AIX update

116

June 2005

In this issue

A tc PUBLICATION

AIX Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues
A year's subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

AIX Update on-line
Code from *AIX Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/aix; you will need to supply a word from the printed issue.

Contributions
When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Using Perl and find

INTRODUCTION

We often need to make search-and-replace-style changes in text files. If we need to do this in one file, it isn't a problem. Using a favourite editor such as vi, we can issue a command like:

```
:1,$:s/old_string/new_string
```

This would substitute only the first occurrence of *old_string* on a line with *new_string*.

or:

```
:1,$:s/old_string/new_string/g
```

This would substitute all occurrences of *old_string* with *new_string* on a line (commonly known as 'global search and replace').

Another way to do this would be to use **sed**:

```
# sed -e 's/old_string/new_string/' inputfile > outputfile
```

or:

```
# sed -e 's/old_string/new_string/g' inputfile > outputfile
```

Now, we would need to rename the temporary work file back to the original name:

```
# mv outputfile inputfile
```

This works nicely except that we end up having a temporary work file and two commands to do the job.

A SIMPLER WAY

A cool option that I like to use in Perl is the **-i** flag, which specifies that files processed are to be edited in-place. Perl does this by renaming the input file, opening the output file using the original name, and selecting that output file as the default for print statements. The extension, if supplied, is added to the name of the old file in order to make a back-up copy. If no extension is supplied, no back-up is made:

```
# perl -i -p -e "s/old_string/new_string/" inputfile
```

But, what if we have multiple files to modify or, better yet, we don't know which files need modifying?

By combining this Perl one-liner with the **find** command, we get:

```
# find . -name "*.html" -exec perl -i -p -e "s/old_string/new_string/"
{} \;
```

This would modify all HTML files with the string *old_string* to *new_string*.

We can do the same using a global search and replace in all Perl scripts:

```
# find . -name "*.pl" -exec perl -i -p -e "s/old_string/new_string/g" {}
\;
```

Another example uses **find** to search for all the files starting from a specified directory (eg */mystartdir*):

```
# find /mystartdir -type f -exec perl -i -p -e "s/old_string/new_string/
g" {} \;
```

The last example will make a back-up of the inputfile by adding an extension of **.bak**:

```
# perl -i.bak -p -e "s/old_string/new_string/" inputfile
```

These quick one-liners have saved me hours of work and I hope they help you as well!

Elvio Prattico
Consultant
PRATTICO Consulting (Canada)

Exploring performance tool improvements in AIX Version 5.3

The AIX operating system provides a very rich set of system performance monitoring and tuning facilities. When a new version of the operating system is introduced, these facilities undergo modification and expansion to accommodate new and enhanced features of the kernel. The revolutionary features introduced by AIX 5.3, such as micro partitioning and SMT, demanded significant adjustments in this area – which I will cover in this article.

One of the areas that called for change was the accommodation of configuration changes to the computers in the course of the work of different utilities. As a result of this change, the commands **iostat**, **vmstat**, **sar**, as well as the newly-introduced **mpstat** and **lparstat** utilities, display a system configuration line, which appears as the first line displayed after the command is invoked. If a configuration change is detected during command execution, a warning line will be displayed before the data, which is then followed by a new configuration line and the header. As a result of these changes, scripts that parse command output may need to be modified to properly detect and handle the new output format.

Many new tunable parameters have been added to AIX in AIX 5L Version 5.3. Some have also appeared in AIX 5L Version 5.2 ML5, which was made available at the same time. It is always a good policy to check for new options when major system service is applied because these enhancements do not always appear in product documentation. The **-L** option of the **ioo**, **vmo**, **schedo**, **no**, and **nfso** commands can be used to get the current list of tunable parameters. The **-h** option can be used to display their descriptions and usage.

PROCMON – VISUAL SYSTEM MONITORING TOOL

For first time, IBM shipped a graphical performance monitoring tool as a standard part of the operating system. It is based on the IBM Open Source Eclipse platform and is installed as the following file set:

```
bos.perf.gtools.perfwb      5.3.Ø.Ø  COMMITTED  Performance Workbench
```

The command itself is **/opt/perfwb/perfwb**.

The two tabs that are presented for user selection are **Partition Performance** and **Processes**.

The **Partition Performance** tab displays two dynamically-updated graphs. The first one displays the distribution of CPU consumption among User, System, Idle, and Wait modes. The second one presents the total and available memory and swap space. Additional windows list the partition information such as the number of CPUs, kernel bit mode, total number of processes, and the uptime.

The **Processes** tab displays a constantly-updating sorted list of processes and information about them. A variety of process information items such as CPU statistics, memory statistics, PID, PPID, owner, etc can be selected for display. It is possible to sort the table on any of the fields and perform searches with output conveniently highlighted. Any process can be selected to perform one of the following operations: kill, renice, list of threads display, and execution of external performance tools such as svmon.

ASYNCHRONOUS I/O STATISTICS REPORTING

An asynchronous I/O technique has been implemented in AIX since Version 4.3. It has been enhanced in AIX 5.2 by the addition of a POSIX-compatible implementation. This I/O optimization technique is widely used by various database vendors such as Oracle in order to improve the throughput of the I/O. Both legacy and POSIX-based implementations implement a set of kernel processes, called aioservers, that perform I/O on behalf of requests made by applications. In order to use the AIO drivers, we have to enable the AIO device driver using the **mkdev -l aio0** command or through **smit aio**.

In order to enable the POSIX AIO device drivers, you have to use the **mkdev -l posix_aio0** or **smit posixaio** commands.

By using SMIT and going through menus **Devices/ Asynchronous I/O/Asynchronous I/O (Legacy)/Change/ Show Characteristics of Asynchronous I/O** you can set the characteristics of the AIO, like minservers, maxservers, maxreqs, kprocprio, autoconfig, or fastpath.

In the previous versions of AIX there were no tools available to monitor the AIO; from Version 5.3 the **iostat** command is enhanced to monitor the AIO.

The **iostat** command reports CPU and I/O statistics for the system, adapters, TTY devices, disks, and CD-ROMs. This command is enhanced by new monitoring features and flags to get the AIO and the POSIX AIO statistics.

The following new flags are added to the **iostat** command:

- **-A** – reports legacy AIO statistics along with utilization metrics.
- **-P** – reports POSIX AIO statistics along with utilization metrics.
- **-q** – reports each AIO queue's request count.
- **-Q** – reports AIO queues associated with each mounted filesystem and the queue request count.
- **-l** – displays the data in a 132 column width.

The iostat -A option

When using the **-A** option, the output from the **iostat** command gives additional statistics about the AIO. The following information is added:

- *avgc* – average global non-fastpath AIO request count per second for the specified interval.
- *avfc* – average AIO fastpath request count per second for the specified interval.

- *maxg* – maximum non-fastpath AIO request count since the last time this value was fetched.
- *maxf* – maximum fastpath request count since the last time this value was fetched.
- *maxr* – maximum AIO requests allowed. This is the AIO device maxreqs attribute.

When the AIO device driver is not configured in the kernel, the **iostat -A** command indicates in an error message that the AIO is not loaded, as in the following example:

```
# iostat -Aq
aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow physc %entc
iostat: Ø551-157 Asynchronous I/O not configured on the system.
```

The next example demonstrates the use of the **iostat -A** command:

```
#:  iostat -A

System configuration: lcpu=4 drives=3

aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow
       25    6   29   1Ø    4Ø96        4.5  22.9  72.4  Ø.3

Disks:        % tm_act     Kbps       tps    Kb_read   Kb_wrtn
hdiskØ           2.Ø      33.4        4.5   122ØØ6486  11716606Ø
hdisk1           Ø.8      42.6        2.4   128209Ø83  17718Ø326
cdØ              Ø.Ø       Ø.Ø        Ø.Ø          Ø         Ø
```

Notice the *avgc* column, which shows the average number of AIO requests in the queues, and the *maxg* column, which shows the maximum number of AIO requests in the queues for the last measuring period. If the *avgc* or *maxg* is getting close to the *maxr*, then tuning of the *maxreqs* and *maxservers* attributes is required.

