'PARA' as a paragraph separator.

Note that the default separator is a blank line. You also need to be aware that, if you string a number of **grep** flags together, the **-p** flag must be the last one if you want to use the default separator. This makes sense if you think about it – for instance, if you specified flags **-ipv**, **grep** treats the 'v' as the paragraph separator rather than as the 'invert' flag. The correct order in which to list the flags in this example is **-ivp**. While the paragraph separator you specify is not displayed in the output, **grep** inserts blank lines between entries to make the output more readable.

Specifying search patterns in a file (-f)

Under certain circumstances you may wish to store the search pattern in a file and then call the file using the **grep** command. This is useful if you have a complex search pattern or one that you use frequently. The pattern file may also contain more than one pattern, with each pattern on a separate line.

For example, suppose you have a text document that you need to search for all occurrences of the strings ‘administrate’, ‘administrator’, and ‘administration’, but excluding the string ‘administrative’.

One way of doing this is to create a file (called *mypat1.file* in this example) containing the lines:

```
administrate

administrator

administration
```

Then use the following **grep** command:

```
grep -f mypat1.file admin.doc
```

This displays all lines in *admin.doc* that contain one or more occurrences of any of the patterns found in the pattern file. As you’ve not included the string ‘administrative’ in the search pattern, lines containing this string are not displayed unless the string appears on the same line as one of the ones you’re searching for, in which case it’s listed along with the others.

Lines containing only the exact search pattern (-x)

The **-x** flag tells **grep** to display only lines that match the search string exactly. Naturally, this form of the **grep** command requires you to format the query in such a way that all files in which you’re interested are found, even when such factors as leading spaces are taken into account. Using the **-x** flag is equivalent to specifying ‘^text$’ as the search pattern (where the string being searched for is ‘text’). In other words it tells **grep** to display only the lines that begin with the first character of the string ‘text’ and end with the last character.

When using **grep -x** to search a file, it’s often useful to use the **-c** and **-n** flags as well, otherwise the output just shows each occurrence of the search pattern. The command **grep -xn** shows the number of all lines that match the search pattern, and **grep -xc** show how many lines in the selected file match the search pattern exactly.

The following example combines the **-v** and **-x** flags to filter out blank lines:

```
grep -vx "" file.spec
```

This example tells **grep** to display all lines in *file.spec* except (**-v**) those lines that contain the null string (**-x** `""`) and nothing else. Redirecting the output of this command to a file rewrites the input file removing blank lines.

Suppress the name of files (-h)

Sometimes you may want to display lines that match a search string without prefacing them with the file name, even when you're working with multiple files. For instance, you may need to suppress file names when piping the output into a sort or other filter or tool. The flag to do this is **-h**.

For example:

```
grep -h text myproj*.c
```

Displays all occurrences of the string 'text' in files matching the descriptor *myproj*.c* without preceding each one with the name of the file.

Suppress writing to standard output (-q)

The **-q** flag prevents output being written to standard output regardless of whether matches are found. The question is, what use is this?

Suppose you have a program that issues **grep** commands during its processing. Using the **-q** flag allows the program to use **grep** commands without generating voluminous output – after all, the program can use **grep**'s return code to determine the necessary course of action. (If **grep** finds a match, it returns a '0', and if no matches are found it returns a '1'. If syntax errors are found or a file is inaccessible, **grep** returns a number greater than 1.)

Using the **-q** flag has the same effect as redirecting **grep**'s output to */dev/null*.

The table below summarizes the flags used with the **grep** command.

-c	Displays only a count of the number of lines that match the search pattern.
-f	Specifies that search patterns are located in a file.

-h	Suppresses file names in **grep**’s output.
-I	Ignores the case of the search pattern.
-l	Lists only names of files, omitting the string that matches the search pattern.
-n	Displays line numbers of matching lines.
-p	Displays the entire paragraph that contains a matching line.
-q	Suppresses writing to standard output.
-v	Displays lines that don’t match the search pattern.
-x	Displays only lines that match the search pattern exactly.

SOME EXERCISES

Below are three exercises to test your knowledge of the **grep** command: two simple ones and one that’s quite complex.

Exercise 1

Question: How would you search for the string ‘-n’ in the file *test.file*?

At first sight you might be tempted to use the command:

```
grep -n test.file
```

However, if you did, **grep** would return a blank cursor. The reason is that **grep** doesn’t interpret the ‘-n’ as a search pattern but as the **-n** flag (display line numbers) and ‘test.file’ as the search pattern. As this leaves no parameters to specify the file to search, **grep** looks to standard input for data to search for the string ‘test.file’. This means you need to press CONTROL+C to restore the cursor.

So, which command would you use to specify the desired string?

Actually there are a few ways to specify a search pattern that begins with a minus sign (-). In previous sections we’ve seen that you could escape the minus sign using either of the following commands:

```
grep '\-n' test.file

grep \\-n test.file
```

The minus is not considered a metacharacter, but still requires special treatment when it's the first character of a search pattern.

You could also specify a search pattern beginning with a minus sign by storing it in a file and using the **-f** flag. For instance, create a file called *mypat1.file* that contains just the string '-n' and then use the command:

```
grep -f mypat1.file test.file
```

Exercise 2

Question: Which command can you use to copy files that contain lines that match a specific search pattern to another directory?

Let's say you have 50 test files in directory */home/tsmith/test1* that match the file specification *myproj*.c*. If you want to determine which ones contain the string 'TMP_DIR' and copy them to */home/tsmith/myproj*, the best way is as follows:

First, it should be apparent that a useful **grep** flag for this task is **-l**, which lists the names of files that contain lines that match the search string no matter how many occurrences of 'TMP_DIR' they contain.

Next, you may know that, when a command line includes a command enclosed in back quotes (`), the enclosed command is executed first and its result is used to replace the quoted string itself, then command line is interpreted by the shell.

If you embed a **grep** command using back quotes in a copy command, you should therefore get the desired results. Consider the following:

```
cp `grep -l pattern filespec` path
```

This command is interpreted by the shell as follows.

1 The shell executes the embedded **grep** command first, storing the output of the command, which is a list of files that match *filespec* and contain one or more lines that include the search pattern *pattern*.

2 The **grep** command in quotes is then replaced by a string of file names, thus generating a command that copies the named files to *path*.

Therefore, the command to accomplish the task set out above is:

```
cp `grep -l TMP_DIR /home/tsmith/test1/myproj*.c` /home/tsmith/
myproj
```

This exercise illustrates how, with practice, you can use **grep** to perform multi-step processes in one operation.

Exercise 3

Question: A file called *myprog.doc* contains numerous references to 'dynamic schedule', 'dynamic scheduler', and 'dynamic scheduling', in which each of the phrases may appear in a mixture of upper and lowercase. You need to find strings that begin with the word 'dynamic' but in contexts other than those mentioned above. You also know that the structure of the document is such that the text you're looking for does not appear on the same line as any of the above phrases.

You have a further requirement – you want to see the entire content of paragraphs that include the search text in order to find situations where the above phrases may be split over two lines. The document uses the following paragraph separators: <*p*> for the 'begin paragraph' and <*/p*> for 'end paragraph'.

The following phrases are typical of the desired output:

```
a dynamic program

it is dynamically allocated

Dynamic resource reduction
```

How would you formulate a **grep** command to achieve this task?

As before, there may be more than one solution. However, let's concentrate on the one that uses the **-p** flag to show full paragraphs, which allows us to determine the context of search text.

We start by breaking the problem down into two separate operations: showing paragraphs that contain 'dynamic' and suppressing those that contain 'dynamic schedule', etc.

First we set up a **grep** command that uses the **-i** flag to disregard the case of the search string 'dynamic'. To this we add the **-p** flag along with the paragraph separators <*p*> and <*/p*>.

It's possible to specify both <*p*> and </*p*> in one pattern – consider the following sequence of characters:

```
'\</*p\>'
```

We escape the 'less than' (<) and 'greater than' (>) characters with the backslash (\) to ensure that the shell does not interpret them as redirection symbols. To specify the forward slash (/), which is part of the 'end paragraph' delimiter, we can use the pattern '/*', which means 'zero or more backslashes'. The 'p' completes the pattern. The above string thus specifies both a begin and end paragraph delimiter, regardless of whether it contains a backslash.

Now we can add the search string 'dynamic' and the file specification *myprog.doc*. Let's see what we have so far:

```
grep -ip'\</*p\>' dynamic myprog.doc
```

If successful, this **grep** command displays every paragraph that contains the word 'dynamic', ignoring case. Even though the **-p** flag doesn't display the paragraph separator, it inserts a blank line between paragraphs to aid readability. You can take advantage of this by piping the result of this command into another **grep** command, again using the **-p** flag.

This second **grep** command also needs the **-i** flag to ensure the search continues to disregard the case of the search pattern. This time, we'll use the **-v** flag to suppress paragraphs that contain the string 'dynamic schedul', which covers the three phrases specified in the problem. Using the **-p** flag without specifying a paragraph separator tells **grep** to use the default blank paragraph separator.

```
grep -ip'\</*p\>' dynamic myprog.doc | grep -ivp 'dynamic schedul'
```

The second **grep** doesn't act on paragraphs that contain the string 'dynamic', but filters out those that contain the phrases 'dynamic schedule', 'dynamic scheduler', or 'dynamic scheduling', and is thus a solution to the problem.

While this exercise may seemed a bit contrived, it's actually fairly typical of the sort of use to which **grep** is put. If you regularly use complex patterns with **grep**, you should consider storing them in files and using the **-f** flag.

SUMMARY

The **grep** command can be used in both simple and highly specialized circumstances. It is a powerful and commonly utilized command. With practice, it can be one of your most useful tools.