If you use raw devices, the *avfc* is interesting because it reflects the use of the average AIO fastpath calls. The *maxf* is also interesting because it shows the maximum value of the fastpath count value. Since the fastpath calls bypass the AIO queues, these statistics give information only on how fastpath AIO is used.

The iostat -Aq option

Details for the allocation of the AIO queues and their use are displayed by the **iostat -Aq** command. The following is an abridged example:

```
#iostat -Aq 1 1
aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow
       25    6   29   10      4096        4.5 22.9  72.4  0.3

q[  0]=0       q[  1]=0       q[  2]=0       q[  3]=0       q[  4]=0
q[  5]=0       q[  6]=0       q[  7]=0       q[  8]=0       q[  9]=0
q[ 10]=0       q[ 11]=0       q[ 12]=0       q[ 13]=0       q[ 14]=24
q[ 15]=0       q[ 16]=0       q[ 17]=0       q[ 18]=0       q[ 19]=0
…
```

If statistics for any single queue are significantly higher than the others, this indicates that applications are using one filesystem significantly more than other filesystems. The queues are usually allocated one per filesystem.

The iostat -AQ option

In order to investigate the issue of imbalance between requests submitted to different AIO queues and relate them to specific filesystems, the **iostat -AQ** command is useful. The following example shows the distribution of the AIO queues to specific filesystems:

```
#iostat -AQ 1 1

aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow
       25    6   29   10      4096        3.0 22.2  74.8  0.0

Queue#         Count          Filesystems
129            0              /
130            0              /usr
132            0              /var
133            0              /tmp
135            10             /home
136            0              /proc
137            0              /opt
139            0              /var/adm/csd
140            12             /local
141            0              /local64
…
```

-P flag usage

If a system is configured to use POSIX AIX, the above-mentioned flags should be preceded by a **P** flag. The output is in the same format as that for the **iostat -A** legacy AIO command; the meaning of the metrics is the same as well, but related to the POSIX AIO calls.

The output format of the **iostat -Pq** command corresponds to **iostat -Aq** and the **iostat -PQ** corresponds to the **iostat -AQ** command.

Other iostat -A usage

There may be several possible combinations of the **iostat** flags, but it is better to run the **iostat** command twice in two different windows than use too many flags and get too much information. There are also several unsupported combinations of flags. One example of a reasonable use would be the **iostat -A -l hdisk3 hdisk4 1 10** command that has combined flags.

DISK SERVICE AND WAIT TIME MONITORING

Dkstat and device support

The dkstat structure defined in the *sys/iostat.h* header file is used by the device drivers to provide the statistical data and the performance monitoring tools, such as **iostat -D** or **sar -d**, to read the data.

The dkstat structure is updated in AIX 5L Version 5.3 to support additional statistics related to disk service times.

Avwait and avserv enhancements for the sar -d command

Starting with Version 5.3, the implementation of output produced by the **-d** flag is modified for the **sar** command. The **-d** flag of the **sar** command is intended to give statistics about the I/O activity on the disks. The **sar -d** command reads the dkstat structures of the devices that supply the necessary information.

In older versions of AIX, *avwait* and *avserv* are always reported as zero. From Version 5.3 the *avwait* and *avserv* are implemented. The meaning of the *avque* statistics has changed from Version 5.3. The following is sample output from this command:

```
# sar -d 1 2

AIX myhost 3 5 ØØ573ØDA4CØØ    Ø2/Ø6/Ø5

System configuration: lcpu=4 drives=3

11:24:39     device     %busy     avque     r+w/s     Kbs/s    avwait    avserv

11:24:4Ø     hdiskØ     1ØØ       Ø.Ø       191       762       Ø.Ø       7.6
             hdisk1       Ø       Ø.Ø         Ø         Ø       Ø.Ø       Ø.Ø
                cdØ       Ø       Ø.Ø         Ø         Ø       Ø.Ø       Ø.Ø


11:24:41     hdiskØ      97       Ø.Ø       184       738       Ø.Ø       7.3
             hdisk1       Ø       Ø.Ø         Ø         3       Ø.Ø       9.3
                cdØ       Ø       Ø.Ø         Ø         Ø       Ø.Ø       Ø.Ø


Average      hdiskØ      98       Ø.Ø       187       75Ø       Ø.Ø       7.4
             hdisk1       Ø       Ø.Ø         Ø         1       Ø.Ø       4.6
                cdØ       Ø       Ø.Ø         Ø         Ø       Ø.Ø       Ø.Ø
```

There are two new measured statistics and one that has been modified:

- *avwait* – average wait time per request in milliseconds. This is the time that the request is waiting in the device driver I/O queue to be passed to the I/O device.
- *avserv* – average service time per request in milliseconds. This is the length of time between the I/O being passed to the device and the result being returned from the device.
- *avque* – average number of requests waiting to be sent to disk. Imagine this as the average length of the occupied I/O queue.

There is a change in the *avque* interpretation introduced in Version 5.3. In the previous versions of AIX the *avque* statistics represented the instantaneous number of requests **sent** to disk but not completed.

The other columns, like *Kbs/s*, *r+w/s*, *%busy*, *device*, and so on, have the same meaning as before.

PMAPI M:N PTHREADS SUPPORT

Under the M:N threading model, M user threads are mapped to N kernel threads, with M being typically considerably larger than N to allow large numbers of pthreads to run. Making PMAPI calls from a program running in this mode was previously not supported.

The PMAPI library has been updated to handle the M:N thread model; the current unchanged interfaces simply work in M:N mode. The only significant change is for third-party API callers, for example debuggers, where new interfaces with pid, tid, and ptid must be used.

MICRO PARTITIONING AND SMT SUPPORT

The introduction of micro partitioning and SMT support in AIX 5.3 required enhancement to performance tracking and reporting system utilities.

The new Process Utilization Resource Register (PURR), provided by the POWER5 CPUs, is used to provide an actual count of physical processing time units that a logical processor has used. All performance tools and APIs utilize this PURR value to report CPU utilization metrics for micro partitioning and SMT systems.

Perfstat library enhancements

The performance library API libperfstat is enhanced to capture the new PURR-based utilization statistics.

The following enhancements are introduced by AIX 5L Version 5.3 to the libperfstat library:

- perfstat_cpu_t – the structure defines the basic processor statistics on a per processor basis. The structure is extended with PURR-based utilization and micro partitioning statistics. It has also been extended with statistics displayed by the **mpstat** command, like affinity and interrupt statistics.
- perfstat_cpu_total_t – the structure defines the basic processor statistics system-wide. The structure is extended with PURR-based utilization and micro partitioning statistics. It has also been extended with more detailed interrupt statistics.
- perfstat_partition_total_t – this is a new structure containing partition-specific information and statistics similar to those the **lparstat** command can display.
- perfstat_partition_total() – this is a new function that returns information in a perfstat_partition_total_t structure.

Vmstat command enhancements

The **vmstat** command has been enhanced to support micro partitioning and can now detect and tolerate dynamic configuration changes.

The **vmstat** command has two new metrics that are displayed – physical processor consumed and percentage of entitlement consumed. They are represented as *pc* and *ec* in the output format. The physical processor consumed represents the number of physical processors consumed by the partition during an interval. The percentage of entitlement consumed is the percentage of entitled capacity consumed by a partition during an interval (pc/ent)*100. These new metrics will be displayed only when the partition is running as a shared processor partition. If the partition is running as a dedicated processor partition, the new metrics will not be displayed.

In a change from previous releases (and from other Unix operating system implementations), the first interval of the command output is now meaningful and does not represent statistics collected from system boot. Internal to the command, the first interval is never displayed, and therefore there may be a slightly longer wait for the first displayed interval to appear. Scripts that discard the first interval should function as before, but should be adjusted because the first interval represents valid system sampling.