David Chakmakian (USA) © Xephon 1998

Using error logging in AIX

CONCURRENT ERROR NOTIFICATION

Sometimes it's useful to set up the **errpt** command so that it runs continuously. Then, if an error occurs, it is immediately notified either to the console or by e-mail to an operator. This is easily accomplished with the **errpt -c** command. The output of this command can then be directed to a file, which may be 'tailed' continuously, as in the following example:

```
# errpt -c > /tmp/errors.out &
# tail -f /tmp/errors.out
```

THE ERRLOGGER COMMAND

The **errlogger** command may be used to write an operator message to the error log. The format of this command is **errlogger *text_string***. In some situations, such as logging the output of a shell script, this is usually enough, as it provides a time-stamped error record together with some descriptive text. The actual text is written at the end of the error log record, as in the example that follows:

```
# errlogger Cleaned 8mm tape drive...again
# errpt -a | more
------------------------------------------------------------
LABEL:          OPMSG
IDENTIFIER:     AA8AB241
Date/Time:       Mon Mar 24 08:58:38
```

```
Sequence Number: 331783
Machine Id:      000004333500
Node Id:         tcmis
Class:           O
Type:            TEMP
Resource Name:   OPERATOR
Description
OPERATOR NOTIFICATION
User Causes
ERRLOGGER COMMAND        Recommended Actions
        REVIEW DETAILED DATA
Detail Data
MESSAGE FROM ERRLOGGER COMMAND
Cleaned 8mm tape drive...again
```

THE ERROR LOGGING SYSTEM VERSUS SYSLOGD

There is another system in AIX for recording errors – **syslog**. This error logging system is not used by many commands, as the preferred method of recording error messages in AIX is via the error log file. The file */etc/syslog.conf* is used to specify which messages are written to particular locations using **syslogd**. Each line in this file has the format ***subsystem.severity destination***. Just as you write to the error log using the **errlogger** command, so you write to the **syslog** using the **logger** command.

THE LOCATION AND SIZE OF THE ERROR LOGGING FILE

To determine the current path, size, and buffer size of the error log, use the command **errdemon -l**, as in the example below.

```
# /usr/lib/errdemon -l
Error
Memory Buffer Size      8192 bytes Log Attributes
---------------------------------------------
Log File                /var/adm/ras/errlog
Log Size                1048576 bytes
```

On a large system, with many devices and much system activity, it may be necessary to increase the size of the error log. This can be done using the command **/usr/lib/errdemon -s *size***. Note the output of the first command, which reduced the size of the log file.

```
# /usr/lib/errdemon -s 16384
Decreasing the error log file maximum size caused
```

```
     /var/adm/ras/errlog to be moved to /var/adm/ras/errlog.old.
# /usr/lib/errdemon -s 32768
#
```

BOOT ERRORS

During system boot the error subsystem is not initialized. To view errors that occur at this time use the command **alog -o -t boot | more**, which allows you to view the file */var/adm/ras/bootlog*. The last error recorded is placed in non-volatile RAM and is copied into */var/adm/ras/errlog* when the machine is booted.

ANALYZING THE ERROR LOG FOR HARDWARE ERRORS

The utility below analyzes hardware error entries – enable it as shown below. To switch it off, use the parameter **DISABLE**.

```
# /usr/lpp/diagnostics/bin/ ENABLE
```

HOW TO DISABLE LOGGING FOR A PARTICULAR EVENT

Sometimes it's useful to switch off error logging for a particular event. An example of this is a device that is known to be defective. Each time an error is detected for this device, it is logged, possibly clogging up the error log and obscuring other entries (this is especially true of network card errors). Use the **errupdate** command to switch off error logging for an event. This command may be invoked in command mode, though its syntax is counter-intuitive – the example below shows the command being used to disable the reporting of a specific problem associated with a Token Ring adapter. The error code is the one given by **errpt -t** for the event you want excluded.

```
# errupdate
=E0EA14BF:
Log = False  [Enter] [Control-D] [Control-D]
0 entries added.
0 entries deleted.
1 entries updated.
#
```

To find out if there are any events for which logging is disabled (this is important, as known problems are easily forgotten), use the command:

```
errpt-t -F Log=0
```

A sample output is shown below, showing that just one error event ID is flagged as disabled on this system.

```
# errpt -t -F Log=0
Id       Label             Type CL Description
E0EA14BF TOK_BEACON1       TEMP S  OPEN FAILURE
#
```

ERROR TEMPLATES

As discussed in *Error logging and reporting* in *AIX Update* Issue 15, AIX keeps ‘error templates’ that contain skeleton text for each error message. This text is not stored in the error log, but is merged with information from the error log when a report is requested. The **errpt -t** command shows all the error templates defined.

Error logging templates are stored in */var/adm/ras/errtmplt*. Note that this file cannot be edited using **vi**. A sample error log is shown below (note the third entry, which is used as an example later).

```
D27394BD CTOK_ADAP_OPEN      PERM H  OPEN FAILURE
DD0E4902 TOK_RCVRY_EXIT      TEMP H  PROBLEM RESOLVED
TEMP S  OPEN FAILURE
E535D331 CTOK_RCVRY_EXIT     TEMP H  PROBLEM RESOLVED
ECEDB274 CTOK_MEM_ERR        UNKN S  OUT OF MEMORY
```

For details of specific errors, use the command **errpt -ta**. The output of this command for the ‘beaconing’ error in the above report is shown below:

```
# errpt -ta -j "E0EA14BF"
-----------------------------------------IDENTIFIER E0EA14BF
Label: TOK_BEACON1
Class: S
Type:  TEMP
Loggable: YES   Reportable: YES   Alertable: NO
Description
OPEN FAILURE
Probable Causes
TOKEN-RING FAULT DOMAIN
Failure Causes
TOKEN-RING FAULT DOMAIN
Recommended Actions
        ATTEMPT TO REOPEN THE ADAPTER AFTER 30 SECONDS
        IF PROBLEM PERSISTS THEN DO THE FOLLOWING
        REVIEW LINK CONFIGURATION DETAIL DATA
        CONTACT TOKEN-RING ADMIN RESPONSIBLE FOR THIS LAN
Detail Data
SENSE DATA increased
ADDITIONAL SUBVECTORS
#
```

DEFINING NEW ERROR MESSAGES

Use the command **errmsg** to define new error messages. This command uses a code to determine the ‘message set’ to which the message belongs – the codes used are shown below.

```
D Detailed data
E Error description
F Failure cause
I Install cause
P Probable cause
R Recommended action
U User cause
```

First look at the messages that are already defined in the error description message set using the command **errmsg -w**, as there may already be one that can be used. There are thousands of error messages defined – a few (all genuine) are shown below.

```
# errmsg -w E
SET E
003 "CPC HARDWARE FAILURE"
1607 "TOPOLOGY PROTOCOL ERROR"
B107 "TOXIC LEAK DETECTED"
E8A6 "A FATAL ERROR HAS OCCURRED"
```

By going into **errmsg** at the command line, messages can be added and deleted, as shown below.

```
# errmsg
SET E
+ "Fairly toxic leak" [Control-D]
SET E
E000 "Fairly toxic leak"
#
```

Note that only user-defined errors can be modified or deleted – trying to modify one of the system’s intrinsic error messages results in the following:

```
# errmsg
```

```
SET E
- B107 [Control-D] [Control-D]
errmsg: "stdin", Line 2:
Skipping Message ID: - b107
For SET E you can only delete Message IDs
in the range [0xE000 - 0xE7FF]
1 total errors. No modifications made to
/var/adm/ras/codepoint.cat
#
```

USING THE ERROR LOG MESSAGES YOU HAVE CREATED

Applications log messages to the error log using the **errlog** subroutine from *librts.a*. This writes an entry to the file */dev/error*. The **errdemon** periodically reads this file and creates a time-stamped error log entry and VPD (if appropriate) to the current error log. The way the subroutine is called is documented in the *AIX 4.1 Problem solving guide and reference*. The header files to look at are */usr/include/sys/errids.h and /usr/include/sys/err_rec.h*

System Programmer (UK) © Xephon 1998

Using xmperf

INTRODUCTION

One of the most important questions that IS managers and administrators have to answer is whether their systems are performing adequately. Given that the answer to this question varies according to such factors as the time of day, processing load, processor capacity, and available memory, perhaps a more appropriate question is: does the system have enough capacity to run its workload or is it underpowered? To answer this question you need accurate performance measurements.

To this end AIX comes with a number of performance measurement tools, such as **iostat** and **vmstat** on standard systems, and **spmon** on SP systems. These tools are all command line-oriented, and are all somewhat limited in terms of the system events they are able to monitor.

The Performance Tool Box (PTX – see *The AIX Performance Toolbox, AIX Update* Issue 29) is another collection of tools that's available as an option from IBM. This comprises a number of utilities designed to monitor just about every aspect of an AIX system that might be of importance. One of these tools, **xmperf**, is a highly configurable program that allows you to measure a number of system events and display the results using a number of different views. As implied by its name, it's a graphic tool that runs under the X-Window system. This article guides you through the steps required to set up the tool and shows you what it's capable of.

INSTALLATION REQUIREMENTS

The Performance Tool Box for AIX 4 comprises the following three components:

- Performance Tool Box Network Feature (*perfmgr.network*). This is the component for monitoring remote systems.
- Performance Tool Box Local Feature (*perfmgr.local*), which is the component for monitoring local systems (it's included in the PTX Network Feature).
- Performance Aide (*perfagent*), which transmits data to either the Network or Local Feature components.

The Performance Aide comes in different versions for AIX 4.1 and 4.2. The Performance Tool Box is on the first CD-ROM, while Performance Aide is on the second, along with the program's documentation, which is mostly in HTML and can be viewed using any browser.

Prerequisites for installing PTX are:

- AIX Version 4

- *bos.net.tcp.client.4.1.0.0*
- *bos.sysmgmt.trace.4.1.0.0*
- *X11.base.rte.4.1.0.0*
- *X11.base.lib.4.1.0.0*
- *X11.motif.lib.4.1.0.0*.