Iostat command enhancements

Beginning with AIX 5L Version 5.3, the **iostat** command reports the number of physical processors consumed (*physc*), the percentage of entitled capacity consumed (*%entc*), and the processing capacity entitlement when running in a shared processor partition. These metrics will be displayed only on shared processor partitions.

In a change from previous releases (and from other Unix operating system implementations), the first interval of the command output is now meaningful and does not represent statistics collected from system boot. Internal to the command, the first interval is never displayed, and therefore there may be a slightly longer wait for the first displayed interval to appear. Scripts that discard the first interval should function as before but should be adjusted because the first interval represents valid system sampling.

Mpstat command enhancements

The **mpstat** command is the basic monitoring tool for logical CPU usage. It accepts the following flags:

- no flag – basic statistics
- **-d** – dispatcher statistics
- **-i** – interrupt statistics
- **-s** – SMT statistics
- **-a** – display all statistics (wide output)
- **-w** – format output to wide columns. The **-a** flag implicitly turns on this flag.

Lparstat command

The **lparstat** command is new in AIX 5L Version 5.3. It reports LPAR-related information and statistics.

The optional command flags for the **lparstat** command are:

- **-i** – lists details on the LPAR configuration.
- **-h** – adds summary hypervisor statistics to the default **lparstat** command output.
- **-H** – provides detailed hypervisor information, including statistics for each of the hypervisor calls.

The **lparstat** command with no options will generate a single report containing utilization statistics related to the LPAR since boot time. The **lparstat** command using the **-h** flag will add hypervisor summary statistics to the default **lparstat** output. If an interval and count are specified, the output report will repeat after the specified *interval* number of seconds for the specified *count* number of iterations.

The following is an example of this command, collecting statistics for one 5-second interval, with the hypervisor summary data highlighted:

```
# lparstat -h 5 1

System configuration: type=Dedicated mode=Capped smt=Off lcpu=4 mem=8192

%user  %sys  %wait  %idle  %hypv hcalls
-----  ----  -----  -----  ----- ------
  3.2  22.Ø    Ø.Ø   74.7    Ø.Ø      Ø
```

To show the actual status of the partition you can use the **lparstat -i** command in the AIX command line interface of the partition:

```
# lparstat -i
Node Name                                  : myserver
Partition Name                             : -
Partition Number                           : -
Type                                       : Dedicated
Mode                                       : Capped
Entitled Capacity                          : 4.ØØ
```

```
Partition Group-ID                          : -
Shared Pool ID                              : -
Online Virtual CPUs                         : 4
Maximum Virtual CPUs                        : 4
Minimum Virtual CPUs                        : 1
Online Memory                               : 8192 MB
Maximum Memory                              : 8192 MB
Minimum Memory                              : 128 MB
Variable Capacity Weight                    : -
Minimum Capacity                            : 1.00
Maximum Capacity                            : 4.00
Capacity Increment                          : 1.00
Maximum Dispatch Latency                    : -
Maximum Physical CPUs in system             : 4
Active Physical CPUs in system              : 4
Active CPUs in Pool                         : -
Unallocated Capacity                        : -
Physical CPU Percentage                     : 100.00%
```

Sar command enhancements

Beginning with AIX 5L Version 5.3, the **sar** command reports utilization metrics *physc* and *%entc*, which are related to micro partitioning and simultaneous multithreading environments. These metrics are displayed only on servers actually using micro partitioning and simultaneous multithreading environments. The *physc* metric indicates the number of physical processors consumed by the partition (in the case of system-wide utilization) or logical CPU (if the **-P** flag is specified), and *%entc* indicates the percentage of the allocated entitled capacity (in the case of system-wide utilization) or consumed entitled capacity (if the **-P** flag is specified).

When the partition runs in capped mode it cannot get more capacity than it is allocated. In uncapped mode, the partition *can* get more capacity than is actually allocated to it. This is called ‘consumed entitled capacity’. If the **-P** flag is specified and there is unused capacity, the **sar** command prints the unused capacity as a separate CPU with *cpu id U* – similar to the way that the **mpstat** command does.

Highlights of command options that have changed in AIX 5L V5.3 are as follows:

- **-u** – reports per processor or system-wide statistics. When used with the **-P** flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. Because the **-u** flag information is expressed as percentages, the system-wide information is simply the average of each individual processor's statistics except when using a shared pool of processors using micro partitioning technology, where the percentages are relative to the entitled capacity.

The following values are displayed:

- *%idle* – percentage of time the CPU or CPUs were idle with no outstanding disk I/O requests.
- *%sys* – percentage of time the CPU or CPUs spent in execution at the system (or kernel) level.
- *%usr* – percentage of time the CPU or CPUs spent in execution at the user (or application) level.
- *%wio* – percentage of time the CPUs were idle during which the system had outstanding disk/NFS I/O requests.
- *physc* – number of physical processors consumed. This is reported only if the partition is running with shared processors or simultaneous multithreading enabled (**-P** only).
- *%entc* – percentage of entitled capacity consumed (physc/ent)*100. This is reported only if the partition is running with shared processors or simultaneous multithreading enabled (**-P** only).

Topas command enhancements

If **topas** runs on a partition with shared processor allocation (micro partitioning), it displays two new values (beneath the CPU utilization):

- *physc* – number of physical processors utilized by the partition (if micro partitioning technology is utilized).

- *%entc* – percentage of entitled capacity utilized by a partition (if micro partitioning technology is utilized).

The following is the output displayed when using the new **-L** flag of the **topas** command. The **-L** flag switches the output to logical partition display:

```
Interval:    2        Machine Name : myhost     Sun Feb  6 12:17:14 2005
                                                Online Memory:   8192.0
Partition CPU Utilization                       Online Logical CPUs:  4
%user   %sys   %wait   %idle
   2     25       0      74
=======================================================================
LCPU  minpf majpf  intr   csw icsw runq lpa scalls usr sys _wt idl   pc
Cpu0      0     0   301   389  285    0 100   1231   1   2   0  97 0.00
Cpu1      0     0   410   486  284    1 100    331   0   2   0  98 0.00
Cpu2      0     0   226     2    2    1 100 186704   6  94   0   0 0.00
Cpu3      0     0   239    86   44    0 100    576   0   0   0 100 0.00
```

You can use **-L** either when invoking the **topas** command or as a toggle during the running of **topas**. In this mode, **topas** displays data similar to the **mpstat** and **lparstat** commands.

PER FILESYSTEM I/O PACING

I/O pacing is used to prevent a large number of I/O page outputs for one file from monopolizing the I/O bus. There are two tunable parameters you can set up:

- *minpout* – minimum number of outstanding I/O pages.
- *maxpout* – maximum number of outstanding I/O pages.

When the number of outstanding output pages for a particular file reaches the *maxpout* value, any other threads waiting to do additional pageouts on that segment will sleep until the number of outstanding output pages falls to the *minpage* value for that file.

In previous versions of AIX, this behaviour was tunable only system-wide using the **smit chgsys fastpath** or the **chdev -l sys0 -a maxpout=MINVAL -a minpout=MAXVAL** command. There were cases when some filesystems, like database filesystems, required values different from other filesystems', like temporary files.

In Version 5.3, the I/O pacing can be tuned on a per filesystem basis. This tuning can be done when using the **mount** command:

```
# mount -o minpout=MINVAL,maxpout=MAXVAL /fsname
```

A more user-friendly way to do this is to use SMIT or to edit the */etc/*filesystems.