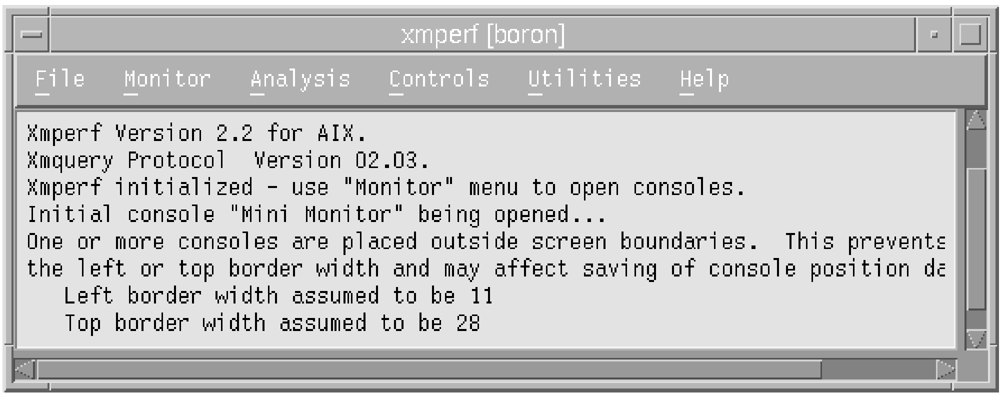
As a standard installation of AIX 4 includes the X-Window system, in the majority of cases you'll find that all the prerequisites are already installed. All PTX's Performance Aide components can be installed using **smit** (**smitty install_latest**). My installation ran without flaw and copied most of the data into */usr/lpp/perfagent*, */usr/lpp/perfmgr*, and */usr/lpp/perfmgr.network*. The documentation (in HTML format) is in */usr/share/man/info/en_US/perfagent*.

STARTING PTX

Once PTX and Performance Aide are installed, **xmperf** can be started from the command line using the command:

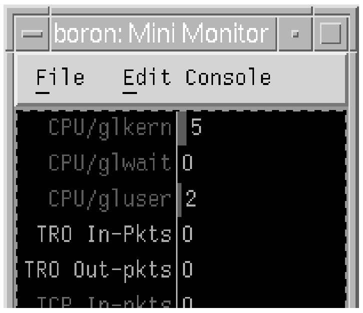
```
xmperf &
```

This is the simplest way of starting the utility, though the command to start **xmperf** can also include a number of options, such as the host name of the system that you wish to monitor. All configuration



```
Xmperf Version 2.2 for AIX.
Xmquery Protocol  Version 02.03.
Xmperf initialized - use "Monitor" menu to open consoles.
Initial console "Mini Monitor" being opened...
One or more consoles are placed outside screen boundaries.  This prevents
the left or top border width and may affect saving of console position da
   Left border width assumed to be 11
   Top border width assumed to be 28
```

*Figure 1: **xmperf**'s main window*



```
  CPU/glkern 5
  CPU/glwait 0
  CPU/gluser 2
 TRO In-Pkts 0
TRO Out-pkts 0
```

Figure 2: ***xmperf****'s Mini Monitor window*

information (roughly 160 KB of it) is kept in the file */usr/lpp/perfmgr/xmperf.cf*. You may also wish to set up the file *$HOME/xmperf.cf*, which may be created from the main window and records the defaults that are then used.

When you start **xmperf**, the utility may exit with a message stating that 'no hosts responded to the invitation'. This message may usually be ignored – just try starting **xmperf** again. Once **xmperf** starts, the program's default configuration is to display two windows: the main window (Figure 1) and a smaller window titled 'Mini Monitor' (Figure 2). This first console contains a few sample performance values from the local host that are displayed by default the first time you start **xmperf**.

It's useful to have a quick look at Mini Monitor, which gives a quick overview of **xmperf**'s facilities, showing some general performance parameters for the local system. All **xmperf**'s features are accessible from the main window. We'll take a quick tour through it and discuss some of the more interesting points.

THE MAIN WINDOW

The three items of most interest in the *File* menu are:

- *Save All Changes*, which allows you to store configuration changes made to consoles.
- *Playback*, which puts **xmperf** in 'playback mode', so that recorded

events can be viewed at various speeds.

- *Refresh Host List*, which allows you to scan the network for new hosts to be monitored (by default this is done every five minutes, though using this option forces a refresh).

The *Monitor* menu is where you create new consoles. In the lower part of this menu are options that allow you to select one of a number of pre-configured consoles, known as ‘skeleton consoles’ (Mini Monitor is an example of this). However, you may also create new consoles from scratch using the option *Add New Consoles*. If you choose to define your own console, you then have to add instruments to it.

The *Analysis* menu is the first of three customizable menus. As the name suggests, you’ll find commands here to execute performance analysis, including a number of pre-configured analysis tools, such as *Real Memory Analysis* and *IO Analysis*. Other analysis programs included in the Performance Tool Box, such as **bf** (BigFoot), may also be made available in this menu by being included in the configuration script (a file that resembles an X-Windows resource file). You may also include entries for shell scripts and utilities stored in binary files in this menu.

The next menu is *Controls*. One of the entries in this menu provides you with a list of local processes. As in the case of the *Analysis* menu, this menu is also fully configurable and may include user-created controls.

The *Utilities* menu includes an entry to create a list of processes on a remote host that may then be selected from a pull-down menu. Many pre-configured menu entries are also available here, including one to produce three-dimensional displays of local and remote processes.

Help displays help on using the current window. This information is also available from submenus of the Main window, showing context-sensitive help information from the file */usr/lpp/perfmgr/xmperf.hlp*.

THE MONITOR MENU

Despite the fact that skeleton consoles are at the bottom of this menu, I’ll discuss them first. The first thing you’ll notice is that active

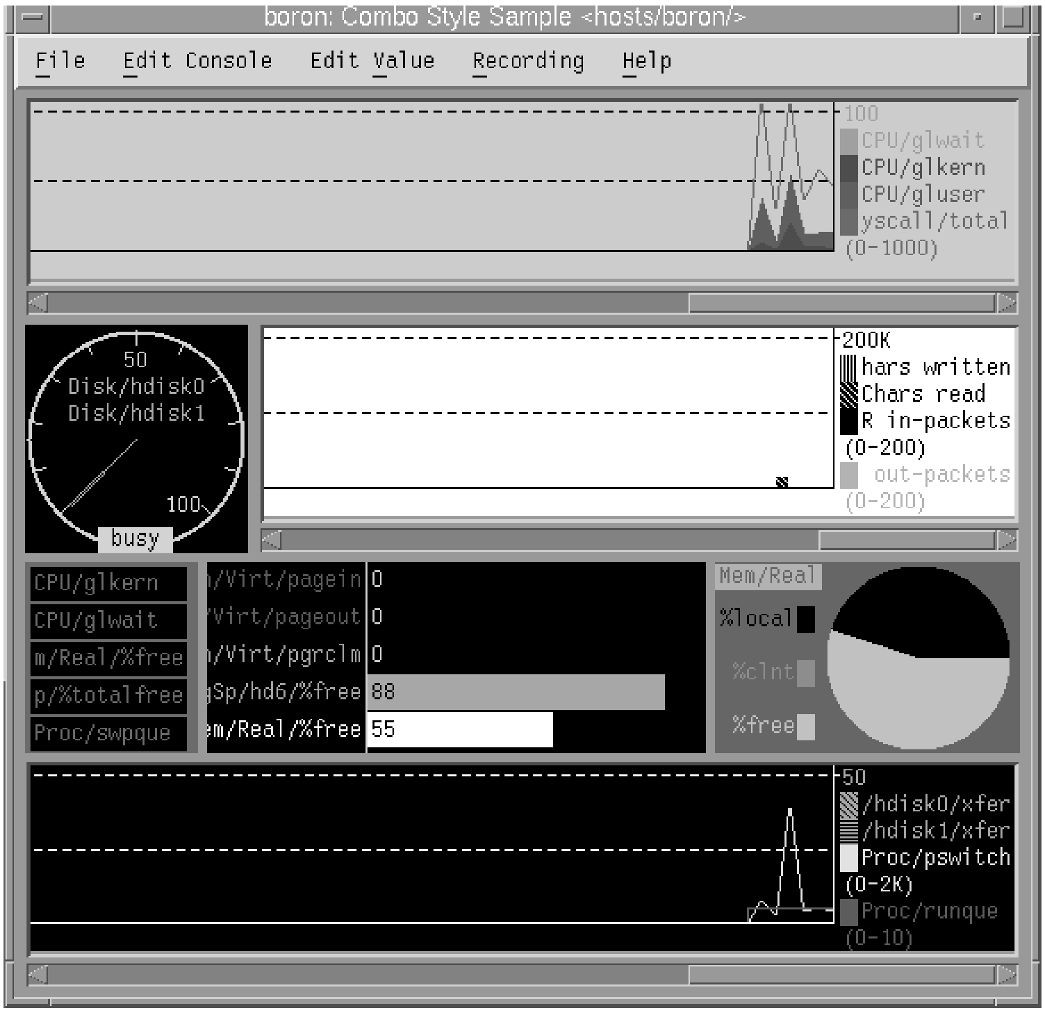


Figure 3: ‘Combo Style Sample’ console

consoles have an asterisk (*) in front of their name in this menu. As mentioned earlier, Mini Monitor (Figure 2) is one such console.