The highlighted mount options of the **smit crjfs2lvstd** menu in the following sample show how to set the *minpout* and *maxpout* parameters on a per filesystem basis.

```
                          Add an Enhanced Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* LOGICAL VOLUME name                                  lvØ1           +
* MOUNT POINT                                         [/test]
  Mount AUTOMATICALLY at system restart?               no             +
  PERMISSIONS                                          read/write     +
  Mount OPTIONS
[minpout=4Ø,maxpout=6Ø] +
  Block Size (bytes)                                   4Ø96           +
  Logical Volume for Log                                              +
  Inline Log size (MBytes)                            []
#
  Extended Attribute Format                            Version 1      +
  ENABLE Quota Management?                             no             +
```

You can check that the filesystem is mounted correctly by using the **mount** command as follows:

```
# mount
  node    mounted        mounted over    vfs       date        options
--------- ------------  --------------  ------ ------------ -----------
/dev/lvØØ      /var/adm/csd     jfs    Feb Ø6 12:31 rw,log=/dev/loglvØØ
/dev/lvØ1      /test            jfs2   Feb Ø6 12:4Ø
rw,maxpout=6Ø,minpout=4Ø,log=/dev/hd8
```

It is also possible to check this using the kdb kernel debugger. First, with the **pdt *** subcommand, get the filesystem slot ID from the list of mounted filesystems, then get the entries related to this slot ID. The following is an example of this:

```
(Ø)> pdt *
               SLOT   NEXTIO           DEVICE  DMSRVAL    IOCNT <name>

vmmdseg+9ØAØØØØ ØØØØ FFFFFFFF 8ØØØØØØAØØØØØØØ2 ØØØØØØØØ ØØØØØØØØ paging
vmmdseg+9ØAØØA8 ØØØ1 FFFFFFFF 8ØØØØØØAØØØØØØ12 ØØØØØØØØ ØØØØØØØØ paging
vmmdseg+9ØA54ØØ ØØ8Ø FFFFFFFF Ø146ØC88 ØØØØØØØØ ØØØØØØØØ remote
vmmdseg+9ØA54A8 ØØ81 FFFFFFFF 8ØØØØØØAØØØØØØØ8 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA555Ø ØØ82 FFFFFFFF 8ØØØØØØAØØØØØØØ7 37Ø26ØØØ ØØØØØØØØ local
client
vmmdseg+9ØA55F8 ØØ83 FFFFFFFF 8ØØØØØØAØØØØØØØ4 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA56AØ ØØ84 FFFFFFFF 8ØØØØØØAØØØØØØØ5 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5748 ØØ85 FFFFFFFF 8ØØØØØØAØØØØØØØ6 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA57FØ ØØ86 FFFFFFFF 8ØØØØØØAØØØØØØØ9 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5898 ØØ87 FFFFFFFF 8ØØØØØØAØØØØØØØC B71F6ØØØ ØØØØØØØØ
filesystem
vmmdseg+9ØA594Ø ØØ88 FFFFFFFF 8ØØØØØØAØØØØØØØB BF1F7ØØØ ØØØØØØØØ log
vmmdseg+9ØA59E8 ØØ89 FFFFFFFF 8ØØØØØØAØØØØØØØD ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5A9Ø ØØ8A FFFFFFFF 8ØØØØØØAØØØØØØØE ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5B38 ØØ8B FFFFFFFF 8ØØØØØØAØØØØØØØF ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5BEØ ØØ8C FFFFFFFF Ø41B98C8 ØØØØØØØØ ØØØØØØØØ remote
vmmdseg+9ØA5C88 ØØ8D FFFFFFFF Ø41B98Ø8 ØØØØØØØØ ØØØØØØØØ remote
vmmdseg+9ØA5D3Ø ØØ8E FFFFFFFF Ø41B7EDØ ØØØØØØØØ ØØØØØØØØ remote
vmmdseg+9ØA5DD8 ØØ8F FFFFFFFF 8ØØØØØØAØØØØØØ1Ø ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5E8Ø ØØ9Ø FFFFFFFF 8ØØØØØØAØØØØØØ11 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5F28 ØØ91 FFFFFFFF 8ØØØØØØAØØØØØØ13 ØØØØØØØØ ØØØØØØØØ local
client
vmmdseg+9ØA5FDØ ØØ92 FFFFFFFF 8ØØØØØØAØØØØØØ14 ØØØØØØØØ ØØØØØØØØ local
client
(Ø)> pdt 92

PDT address F1ØØØ1ØØØ9ØA5FDØ entry ØØ92 of 1FFF, type: LOCAL CLIENT --
XPAGER
next pdt on i/o list  (nextio)   : FFFFFFFF
dev_t or strategy ptr (device)   : 8ØØØØØØAØØØØØØ14
last frame w/pend I/O (iotail)   : FFFFFFFFFFFFFFFF
free buf_struct list  (bufstr)   : F1ØØØ6ØØ53C6CE9Ø
total buf structs     (nbufs)    : ØØØØ
```

```
available (PAGING)     (avail)    : 0000
disk map srval         (dmsrval) : 00000000
i/o's not finished     (iocnt)    : 00000000
device wait list (devwait)        : 0000000000000000
buffer wait list (bufwait)        : 0000000000000000
logical volume lock (lock)        :@F1000100090A6028 00000000
buffer list lock     (buf_lock)  :@F1000100090A6030 00000000
flag bits             (devflags) : 84000000
max phys Xlation ent (maxphys)   : 00000020
pageout down limit    (minpout)  : 00000028
pageout up limit      (maxpout)  : 0000003C
external pager  (pager)   : 0000000001458128
```

REFERENCES

1 *AIX 5L Differences Guide, Version 5.3 Edition*, SG24-7463-00.

2 *Advanced POWER Virtualization on IBM eServer,* p5, ‘Servers: Architecture and Performance Considerations’, SG24-5768.

3 *Advanced POWER Virtualization on IBM eServer,* p5, ‘Servers Introduction and Basic Configuration’, SG24-7940.

4 *The Complete Partitioning Guide for IBM Eserver pSeries Servers*, SG24-7039.

5 *AIX 5L Workload Manager (WLM)*, SG24-5977.

Alex Polak
System Engineer
APS (Israel)

Why not share your expertise and earn money at the same time? *AIX Update* is looking for shell scripts, program code, JavaScript, etc, that experienced users of AIX have written to make their life, or the lives of their users, easier. Articles can be of any length and should be e-mailed to the editor, Trevor Eddolls, at trevore@xephon.com.

File copy script

INTRODUCTION

We sometimes need to copy files from different directories on one AIX server to another one, but don’t wish to do it in manually. In order to help achieve this, I wrote a simple script that uses a common NFS filesystem to copy files from the source server (pull) to a destination server (push), preserving directory location and all user permissions. Additionally, a back-up copy is made if there is an original file on the destination server.

To ensure file copy accuracy, all system command returns codes are validated in the script.

Note: this script can be modified to run on any flavour of Unix by simply modifying the PATH variable in the script to match your operating system.

SCRIPT

```
#!/bin/sh

# Set your PATH variable for system commands here (OK for AIX)
export PATH=/usr/bin:/usr/sbin

# Some script name and server settings
BASENAME='basename $0'                          # Script name
HOSTNAME='hostname'                             # Where am I running

#####################################################################
# Modify these parameters to match your environment
#####################################################################
FILELIST="distfiles.list"      # File with full path to files to process
#####################################################################
# DO NOT MODIFY ANYTHING BELOW HERE UNLESS YOU KNOW WHAT YOU'RE DOING
#####################################################################

# Initialize some variables
FAILCOUNT=0
```

```
# Help on script usage function
function show_help {

echo ""
echo "   The script $BASENAME is used to synchronize files from one"
echo "   server to another."
echo ""
echo "   Syntax: $BASENAME [ push | pull | ? | help ]"
echo ""
echo "   Description of parameters:"
echo ""
echo "       push : push local files to another server"
echo "       pull : pull files from the local server to another"
echo "       ?    : help"
echo "       help : this message"
echo ""
echo "   Default file name for list of files to process : $FILELIST"
echo ""
exit
}

function verify_user {

if [[ $LOGIN != "root" ]];
then
      echo "\nYou are running this script as the user $LOGIN."
      echo "It's possible that this user may not have enough"
      echo "privileges to copy certain system files."
      echo "\n\nDo you want to continue ? (Y|N) : \c"
      read ANSROOT

      case $ANSROOT in

      Y|y|yes|YES)
                  echo "Confirmation received...\n\n"
                  ;;
      N|n|no|NO)
                  echo "Copy cancelled - Good bye !\n\n"
                  exit 1
                  ;;
      *)
                  echo "Invalid response given ($ANSROOT).\n\n"
                  exit 2
                  ;;
      esac
fi
}

function check_input_file {
```

```
if [[ ! -f $FILELIST ]];
then
      echo "\nERROR: List of files to process not available:
$FILELIST\n\n"
      exit 1
fi
}

function clean_local_files {

echo "\nDo you wish to delete all local files? (Y|N) : \c"
read ANSDEL

case $ANSDEL in

Y|y|yes|YES)
                  echo "Confirmation received...\n\n"
                  rm -f =*
                  ;;
N|n|no|NO)
                  echo "Local files untouched."
                  echo "To remove local files manually, type: rm =*\n\n"
                  ;;
*)
                  echo "Invald reponse given ($ANSDEL)."
                  echo "Local files untouched."
                  echo "To remove local files manually, type: rm =*\n\n"
                  ;;
esac

}

function confirm_copy {

echo "\nWe are about to copy the local files onto another server."
echo "List of files to process can be found in the file: $FILELIST"
echo
echo "OK? (Y|N) : \c"
read ANS

case $ANS in

Y|y|yes|YES)
                  echo "Confirmation received...\n\n"
                  ;;
N|n|no|NO)
                  echo "Copy cancelled - Good bye !\n\n"
                  exit 1
                  ;;
*)
```

```
                  echo "Invalid reponse given ($ANS).\n\n"
                  exit 2
                  ;;
esac
}

function display_failed_files {

if [[ $FAILCOUNT != Ø ]];
then
      echo "\n**********************************************"
      echo "WARNING - List of files which copy was unsuccessful:\n"
      for i in ${FAILED_ARRAY[*]}
      do
            echo "\t $i"
      done
      echo "************************************************"
else
      echo "************************************************"
      echo "All files requested were copied with success"
      echo "************************************************"
fi
echo "\n\n"
}

function pull_files {

check_input_file

for f in 'cat $FILELIST'
do
      echo "Copying file $f : \c"
      if [[ ! -f $f ]];
      then
            echo "not found"
            FAILED_ARRAY[$FAILCOUNT]=$f
            FAILCOUNT='expr $FAILCOUNT + 1'
      else
            newf='echo $f|sed -e 's/\//=/g''
            cp -p $f $newf
            if [[ $? != Ø ]];
            then
                  echo "failed"
                  FAILED_ARRAY[$FAILCOUNT]=$f
                  FAILCOUNT='expr $FAILCOUNT + 1'
            else
                  echo "ok"
            fi
      fi
done
```

```
display_failed_files
}

function push_files {

check_input_file
confirm_copy

for f in 'cat $FILELIST'
do
      echo "Copying file $f : \c"
      newf='echo $f|sed -e 's/\//=/g''
      if [[ ! -f $newf ]];
      then
            echo "not found"
            FAILED_ARRAY[$FAILCOUNT]=$f
            FAILCOUNT='expr $FAILCOUNT + 1'
      else
            cp -p $f $f.orig
            if [[ $? != Ø ]];
            then
                  echo "backup failed, skipping"
                  FAILED_ARRAY[$FAILCOUNT]=$f
                  FAILCOUNT='expr $FAILCOUNT + 1'
                  continue
            fi

            cp -p $newf $f
            if [[ $? != Ø ]];
            then
                  echo "failed"
                  FAILED_ARRAY[$FAILCOUNT]=$f
                  FAILCOUNT='expr $FAILCOUNT + 1'
            else
                  echo "ok"
            fi
      fi
done

clean_local_files
display_failed_files
}

# Main
case "$1" in

pull)
      verify_user
      pull_files
      ;;
```

```
push)
      verify_user
      push_files
      ;;

help | ?)
      show_help
      ;;

*)
      echo "\n   ERROR: script parameter missing."
      show_help
      ;;
esac

exit 0
```

RUNNING THE SCRIPT

Before running this script on the source server, you will need to create a list of files to process. The format of the file is the full path to the file with no blank lines allowed.

Here is an example file:

```
/etc/passwd
/etc/group
/etc/security/passwd
/etc/security/group
/etc/security/user
/etc/sudoers
/etc/security/login.cfg
/etc/motd
/etc/security/.profile
/etc/security/.kshrc
/usr/lib/security/mkuser.sys
/etc/profile
/root/.rhosts
/usr/HTTPServer/conf/httpd.conf
/etc/security/limits
/usr/local/samba/lib/smb.conf
```

The default name for this file is *distfiles.list*. You may change it and, optionally, add a path, but don’t forget to modify the script accordingly. By default, this file is looked for in the current directory.

Now, you can pull the files from the source server onto the common NFS filesystem using the command:

```
# distfiles pull
```

Verify the output to ensure that no errors occurred. If everything ran smoothly, you may run this command on the destination server to push the files into their proper location:

```
# distfiles push
```

Again, verify that no errors occurred in the output.

You may run this command using any user, but a warning is issued if you are not the root user. This is done to warn you that permissions may prevent the script from working correctly.

Elvio Prattico
Consultant
PRATTICO Consulting (Canada)

More AIX–Solaris differences – part 2

This month we conclude our second look at our two-way guide for system administrators working in multi-platform environments with AIX and Solaris.

Adding and activating paging space

AIX:

```
# smitty mkps
```

Solaris:

- Making an empty file:

  ```
  # mkfile <size> <filename>
  ```

- Adding this empty file to swap:

  ```
  # swap -a <filename>
  ```

- Adding an entry for the swap file in */etc/vfstab* file so that it is activated automatically when the system is booted:

  ```
  /path/filename - - swap - no -
  ```

Changing filesystem attributes

AIX:

```
# chfs [ -n NodeName ] [ -m NewMountPoint ] [ -u MountGroup ] [ -A { yes
| no } ][ -p { ro | rw } ] [ -t { yes | no } ] [ -a Attribute=Value ] [
-d Attribute ] FileSystem
```

For example, to increase the size of the */test* Journaled File System, enter:

```
# chfs  -a size=+8192 /test
```

Solaris:

```
# tunefs [ -a maxcontig ]  [ -d rotdelay ]  [ -e maxbpg  ]   [-m minfree
]  [  -o [ space | time ]  ]  special | filesystem
```

Importing/exporting a volume group/disk group

AIX:

- The procedure to remove a volume group without losing data is called exporting. If you want to export a volume group you must deactivate it with **varyoffvg**:

  ```
  # varyoffvg <volume-group>
  ```

  Then:

  ```
  # exportvg <volume-group>
  ```

- When a system wants to access an existing volume group:

  ```
  # importvg -y <volume-group> <hdiskN>
  ```

Solaris:

- In order to deport a diskgroup:

  – stop all applications that are using the diskgroup and umount filesystems.

  – then stop volumes:

```
# vxvol -g <disk-group> stopall
```

  – then:

```
# vxdg deport <disk-group>
```

- To access a diskgroup:

```
# vxdg import <disk-group>
```

Moving a logical volume

AIX:

```
# migratepv -l <lv-name> <hdisk-from> <hdisk-to>
```

(You can do it online.)