Another interesting skeleton console is ‘Combo Style Sample’, shown in Figure 3. This console is good for demonstrating **xmperf**’s display capabilities. The first panel shows CPU statistics (this type of display is known as ‘lines and filled lines’), the next contains a representation of disk usage statistics using a cockpit-style dial, and also shows an example of a ‘block-style’ display with information about characters read from and written to disk. The horizontal bar chart in the next panel shows statistics from the virtual memory manager (the panel

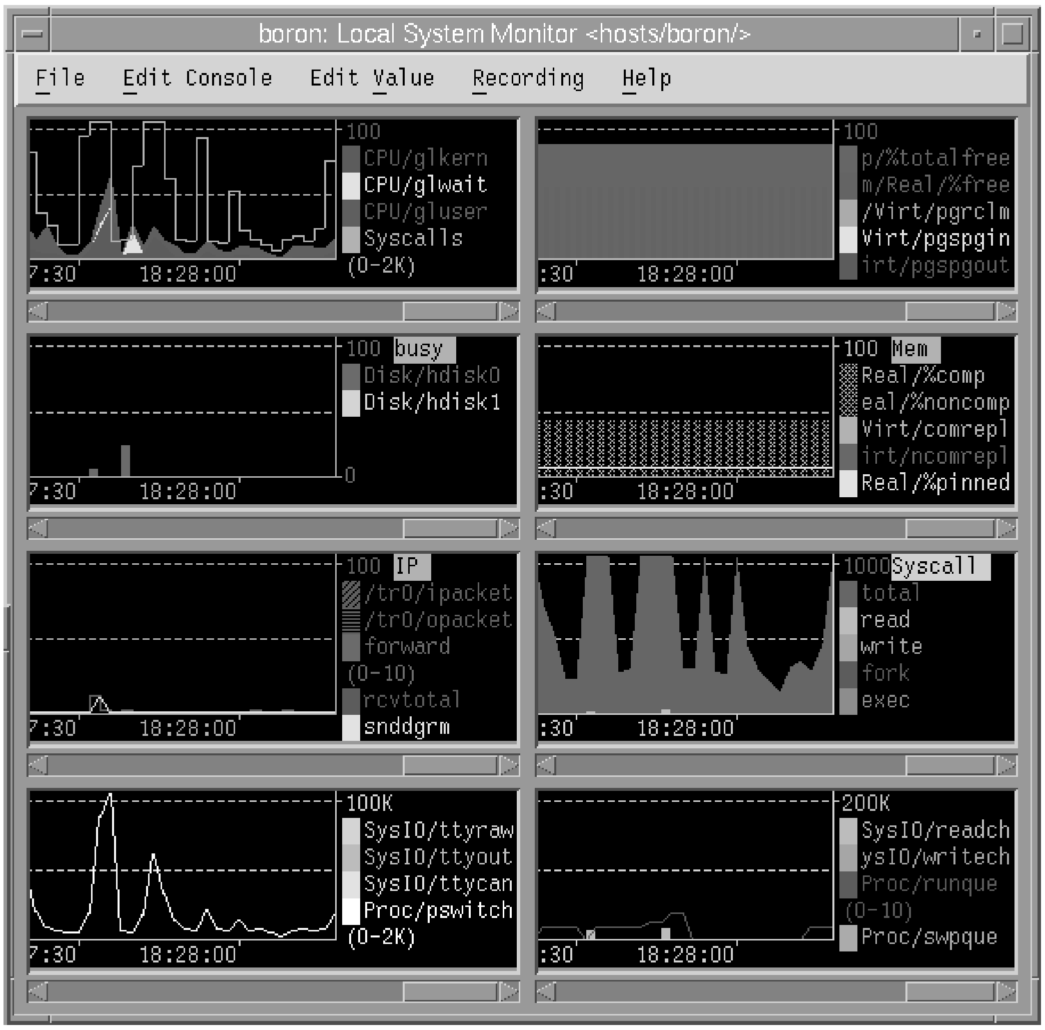


Figure 4: The Local System Monitor

also contains a pie chart that shows memory contention). The last panel contains a line chart showing a number of performance indicators displayed in a conventional way. Another useful skeleton console worth having a closer look at is Local System Monitor (see Figure 4).

Using the menu item *Add New Console*, **xmperf** allows you to build new consoles from scratch. Initially an empty console is displayed, along with a number of pull-down menus that allow you to customize the console to suit your needs (see Figure 5). To populate the console with instruments, choose either *Add Local Instrument* or *Add Remote Instrument* from the *Edit Console* menu. Selecting either of these

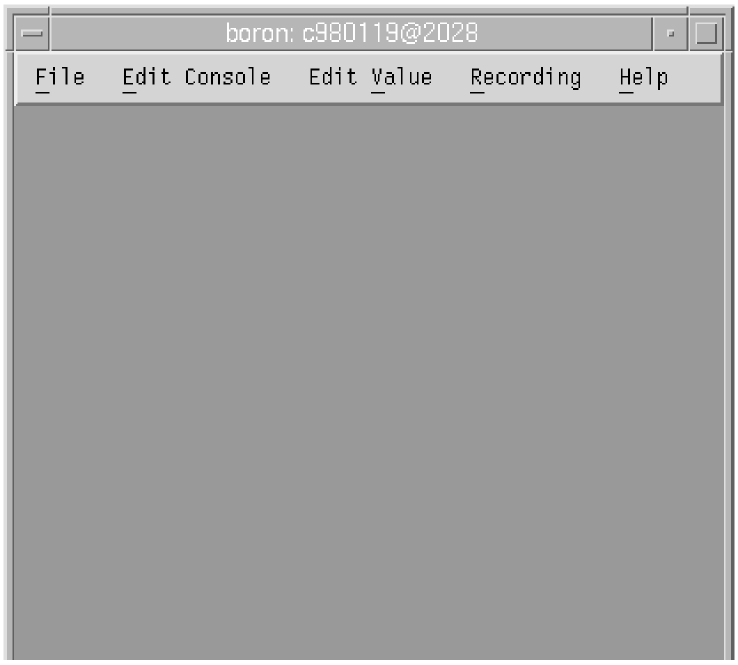
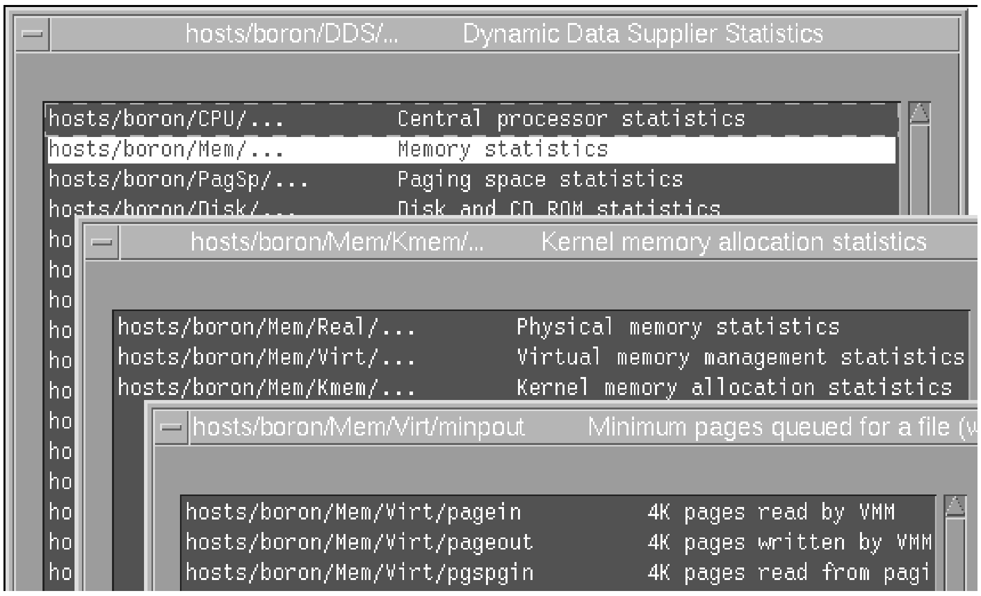
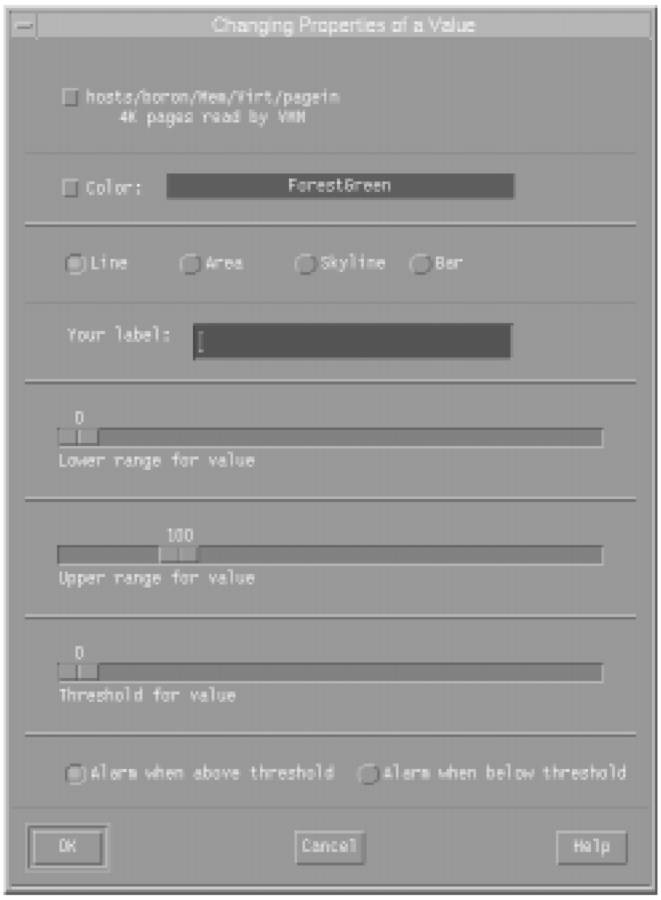


Figure 5: Surface for designing custom console



```
hosts/boron/DDS/...        Dynamic Data Supplier Statistics

hosts/boron/CPU/...        Central processor statistics
hosts/boron/Mem/...        Memory statistics
hosts/boron/PagSp/...      Paging space statistics
hosts/boron/Disk/...       Disk and CD ROM statistics
ho
ho
ho
ho
ho
ho
ho
ho
ho
ho
ho
ho

hosts/boron/Mem/Kmem/...      Kernel memory allocation statistics

hosts/boron/Mem/Real/...      Physical memory statistics
hosts/boron/Mem/Virt/...      Virtual memory management statistics
hosts/boron/Mem/Kmem/...      Kernel memory allocation statistics

hosts/boron/Mem/Virt/minpout      Minimum pages queued for a file (w

hosts/boron/Mem/Virt/pagein       4K pages read by VMM
hosts/boron/Mem/Virt/pageout      4K pages written by VMM
hosts/boron/Mem/Virt/pgspgin      4K pages read from pagi
```

Figure 6: Performance indicators available for monitoring



Changing Properties of a Value

hosts/boron/Mem/Virt/pagein
4K pages read by VMM

Color: ForestGreen

[✓ Line] Area Skyline Bar

Your label:

0
Lower range for value

100
Upper range for value

0
Threshold for value

[✓ Alarm when above threshold] Alarm when below threshold

Figure 7: Setting an instrument's properties

menu items presents you with a list of available statistics, which are organized in sets. Selecting one of these items brings up a list of the available subsets of statistics, and choosing one of these brings up a list of statistics in the subset (see Figure 6). Click an item to select it, which brings up another dialogue where you set properties for the value being displayed in the instrument (see Figure 7).