Solaris:

```
# vxassist move !<disk-from> <disk-to>
```

Change logical volume settings

AIX:

```
# chlv …
```

For example to change the permission of a logical volume *test_lv*:

```
# chlv -p r test_lv
```

Solaris:

```
# vxedit set …For example to change permissions of a volume test_vol:
```

```
# vxedit -g rootdg set user=oracle test_vol
# vxedit -g rootdg set group=dba test_vol
# vxedit -g rootdg set mode=664 test_vol
```

Listing device configuration

AIX:

```
# prtconf (available in AIX 5L)
```

- Listing disks in ODM:

```
# lsdev -Cc disk
```

- Making a query in ODM for the adapter class:

  ```
  # lsdev -Cc adapter
  ```

- Get attributes of a tape device:

  ```
  # lsattr -El rmtØ
  ```

- Get VPD (Virtual Product Data) for the tape drive:

  ```
  # lscfg -vl rmtØ
  ```

  lsattr and **lscfg** can run only with configured devices.

Solaris:

- This command displays the global system configuration:

  ```
  # prtconf
  ```

- This command provides a more detailed output for the devices attached to the system, including pseudo devices, loadable modules, and some kernel parameters:

  ```
  # sysdef
  ```

- This command also displays the information about all the devices attached on the system since the last boot:

  ```
  # dmesg
  ```

- This command is used to display information about SCSI devices and to find out the devices configured on each controller:

  ```
  # cfgadm -al
  ```

Adding a SCSI device

AIX:

```
# smitty devices
```

Or with **mkdev**, for example, to configure an additional tape drive in the system:

```
# mkdev -c tape -s scsi -t scsd -p scsiØ -w 5,Ø
```

Solaris:

```
# cfgadm -x insert_device cX
```

where *X* represents the number of the controller where you attached the new device.

or:

```
# cfgadm -c configure cX
```

Back-up and restore

AIX:

- Back-up:
  - making a full back-up of the directory */test*:

    ```
    # find /test -print | backup -ivf /dev/rmtØ
    ```

  - in AIX 5L:

    ```
    # backup -Øuf /dev/rmtØ  /test
    ```

  - or:

    ```
    # smitty backfile
    # smitty backfilesys
    ```

- Restore:
  - to display the contents of the media:

    ```
    # restore -Tvf /dev/rmtØ
    ```

  - to restore individual files or directories:

    ```
    # restore -xvf /dev/rmtØ  /path/filename
    ```

  - to restore the entire filesystem:

    ```
    # restore -rqvf /dev/rmtØ
    ```

  - or:

    ```
    # smitty restfile
    # smitty restfilesys
    ```

Solaris:

- Back-up:
  - making a full back-up of the directory */test*:

    ```
    # ufsdump -Øuf /dev/rmt/Ø  /test
    ```

- Restore:
  - to display the contents of the tape:

    ```
    # ufsrestore tvf /dev/rmt/Ø
    ```

  - to interactively restore:

    ```
    # ufsrestore ivf /dev/rmt/Ø
    ```

  - to restore the entire back-up

    ```
    # ufsrestore rvf /dev/rmt/Ø
    ```

  - to restore the file specified in the command line:

    ```
    # ufsrestore xvf /dev/rmt/Ø ./user1/file1
    ```

Configuring NFS (Network File System)

AIX:

- NFS uses the following daemons:
  - rpc.lockd – processes lock requests
  - rpc.statd – cash and recovery
  - biod – sends the clients read and write requests
  - rpc.mountd – answers requests from clients
  - nfsd – starts the daemons that handle a client’s request
  - portmap – maps RPC program numbers to Internet port numbers.
- To start all NFS daemons:

  ```
  # startsrc -g nfs
  ```

- To create the exports in */etc/exports* file:

```
# exportfs -i /<dirname>
```

- To mount the remote filesystem on an NFS client:

```
# mount <nfs-server>:<pathname> <local-mount-point>
```

Solaris:

- NFS uses the following daemons:
  - mountd – handles the mount requests from clients
  - nfsd – NFS server daemon
  - statd and lockd – cash and recovery daemons for client.
- Edit the */etc/dfs/dfstab* file and add an entry like this for the filesystem to share automatically:

```
share -F nfs <filesystem>
```

- Stop and restart the NFS daemons:

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

- To mount the remote filesystem on an NFS client:

```
# mount <nfs-server>:<pathname> <local-mount-point>
```

Configuring DNS clients

AIX:

Create */etc/resolv.conf* and specify the names server and the domain name in this file.

- nameserver – <ip>
- domain – <domain>

In AIX 5L, the default name resolution order can be overridden by creating the */etc/netsvc.conf* configuration file.

For example:

```
hosts=bind,local
```

Solaris:

Create */etc/resolv.conf* and specify the names server and the domain name in this file.

- nameserver – <ip>
- domain – <domain>

Edit */etc/nsswitch.conf* and modify the line containing the hosts entry:

```
hosts: files dns
```

Disabling user account

AIX:

```
# smitty user
```

(Lock/unlock a user's account.)

Solaris:

```
# passwd -l <user-name>
```

Adding a group

AIX:

```
# mkgroup <group-name>
```

Solaris:

```
# groupadd -g <gid> <group-name>
```

Modifying an existing group

AIX:

```
# chgroup [ -R load_module ] Attribute=Value ... Group
```

For example, to add *john* and *carol* to the marketing group, which currently has only *frank* as a member:

```
# chgroup users=john,carol,frank  marketing
```

Solaris:

```
# groupmod [  -g gid  [ -o ]  ]  [ -n name ]  group
```

For example:

```
# groupmod -g 29Ø -o -admins users
```

Deleting a group

AIX:

```
# rmgroup <group>
```

Solaris:

```
# groupdel <group>
```

Checking passwords and group definitions consistency

AIX:

- Scan password file in */etc/security/passwd*:

  ```
  # pwck
  ```

- Verify that all the group members and admins exist in the user database:

  ```
  # grpck -n ALL
  ```

Solaris:

- Scan password file:

  ```
  # pwck /etc/passwd
  ```

- Verify all entries in the group file:

  ```
  # grpck
  ```

Binding or unbinding a process

AIX:

- To see which processors are available:

  ```
  # bindprocessor -q
  ```

- To bind:

  ```
  # bindprocessor <pid> <processor-number>
  ```

- To unbind:

```
# bindprocessor -u <pid>
```

Solaris:

- To bind:

```
# pbind -b <processor-id> <pid …..>
```

- To unbind:

```
#  pbind -u <pid …..>
```

CPU timeslice and process priority

AIX:

```
# schedtune -t <# of ticks>
```

(The -t flag allows you to change the timeslice, clock ticks=10msec units.)

Solaris:

```
# priocntl -e -c RT -t <# of milliseconds> command
```

Stopping/starting syslog daemon

AIX:

```
# refresh -s syslogd
```

Solaris:

```
# /etc/init.d/syslog <stop/start>
```

Packet tracing for Internet protocols

AIX:

```
# iptrace <file-to-record>
```

Solaris:

```
# snoop
```

System accounting

AIX:

```
/etc/utmp
/var/adm/wtmp
```

Solaris:

```
/var/adm/utmpx
/var/adm/wtmpx
```

Specify users who have access and no access to CRON

AIX:

```
/var/adm/cron/cron.allow
/var/adm/cron/cron.deny
```

Solaris:

```
/etc/cron.d/cron.allow
/etc/cron.d/cron.deny
```

Install OS on another disk

AIX:

Alternate disk installation (alt_disk_install) is the ability to install a complete new operating system on another disk or part of a disk while the production environment is up and running.

Solaris:

Live Upgrade is the ability of operating systems to continue to run while an administrator upgrades to the latest release of the operating system, applies patches, or does routine maintenance on the inactive or duplicate boot environment. When satisfied with the process, the administrator simply reboots the system to run the latest or updated operating environment.

Adnan Akbas
Senior System Administrator
TURKCELL (Germany)

Tape management system

Back in March 2001, my employer purchased a tape library with a bar-code reader. However, our tape back-up software did not support the reading of tape bar-codes. Our tape management procedures involved manually written adhesive tape labels and an Excel spreadsheet. I decided that this process was too susceptible to human error and other failures, so I wrote a tape management system.

It involves some flat files, several shell scripts, and two C programs. It works on AIX 4.3.3 and up (but hasn't been tested on AIX 5.3 – I have no doubt it will work because I used standard AIX ksh).

It is fairly comprehensive and supports an infinite number of systems (requires NFS for clients). It supports any standard tape drive (and multiple and different models of tape drives), and includes a somewhat detailed reporting module.

The back-up system allows for before/after images of database updates, log file purging and back-up file purging (housekeeping), and is fairly well documented.

README

```
#
# README
#
# ################
# General overview
# ################
#
# The directory this file is contained in is our "tape database".
# This is a home-grown tape management system designed by Bill Verzal
# March, 2001.  The development of this tape management system
# is in relation to the implementation of a tape library here.
#
# The library, although it came with a bar-code reader, did not
# manage tapes well, because our backup software (Sysback/6000)
# does not interface with tape library bar-code readers.
# Should we ever convert to a TSM-based or other back-up software
# (Syncsort Backup Express, CA Brightstore, etc), I'm sure the
# bar code reader could be utilized and this home-grown package
# could be decommissioned.
#
# As of October, 2004, the systems utilizing this home-grown package
# are:
#
# sap05     -  The "RIP" system
# sap80i    -  The bread and butter DB2 server for SAP/PR1
# sapbo     -  The SAP BW test system
# sapbwp    -  The SAP BW prod system
# sapcws    -  The SAP SP Control workstation
# sapent    -  The SAP Enterprise test system/Ramp up/general sandbox
# sapqam    -  The SAP DV1/QA1 system and EMC Timefinder host
#              The tape database lives on this server.
#              This is set with the variable '$dbhost' on remote servers
#              in the backup scripts.  The directory is pre-set to
#              '/var/tapesys' and is set with the '$db_dir' variable.
# saplnt    -  SAP L&T server for extension set 2
#
# The files in this directory, as of April, 2004 are:
#
#  backups - directory containing default location for back-ups from
#              backup.ksh
#  backup_pointer - text file, shows number of back-ups currently stored
#                      in the backups directory.
#
#  bin - directory containing executables and scripts
#
#  bin/afterprt.ksh - ksh script, creates a scratch report
#  bin/backup.ksh - ksh script, backup tape database to another location
#  bin/batch_tape_init.ksh - ksh script, used for initializing tapes
#  bin/clean_backups.ksh - ksh script, removes old back-up images
#  bin/daily_report.ksh - ksh script, all tapes created in the past
#                          24-hour period
#  bin/db_update.ksh - ksh script, performs updates to the database
#  bin/defcheck.ksh - ksh script, checks for defective tapes
#  bin/label_log.ksh - ksh script, logs $@ to log/tapesyslog
#  bin/lockdb.ksh - ksh script, locks the database
#  bin/printdf.ksh - ksh script, generates a "df" command with totals
#  bin/printl.ksh - ksh script, prints a report in 'landscape' mode
#  bin/printp.ksh - ksh script, prints a report in 'portrait' mode
#  bin/proc_hist_report.ksh - ksh script, called by tape_report.ksh,
#                            reports on the history of a specific volser
#  bin/proc_range.ksh - ksh script, streamlines user input for
#                            processing tape ranges
#  bin/s2d - C program executable, converts Unix Epoch seconds
#              to 'date' format. Similar to
#              'perl -we "print scalar localtime [epoch]"'
#  bin/scratch_report.ksh - ksh script, create a report of all active
#                           scratch tapes
#  bin/scratch_test.ksh - ksh script, see if a tape is scratch by
#                        reading its header
#  bin/sec - C program executable, displays Unix Epoch
#                        Similar to 'perl -we "print time"'
#  bin/sync.ksh - Reads labels and compares with database to check
#                 for inconsistencies
#  bin/tape_label.ksh - ksh script, I/O functions on tape labels
#  bin/tape_report.ksh - ksh script, generates tape reports and performs
#                              tape database actions
#  bin/unlockdb.ksh            - ksh script, unlocks the tape database
#  bin/weekend_report.ksh      - symbolic link to daily_report.ksh,
#                                creates a list of tapes created over
#                                a 3-day period, typically weekends
#  bin/weekly_report.ksh - symbolic link to daily_report.ksh, creates a
#             list of tapes created over a 5-day period, typically M-F
#  cron - directory containing suggested entries for 'root' crontab
#  cron/cron.entries - text file, crontab entries for 'root' crontab for
#                       performing scheduled actions at specific times.
#  db - directory containing database files
#
#  db/tape_history.db- text file, critical to functionality - database
#                              activity of all tapes since creation
#  db/tapes.db  - text file, critical to functionality - database
#        activity of all active tapes.  This is the main database file.
#        If this file is corrupted or lost, the entire system breaks.
#  db/tape_drive.def - text file.  Documentation only.
#                       Not used for any processing.
#             This file simply defines what systems are attached to what
#                               tape drives.
#  log - directory containing log files
#  log/db_backuplog.log - log file, status of last run from the
#                          backup.ksh script
#  log/db_cleanbackuplog.log - log file, status of last run of
#                             clean_backups.ksh
#  log/dblocklog.log - log file, traces activity of database locks
#  log/tapesyslog - log file, tracks all activity of the tape system
#  reports   - directory containing reports
#
#  reports/agereport.txt  - report file, lists first and last uses for
#                          all tapes as well as a usage count.
#  reports/alltapereport.txt - report file, complete listing of
#                             active tapes
#  reports/backupreport.txt   - report file, active tapes by backup name
#  reports/dailytapereport.txt - report file, all tapes created in the
#                               past 24-hour period
#  reports/discardreport.txt     - report file, all discarded/lost/
#                                 defective tapes
#  reports/hostreport.txt - report file, all tapes created on a
#                           specific host
#  reports/offsitereport.txt - report file, all active tapes marked
#                               as "offsite"
#  reports/onsitereport.txt - report file, all active tapes marked
#                              as "onsite"
#  reports/othertapereport.txt   - report file, all active tapes marked
#                                   as neither "offsite" nor "onsite"
#  reports/scratchreport.txt     - report file, created by
#                          scratch_report.ksh and/or tape_report.ksh
#                                   lists all active scratch tapes
#  reports/weekendreport.txt - report file, output of weekend_report.ksh
#  reports/weeklyreport.txt - report file, output of weekly_report.ksh
# ####################
# Functional overview
# ####################
# How does all this work?  Well, here we'll explain it all.
#
# First, a tape is loaded on a system.  By default, we look for
# an LTO tape drive.  Where that is not the case, we change the specs
# of what we are looking for.  Once a suitable tape drive is found,
# we read the header on the tape in the drive and begin processing it.
# ### Where applicable (read that 'SAPBWP') we change the density
# ### of the drive before usage because the pool of tapes shared by that
# ### and other hosts includes some older style tape drives, so we need
# ### to be sure that the tapes are compatible across all the systems.
# The above 4 lines of comment are no longer valid because the tape
# drive configurations have been changed.
#
# Once we have a drive and a tape label has been read, we check to see
# if the tape is flagged as a defective tape.  Once it is confirmed to
# be a 'good' tape (via defcheck.ksh), we check to see if the tape is
# scratch and also if the information on the tape matches with what is
# in the database (tapes.db).  The script 'scratch_test.ksh' performs
# both of these tasks.  Once this is completed and the tape is confirmed
# both 'scratch' and 'in sync', we will allow it to be used.
#
# Based on parameters that are passed to 'tape_label.ksh', we change the
# tape label at this point.  This involves a 'rewind', a re-write of the
# tape label, another rewind, and a 'mt -f <tapedrive>.1 fsf 1' which
# positions the tape immediately after the label.  The label consists
# of only 3 fields:
#
# Volser
# Creation time in 'epoch' format (seconds past 'Wed Dec 31 18:00:00
# 1969') Retention in seconds
#
# The label is in the format of 'volser:creation:retention'.
#
# All other tape information is kept in the database 'tapes.db'.  These
# three fields on the tape are all that are needed to perform the
# processing we need to keep our tapes managed correctly.
```

```
#
# Once the tape is positioned after the label, as mentioned above, the
# back-ups is run.  A 'no-rewind' back-up device name must be specified
# as the back-up device (/dev/rmtX.1) where 'X' is the tape drive ID.
# As well, if the back-up solution performs a back-up before use, this
# feature must be disabled.  With Sysback, this is done via the "-n"
# option to the back-up commands.  So, the back-up commands are executed
# with both a "-n" option and a tape drive argument of /dev/rmtx.1.
#
# From there, the 'result code' of the backup command is captured and
# saved.  The tape is rewound.  If the result code is good, the tape is
# re-read and the database is updated by the script called
#  'db_update.ksh'.
# This script requires some additional fields, such as 'backup name'.
# This script runs another script called 'lockdb.ksh', which creates a
# 'lock file' which prevents another system from performing an update to
# the database while this one is running.  That would result in possible
# database corruption or loss of information.  Just prior to the
# database update being committed, a script called "backup.ksh" is run,
# which creates a 'before image' of the database, which give us a back-
# out point, in the event for some reason the database update corrupts
# something.  All binaries, scripts, logs and database files are in each
# back-up.  The previous back-ups are excluded from the back-up files.
#
# After the database has been successfully updated, the 'unlockdb.ksh'
# script is run, which removes the lock file.  Any database updates
# waiting on other systems can now single-thread updates into the
# database.
#
# This completes the functional overview of the system for now.  Please
# leave a message at the beep.
```