Once you've set properties for this performance indicator, the first dialogue (the top one in Figure 6) is displayed again, allowing you to add other indicators to this instrument. Every indicator is shown in the instrument immediately it's been created. When you've finished adding indicators, click the *End Selection* button, and the instrument is ready for viewing. The list of instruments is shown again, allowing you to add more instruments to the console.

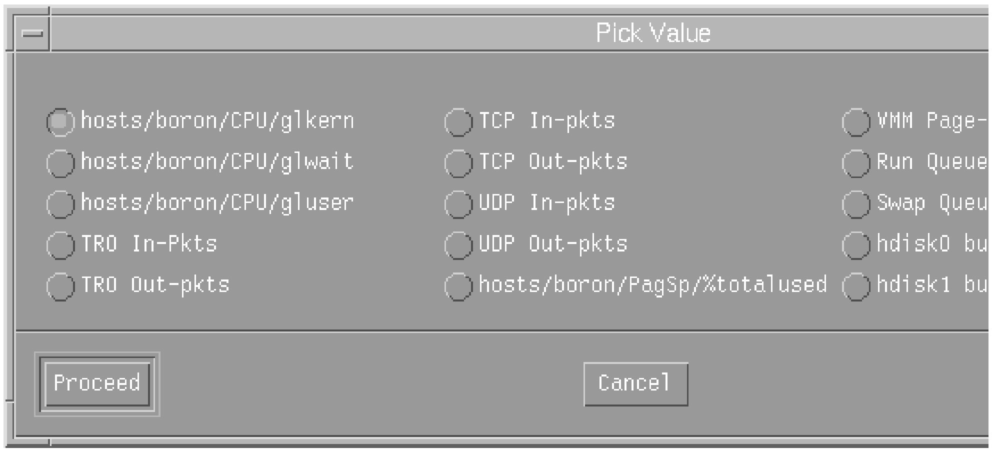


Figure 8: Selecting a value to edit

Display options for an instrument may be edited using the item *Modify Instrument* in the *Edit Console* menu. Simply click on the instrument you wish to change (the instrument is indicated by a dashed line around it) and choose *Modify Instrument*. The dialogue displayed is the same one that's used when defining the instrument.

Using the *Edit Value* menu you can edit the display options for a value (performance indicator) in an instrument. If two or more values are being displayed in the same instrument, a dialogue box allows you to choose which one to edit (Figure 8). After selecting a value using the radio buttons in the dialogue, click *Proceed*, which returns you to the properties dialogue for the chosen value, allowing you to change its display options.

Returning to *Monitor* in the main window, an interesting item in this menu is *Recording*. Using this you can record instruments in the console window for replay later (replay by choosing *File* then *Replay* from the main window). Naturally, if you'd like to be able to replay performance measurements, you should first ensure that they're stored on a file. Do this using the item *Save...* in the *Recording* menu. When replaying measurements, several options are available to allow fine-grained control, for example allowing you to alter the speed of replay. Figure 9 shows the Local System Monitor being replayed.

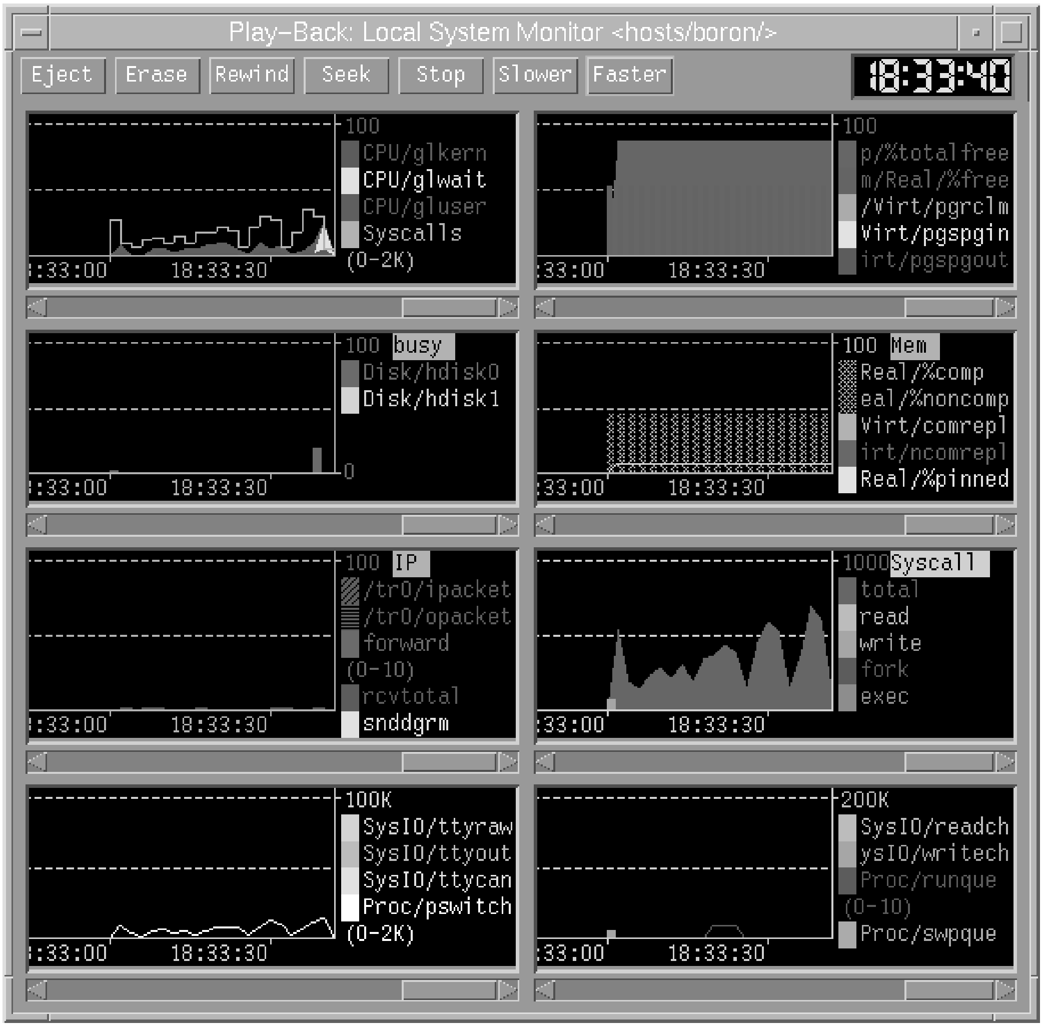


Figure 9: Using the Replay feature

Notice that the time when the recording was made is included in the display, allowing you to correlate events and applications to the performance of the system.

THE ANALYSIS MENU

The *Analysis* menu offers a number of pre-defined methods for analysing common performance conditions. This menu is one of those that offers a wide range of customization options, though I think the

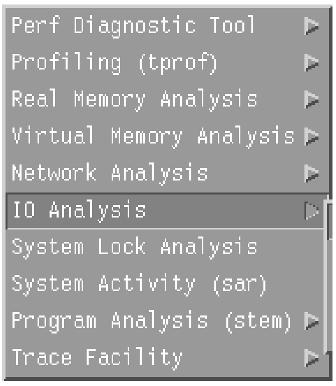


Figure 10: The Analysis menu

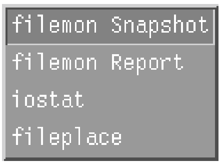
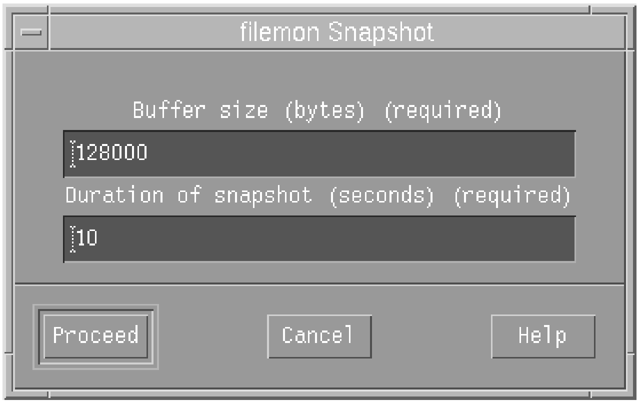


Figure 11: The I/O submenu


filemon Snapshot

Buffer size (bytes) (required)

128000

Duration of snapshot (seconds) (required)

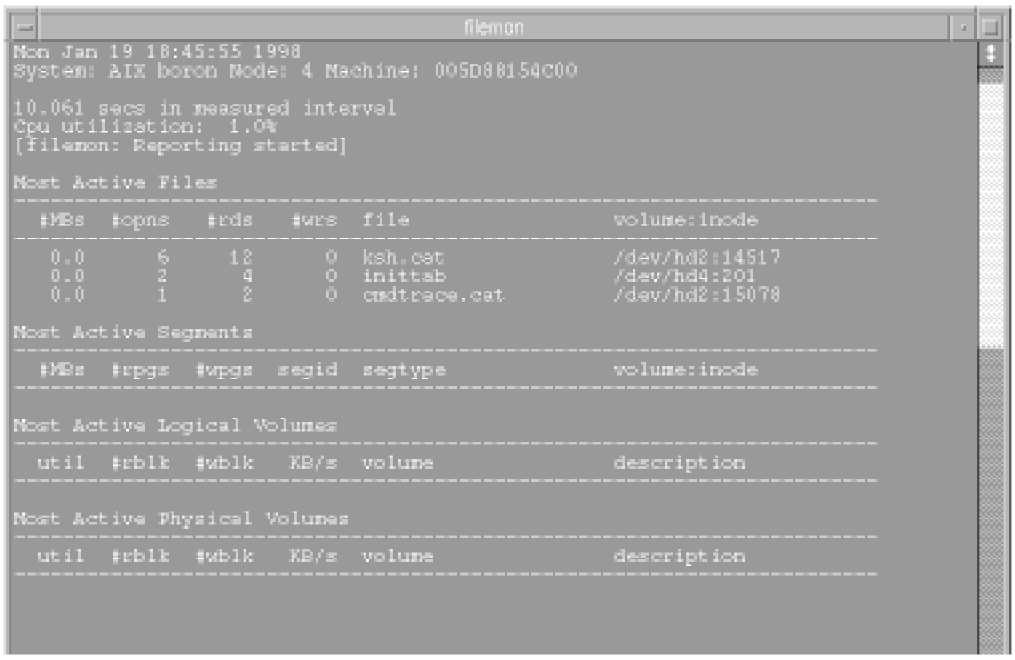
10

Figure 12: The filemon Snapshot tool

standard facilities are likely to be sufficient in most cases. Should you wish to configure the menu, the method used employs a well-documented command language (its documentation is in */usr/share/man/info/en_US/perfagent*).

As an example, let's see how you'd analyse I/O using this menu. The *Analysis* menu provides you with a list of pre-configured analytical tools (see Figure 10). Choosing *IO Analysis* opens a submenu (Figure 11) that allows you to select a particular tool – the *filemon Snapshot* tool in the example (see Figure 12). Once the tool is running, the next dialogue asks you to enter the buffer size and the duration of the snapshot in seconds. The default values are suitable for providing a quick overview of this tool, so I've left them as they are.

Clicking *Proceed* runs **filemon** from the graphical interface using the default values, with the output displayed on an 'aixterm' (Figure 13). To stop processing use **tracestop**, which closes the window. As indicated by the vertical scroll bar in Figure 13, this command produces more output than is shown in a window – **filemon** shows a



```
Mon Jan 19 18:45:55 1998
System: AIX boron Node: 4 Machine: 005D88154C00

10.061 secs in measured interval
Cpu utilization:  1.0%
[filemon: Reporting started]

Most Active Files
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                    volume:inode
------------------------------------------------------------------------
   0.0      6     12      0  ksh.cat                 /dev/hd2:14517
   0.0      2      4      0  inittab                 /dev/hd4:201
   0.0      1      2      0  cmdtrace.cat            /dev/hd2:15078

Most Active Segments
------------------------------------------------------------------------
  #MBs  #rpgs  #wpgs  segid  segtype                 volume:inode
------------------------------------------------------------------------

Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                  description
------------------------------------------------------------------------

Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                  description
------------------------------------------------------------------------
```

Figure 13: The results of filemon Snapshot

summary of the most active files, segments, logical volumes, and so on, thus providing a view of I/O activity during the specified interval.

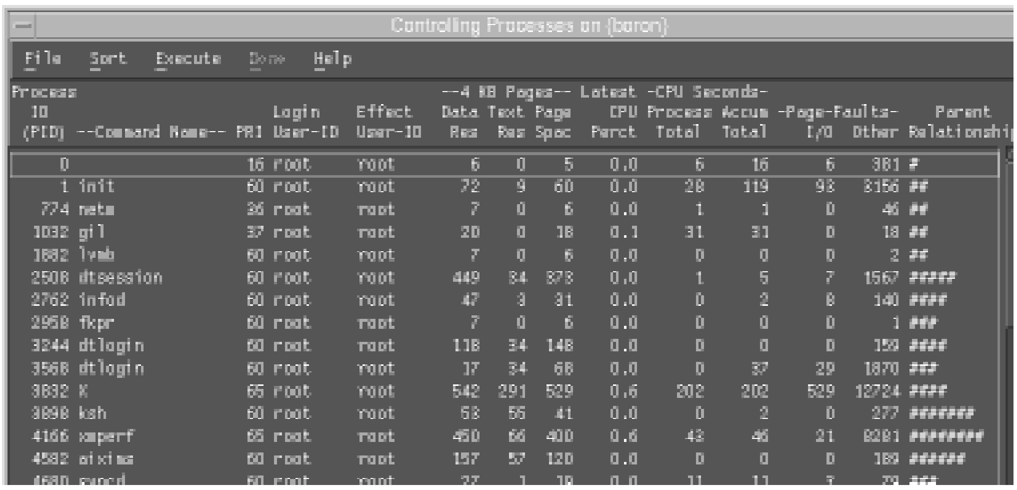
THE CONTROLS MENU

The *Controls* menu is the second of the configurable menus. While offering fewer options than other menus, its options are nonetheless very useful. *Controls* is customized using the same method as the *Analysis* menu.

Controls presents you with three options: *Process Controls, Network Tuning,* and *Program Optimization.*

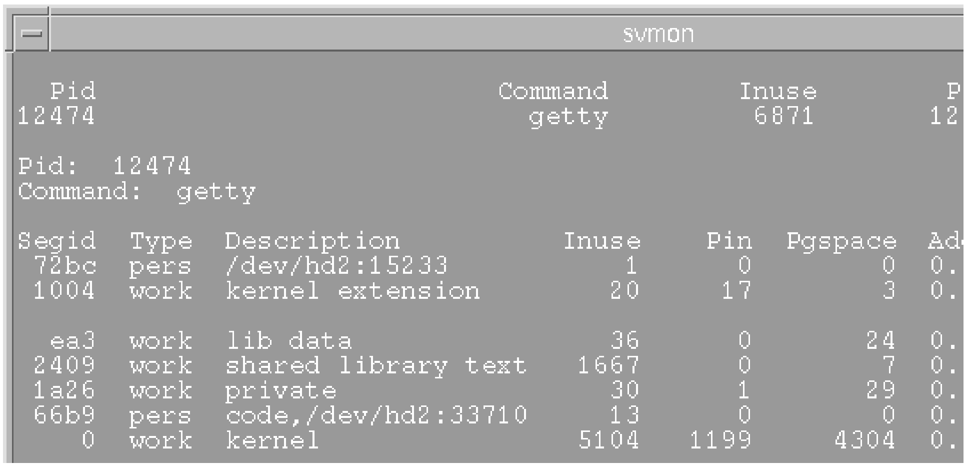
Process Controls opens a window in which all running processes are displayed (Figure 14). Using the *Sort* menu, processes may be sorted by various fields, including process id, owner, and process hierarchy. The *Execute* menu allows you to run various commands on marked processes, such as changing priority settings and stopping processes with **kill**. You can use this menu to run **svmon** on a process. This provides some interesting statistics pertaining to the process, including the number of segments belonging to the process and their ids, as well as the address range in which the process runs (see Figure 15).

The *Network Tuning* option lets you change settings to optimize



```
Process                                  --4 KB Pages-- Latest -CPU Seconds-
  ID                       Login   Effect   Data Text Page   CPU Process Accum -Page-Faults-   Parent
 (PID) --Command Name-- PRI User-ID User-ID   Res  Res Spac  Perct  Total  Total   I/O  Other Relationshi
     0                   16 root    root        6    0    5   0.0      6     16     6    381 #
     1 init              60 root    root       72    9   60   0.0     28    119    93   8156 ##
   774 netm              36 root    root        7    0    6   0.0      1      1     0     46 ##
  1032 gil               37 root    root       20    0   18   0.1     31     31     0     18 ##
  1882 lvmb              60 root    root        7    0    6   0.0      0      0     0      2 ##
  2508 dtsession         60 root    root      449   84  878   0.0      1      5     7   1567 #####
  2762 infod             60 root    root       47    3   31   0.0      0      2     8    140 ####
  2958 fkpr              60 root    root        7    0    6   0.0      0      0     0      1 ###
  3244 dtlogin           60 root    root      118   34  148   0.0      0      0     0    159 ####
  3568 dtlogin           60 root    root       17   34   68   0.0      0     37    29   1870 ###
  3832 X                 65 root    root      542  291  529   0.6    202    202   529  12724 ####
  3898 ksh               60 root    root       53   55   41   0.0      0      2     0    277 #######
  4166 xmperf            65 root    root      450   66  400   0.6     43     46    21   8281 ########
  4582 aixims            60 root    root      157   57  120   0.0      0      0     0    169 ######
  4680 [illegible]       60 root    root       27    1   18   0.0     11     11     3     73 ###
```

Figure 14: Current processes displayed in Process Controls



```
svmon

  Pid                         Command       Inuse       P
12474                           getty        6871      12

Pid:  12474
Command:  getty

Segid  Type  Description          Inuse    Pin  Pgspace  Ad
 72bc  pers  /dev/hd2:15233           1      0        0  0.
 1004  work  kernel extension        20     17        3  0.

  ea3  work  lib data                36      0       24  0.
 2409  work  shared library text   1667      0        7  0.
 1a26  work  private                 30      1       29  0.
 66b9  pers  code,/dev/hd2:33710     13      0        0  0.
    0  work  kernel                5104   1199     4304  0.
```

Figure 15: Using ***svmon*** *to display process statistics*

network traffic under various conditions, while the *Program Optimization* option shows you how a process behaves in different environments (for instance, by running a limited memory simulation to see how process performance is affected by memory constraints).

THE UTILITIES MENU

Utilities (see Figure 16) contains tools for monitoring system events and for analysing the data produced. Also in this menu is a utility for

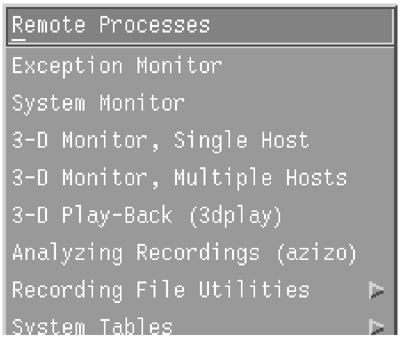
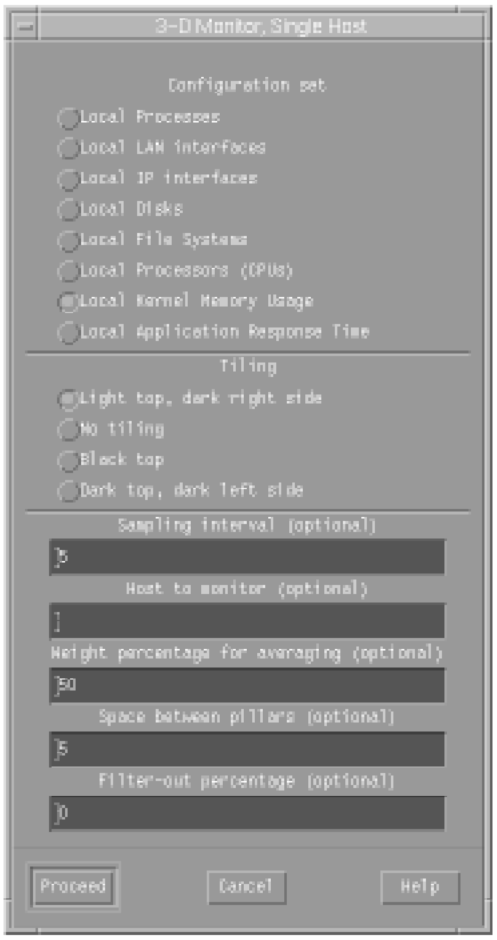


Figure 16: The Utilities menu



3-D Monitor, Single Host

Configuration set

- Local Processes
- Local LAN interfaces
- Local IP interfaces
- Local Disks
- Local File Systems
- Local Processors (CPUs)
- [✓ Local Kernel Memory Usage]
- Local Application Response Time

Tiling

- [✓ Light top, dark right side]
- No tiling
- Black top
- Dark top, dark left side

Sampling interval (optional): 5

Host to monitor (optional):

Weight percentage for averaging (optional): 50

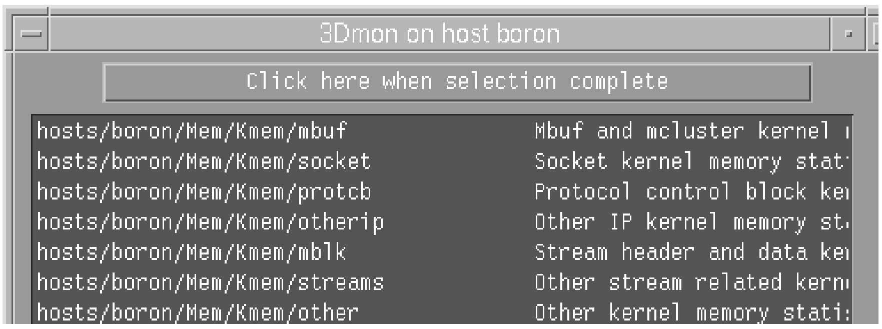
Space between pillars (optional): 5

Filter-out percentage (optional): 0

Figure 17: 3D Monitor for a single host

showing the status of the process that collects data over the network, **xmservd**, along with various statistics about it.

As Figure 16 shows, the most prominent tool in this menu is **3dmon**, which is accessible via the *3D Monitor* submenus. In the example that follows I've selected the option 3*D Monitor, Single Host*, which opens a dialogue for the host, presenting the configuration set, tiling, and interval for collecting data (see Figure 17). Clicking *Proceed* opens dialogue with a list of values that can be selected for monitoring (Figure 18). Once selection is complete, you proceed to the 3D performance display, which is shown in a new window (Figure 19). 3D Monitor provides an impressive view of system events, also



```
3Dmon on host boron

Click here when selection complete

hosts/boron/Mem/Kmem/mbuf          Mbuf and mcluster kernel
hosts/boron/Mem/Kmem/socket        Socket kernel memory stat
hosts/boron/Mem/Kmem/protcb        Protocol control block ke
hosts/boron/Mem/Kmem/otherip       Other IP kernel memory st
hosts/boron/Mem/Kmem/mblk          Stream header and data ke
hosts/boron/Mem/Kmem/streams       Other stream related kern
hosts/boron/Mem/Kmem/other         Other kernel memory stati
```

Figure 18: Values for monitoring with 3D Monitor

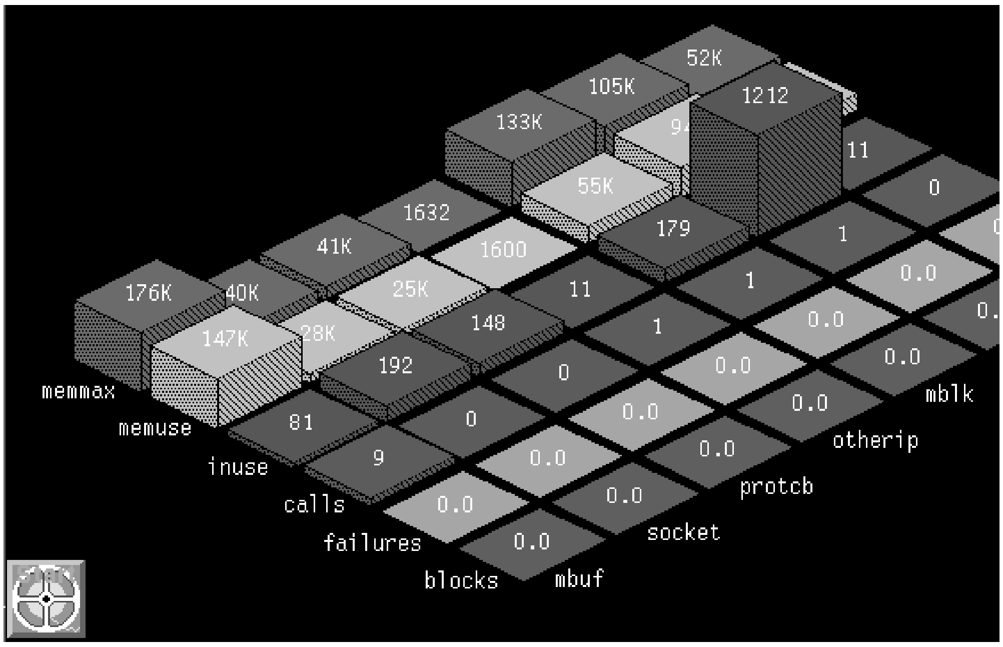


Figure 19: 3D Monitor's performance display window

allowing you to record events. A *Replay* facility in the file menu allows you to view data recorded by this tool.

While 3D Monitor is not the best type of display for those that require detailed quantitative information, it is (in my view) the most intuitive.

I find it much easier to find dependencies between processes when data is presented in this format than when it's in tabular form.

SUMMARY

The Performance Tool Box's **xmperf** is one of the most useful programs for those that want to monitor performance and tune their system. It's a 'Swiss Army Knife' of a tool, integrating a number of other PTX tools, offering an intuitive user interface, and displaying data in an attractive way.

xmperf is also extremely customizable, offering great flexibility for integrating performance measuring tools delivered with either AIX or PTX with those written by system programmers. If you need to manage performance for a specific system configuration or intend to optimize your programs to run faster, then **xmperf** is definitely worth a look.

Peter Wuestefeld
ResNova Consult (Germany)

netmon

netmon is a simple script that allows you to monitor traffic through an Ethernet or ATM network adapter. The information displayed by **netmon** includes the number of packets and the number of bytes.

To call the script, use the syntax below.

```
./netmon [AdapterName]
```

AdapterName is an optional parameter that specifies the name of the network adapter. If this parameter is omitted, **netmon** displays statistics for the *en0* adapter.

netmon's sampling interval may be modified by changing the value of *INTV* in the script (the default is one second).

To stop the monitor, use the key sequence *CTRL-C*.

EXAMPLES

To display statistics for the *en0* Ethernet adapter, use the command:

```
    $ ./netmon en0
```

To display statistics for the *at0* ATM adapter, use the command:

```
    $ ./netmon at0
```

EXAMPLE OUTPUT

Below is a sample output of this command.

```
P A C K E T S          B Y T E S     KBYTES PER 1 SEC

    xmit  receive      xmit  receive      xmit  receive

       3        8       162     3160         0        3

       5       10       270     3380         0        3

       4        7       216     2594         0        2
```

THE SCRIPT

```
#!/bin/ksh
#
# netmon - a network traffic monitor
#
# Command: netmon adapter
#
# Restrictions: Do not reset adapter statistics while
# running the script.
#

INTV=1  # Interval time in seconds

if [[ $1 = "" ]]; then
        adapt=en0
else
        adapt=$1
fi

x=`echo $adapt|cut -b1`
if [[ $x = "a" ]]; then
        CMD=/usr/bin/atmstat
```

```
else
        CMD=/usr/bin/entstat
fi

#
# collectinfo: collect information using the entstat command
#
collectinfo() {

$CMD -d $adapt | while read type1 amount1 type2 amount2 trash
do
        if [[ $type1 = "Packets:" ]]; then
                xmit_packets=$amount1
                rcv_packets=$amount2
        elif [[ $type1 = "Bytes:" ]]; then
                xmit_bytes=$amount1
                rcv_bytes=$amount2
                break;
        fi
done

}

#
# mainline
#

collectinfo
xmit_packets1=$xmit_packets
rcv_packets1=$rcv_packets
xmit_bytes1=$xmit_bytes
rcv_bytes1=$rcv_bytes

printf "    P A C K E T S         B Y T E S     KBYTES PER %d SEC\n"
$INTV
printf "    xmit  receive     xmit  receive     xmit  receive\n"
while true
do
        sleep $INTV
        collectinfo
        ((xmit_packets2=$xmit_packets-$xmit_packets1))
        ((rcv_packets2=$rcv_packets-$rcv_packets1))
        ((xmit_bytes2=$xmit_bytes-$xmit_bytes1))
        ((rcv_bytes2=$rcv_bytes-$rcv_bytes1))
        ((xmit_bytes3=$xmit_bytes2/1024))
        ((xmit_bytes3=$xmit_bytes3/$INTV))
        ((rcv_bytes3=$rcv_bytes2/1024))
        ((rcv_bytes3=$rcv_bytes3/$INTV))
        printf "%8d %8d %8d %8d %8d %8d\n" $xmit_packets2 $rcv_packets2
$xmit_bytes2 $rcv_bytes2 $xmit_bytes3 $rcv_bytes3
```

```
        xmit_packets1=$xmit_packets
        rcv_packets1=$rcv_packets
        xmit_bytes1=$xmit_bytes
        rcv_bytes1=$rcv_bytes
done
```

System Programmer (Switzerland) © Xephon 1998

lspid

lspid is a command, implemented as a shell script, that displays a process and its parents as a parent-child sequence.

This type of command is very useful when you are diagnosing problems and need to know a process's parents. Another command that's useful in this type of situation is the public domain utility **pstree**, which allows you to take a quick look at a process's tree structure. However, **pstree** lacks the ability to display information about process resource utilization. For this reason I created the **lspid** command.

lspid displays process tree structure, from parent to child. For each process detailed information on the paging, memory, and CPU utilization is displayed. Use the syntax below to run the command, supplying the process id (*pid*) of the process about which you wish to know (*14672* in the example).

```
    $ ./lspid 14672
```

SAMPLE OUTPUT

Figure 1 overleaf shows a sample output of this command.

THE SCRIPT

```
#!/bin/ksh
#
# Display a ps tree
#
# Usage: lspid pid
```

```
PID   TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
 2484   - A     0:00   26   232     8    xx    96     0  0.0  0.0 /usr/dt/bin/dtlogin -daemon
 3282   - A     0:00   40   548     8    xx    96     0  0.0  0.0 dtlogin <:0>        -daemon
 3378   - A     2:45 6710  7924   368 32768    94    16  0.0  0.0 /usr/dt/bin/dtsession
10878   - A     2:12 3564  1960  1384 32768   436   288  0.0  1.0 dtwm
14672   - A     0:00    1   172    12 32768    12     0  0.0  0.0 /usr/dt/bin/dtexec -open 0 -
ttprocid 2.q


$ pstree -p 14672


-+- 00001 root /etc/init
 \-+- 02484 root /usr/dt/bin/dtlogin -daemon
   \-+- 03282 root dtlogin <:0>        -daemon
     \-+- 03378 fred /usr/dt/bin/dtsession
       \-+- 10878 fred dtwm
         \-+- 14672 fred /usr/dt/bin/dtexec -open 0 -ttp
```

Figure 1: Sample output of ***lspid***

```
#
pid=$1

if [[ $1 = "" ]]; then
        echo "Usage: "$0" pid"
        exit 1
fi

tmp=/tmp/$$

rm $tmp >/dev/null 2>&1

while [[ $pid != 1 ]]
do
        ps -f -o'%p %P' -p $pid | tail -1 | read pid parent
        echo $pid >>  $tmp
        pid=$parent
done

# Display header

ps gvww 1 | head -1
tail -r $tmp | while read pid ppid
do
        ps gvww $pid | tail -1
done
```

System Programmer (Switzerland) © Xephon 1998

Splitting large files

Have you ever needed to copy a large file to a floppy disk, but the file's just too large? For example, suppose you have an 8 mm Exabyte tape drive and your customer uses 4 mm DAT tape – the easiest way to transfer data between the two systems is by floppy. Or suppose you've downloaded Netscape Navigator to your AIX system using your business's Internet connection, and now you want to transfer it to your Microsoft Windows PC by floppy. However, the software takes up several megabytes and won't fit on a floppy disk. What do you do now?

SPLIT COMMAND

AIX implements the Unix **split** command that allows you to split a file into parts. This command breaks files into pieces of whatever size you select – hence its name. Later you can reassemble the pieces back into a single file.

The **split** command reads the specified file and writes it in 1000-line blocks to a set of output files. The name of the first output file is generated by combining a specified prefix (‘x’, by default) with the suffix ‘aa’; the second file has the same prefix along with the suffix ‘ab’, and so on to ‘zz’, yielding a maximum of 676 files. The number of letters in the suffix, and consequently the number of output name files, can be increased using the **-a** flag. The **-l** parameter lets you specify the number of lines in each output file, the default being 1000.

If you’re working with binary files then the **-b** ***Number*** flag is used to specify the number of bytes each output file should contain. Appending a ‘k’ (kilobyte) or ‘m’ (megabyte) to ***Number*** causes the original file to be split into output files of ***Number*** times 1024 or ***Number*** times 1,048,576 bytes respectively.

EXAMPLES

- To split the file *book* into 1000-line segments, use the command:

  ```
  split book
  ```

  This splits *book* into files called *xaa*, *xab*, *xac*, and so on.

- To split *book* into 50-line segments and specify that the output file names are to have the prefix *sect*, use the command:

  ```
  split -l 50 book sect
  ```

  This splits *book* into 50-line segments named *sectaa*, *sectab*, *sectac*, and so on.

- To split *book* into 2 KB segments, use the command:

  ```
  split -b 2k book
  ```

  This splits *book* into 2048-byte segments named *xaa*, *xab*, *xac*, and so on.

- To split *book* into more than 676 segments, use the command:

  ```
  split -l 5 -a 3 book sect.
  ```

  This splits *book* into 5-line segments named *sect.aaa, sect.aab, sect.aac*, and so on, up to *sect.zzz* (a maximum of 17,576 files). Notice the period (.) after ‘sect’ in the command.

Use **info** or **man** to obtain more information on **split**.

SPLITTING FILES FOR FLOPPY DRIVES

If you’re using a 1.44 MB floppy disk drive, it makes sense to split files into 1.4 MB chunks. To do this, use the command:

```
split -b 1400k BigFile chunk.
```

This results in files named *chunk.aa*, *chunk.ab*, *chunk.ac*, and so on. Each file is 1,433,600 bytes long with the exception of the last one, which contains the remaining bytes. You can now copy the smaller files to your floppy disks. Use **dosread** if you’ll need to read the floppy disks with your PC later.

REASSEMBLING THE FILE

On an AIX system, use **cat** to reassemble the segments back into *BigFile*.

```
cat chunk.aa chunk.ab chunk.ac [and so forth] > BigFile
```

cat sends the output of each file, in the specified order, to the standard output, which is then re-routed to *BigFile*. When the filename includes wildcards, your shell automatically sorts the matching files before feeding them to **cat**. As **split**’s output is in alphabetical order, this is a convenient way to reassemble files.

```
cat chunk.* > BigFile
```

This is much simpler than listing the files individually, especially when you have numerous *chunk* files.

Use **copy** along with its ‘+’ parameter (it’s a parameter, even though it looks and behaves like an operator) to reassemble the *chunk* files on a PC:

```
copy chunk.aa + chunk.ab + chunk.ac ... BigFile
```

MS-DOS displays each filename as the file is copied. Note that, while DOS allows you to use wildcards to specify filenames in the same way as AIX does, there's no guarantee that the files are reassembled in the right order, as neither the copy command nor the DOS shell sorts them first, instead copying files in the order in which they are found.

Werner Klauser
Klauser Informatik (Switzerland)

Setting up a 7025/F50

If you're setting up an RS/6000 F50 server for unattended operation (typically because the unit is located where no-one with the right skill set is available), don't forget to customize the internal service processor.

So, what do you do to ensure the system is in unattended operation mode?

1 First, start **smit** and navigate to the screen labelled *Change/Show Characteristics Of Operating System* and set the *Automatically Reboot System After A Crash* flag to true.

2 Next, shut down the system using **shutdown -F**.

3 When the service processor *Firmware Main Menu* comes up, make the following menu selections:

   2 (*System Power Control Menu*)

   1 (*Enable/Disable Unattended Start Mode*)

   4 (*Power On System*).

4 Test the setting (this may be something that you're reluctant to do, but you need to test the change to ensure that it's properly implemented and will work when the system is unattended): pull out the main power supply lead and reconnect it after a pause to

simulate a power failure. The system should come up without any user intervention.

If you do not customize the service processor, the system will not boot after a power failure. This means you'll have to find a competent operator who can be relied upon to restart the server (hopefully this just requires them to push the power button). However, remember that you may have to find such a person in the middle of the night at a remote location without access to the site yourself.

Also note that, if you implement the unattended operation mode after going into production, you'll have to shut down the system and hence take it off-line. This means reduced system availability and unhappy users.

Michael Imhotep (Australia) © Xephon 1998

Contributing to *AIX Update*

AIX Update is primarily written by practising AIX specialists in user organizations – not journalists, or consultants, or marketing people. In our view, the information and advice provided by such people – people like you and your colleagues – are far more valuable to their fellow professionals than the alternatives available from other sources.

We pay good rates for the articles we publish: $250 (£170) per 1000 words if we get copyright, and $140 (£90) per 100 lines of code.

A copy of *Notes for contributors* can be downloaded from Xephon's Web site at *www.xephon.com*. Articles for submission to *AIX Update* can be sent to the editor, Harry Lewis, at *HarryLewis@compuserve.com.*

AIX news

IBM has announced the second release of its San Francisco Java business process components framework, with new modules for building order processing and warehouse/inventory management applications.

The new release provides national language support, more sample JavaBeans, performance enhancements to the Foundation layer, a framework for building GUIs to San Francisco-built applications, and support for additional deployment platforms and data formats.

The new components cost US$470 per seat and, in addition to AIX, also run on OS/400 and NT.

Also new is Net.Commerce Pro, which is aimed at large corporates, and includes catalogue tools and back-end integration tools for accessing software such as CICS, MQSeries, IMS, SAP R/3, and EDI. It runs on AIX, and also on OS/390, NT, and Solaris.

For further information contact your local IBM representative.

* * *

One way of running 32-bit Windows applications from AIX systems is to configure AIX as a client of an NT server. To enable this, Citrix has announced AIX workstation clients based on its Independent Computing Architecture (ICA) technology for 'thin-client/server' computing. This allows AIX users to run applications remotely on the NT system. Other Unix clients supported are Solaris, HP-UX, Digital Unix, and IRIX workstations.

For further information contact:
Citrix Systems, 6400 Northwest 6th Way, Fort Lauderdale, FL 33309, USA
Tel: +1 954 267 3000
Fax: +1 954 267 9319
Web: http://www.citrix.com

Citrix Systems UK Ltd, Eagle House, The Ring, Bracknell, Berks RG12 1HB, UK
Tel: +44 1344 382100
Fax: +44 1344 382136

* * *

Candle has announced a version of its Command Center for R/3, geared to increasing performance and availability of R/3 applications, with promises to reduce downtime and increase application response times. The product is available for AIX, HP-UX, Solaris, and Digital Unix, with support for Oracle and Informix.

Out now, prices start at US$20,000.

For further information contact:
Candle Corp, 2425 Olympic Boulevard, Santa Monica, CA 90404, USA
Tel: +1 310 829 5800
Fax: +1 310 582 4287
Web: http://www.candle.com

Candle Ltd, 1 Archipelago Way, Lyon Way, Frimley, Camberley, Surrey GU16 5ER, UK
Tel: +44 1276 414700
Fax: +44 1276 414777

xephon