The following files can be found in the *bin* directory.

AFTERPRT.KSH

```
scratch_report() {
clear
now='$sec'
now_date='$s2d $now'
tmp="/tmp/report.$$"
>$tmp
echo "Scratch tapes as of: $now_date\n" >> $tmp
printf $format "Volser" "Creation time" "Expiration time" "Host" "LOC"
"Contents" "   Usage" >> $tmp
printf $format "--------" "------------" "--------------" "----" "--" "-
-------" "   ----" >> $tmp
found=0
echo "Processing \c"
for tape_rec in 'cat $tapes|sort -n -k3,6' ; do
      volser='echo $tape_rec|cut -f1 -d":"'
      creation='echo $tape_rec|cut -f2 -d":"'
      lifetime='echo $tape_rec|cut -f3 -d":"'
      host='echo $tape_rec|cut -f4 -d":"'
      backup='echo $tape_rec|cut -f5 -d":"'
      location='echo $tape_rec|cut -f6 -d":"'
      creation_time='$s2d $creation'
      let foo=$creation+$lifetime
      expiration_time='$s2d $foo'
      passes='grep $volser $tape_history|wc -l'
      if [ "$location" -eq "Ø" ] ; then
            loc="ON"
      elif [ "$location" -eq "1" ] ; then
            loc="OFF"
      else
            loc="UNK"
      fi
      if [ "$foo" -le "$now" ] ; then
            printf $format "$volser" "$creation_time" "$expiration_time"
"$host" "$loc" "$backup" "$passes" >> $tmp
            found='expr $found + 1'
      fi
      echo ".\c"
done
      clear
      echo "\nRecords found: $found\n" >>$tmp
      cat $tmp
      /home/OPERATOR/bin/printl.ksh versap5 $tmp
      sleep 3
      cp $tmp $home/reports/scratchreport.txt
      rm $tmp
            }
home="/var/tapesys"
tape_history="$home/db/tape_history.db"
tapes="$home/db/tapes.db"
sec="/$home/bin/sec"
s2d="/$home/bin/s2d"
format="%-13s%-29.26s%-29.26s%-1Ø.1Øs%-5s%-4Ø.38s%-5s\n"
scratch_report
```

BACKUP.KSH

```
#!/bin/ksh
#
# Program to back up the tape database
#
# Written 6/4/2ØØ1 Bill V.
#
```

```
home="/var/tapesys"
log="$home/log/db_backuplog.log"
tapes="$home/db/tapes.db"
tape_history="$home/db/tape_history.db"
label_log="$home/bin/label_log.ksh"
#
# Check freespace in database directory
#
df -k $home|tail -1|read a b free d e
$label_log "Database freespace: $free kB"
if [ "$free" -lt "10240" ] ; then
   $label_log "Error allocating freespace for backups"
   echo "Error backing up tape database - directory full"
   echo "Call AIX Support.  Process will pause here."
echo "\n`tput smso`DO NOT Press [ENTER] until told to do so.`tput rmso`"
   read foo
   echo "Safety check - Press [ENTER] again."
   read anotherfoo
fi
$home/bin/lockdb.ksh "DB Backup"
$label_log "Database lock created for DB Backup"
exec 1>$log 2>&1
   date
   nbkups='ls -la $home/backups|wc -l|awk {'print $1'}'
   $label_log "Number of backups currently online: $nbkups"
   echo "Number of backups currently online: $nbkups"
   cd $home
   echo "\nBacking up tape database\n"
   pwd
   echo " "
   yymdhms='date +%Y%m%d-%H%M%S'
   backup="backup-$yymdhms.tar"
   files='ls -tr $home|grep -v backups'
   ls -l $files
   echo " "
   echo "\n\nBackup file is $home/backups/$backup\n"
   tar -cvf /tmp/$backup $files
   $label_log "Backup created"
   mv /tmp/$backup $home/backups
   cp $home/backups/$backup /home/tapesysbackup.tar
   echo "\nBackup copy is stored in /home/tapesysbackup.tar"
   # ### cat $log | mail -s "Tape System Backup Log" root
   compress $home/backups/$backup
   $label_log "Backup compressed"
$home/bin/unlockdb.ksh "DB Backup"
$label_log "Database lock released for DB Backup"
foo='ls $home/backups/|wc -l|awk {'print $1'}'
echo "Backups online: $foo" > $home/backup_pointer
```

BATCH_TAPE_INIT.KSH

```
#!/bin/ksh
if [ "$#" -eq "Ø" ] ; then
   echo "You must specify a filename"
   exit 1
fi
> $1
echo "Enter tape numbers separated by spaces.  Enter just the number"
echo "portion of the tape.\n"
echo "If you have a sequence of tapes, enter the first tape number"
echo "then enter a comma and the number of tapes."
read ftn
cma='echo "$ftn"|grep ','
if [ "$?" -eq "Ø" ] ; then
   cma=1
else
   cma=Ø
fi
echo "Will you be initializing LTO or DLT tapes?"
echo "   - Enter  \"L\" for LTO"
echo "   - Enter  \"D\" for DLT"
typeset -l t
read t
if [ "$t" = "l" -o "$t" = "d" ] ; then
   :
else
   echo "Invalid option entered for tape type"
   echo "Only \"L\" or \"D\" are allowed inputs"
   exit 1
fi
if [ "$cma" -eq "1" ] ; then
   cnt=1
   ftn1='echo "$ftn"|cut -f1 -d","'
   ftn2='echo "$ftn"|cut -f2 -d","'
   ftn="$ftn1"
   while [ "$cnt" -lt "$ftn2" ] ; do
      ftn1='expr $ftn1 + 1'
      cnt='expr $cnt + 1'
      ftn="${ftn} ${ftn1}"
   done
fi
A="A"
B="DLT"
zero="ØØØØØØØ"
cut=Ø
cut1=6
cut2=4
cut=$cut1
x=$ftn
```

```
if [ "$t" = "d" ] ; then
   A=$B
   cut=$cut2
fi
for qtape in $ftn ; do
   b="${zero}${qtape}"
   c='echo "$b"|wc -c|awk {'print $1'}'
   d='expr $c - 1'
   e='expr $d - $cut'
   f='echo "$b"|cut -c ${e}-${d}'
   g="${A}${f}"
   echo $g >> $1
done
```

CLEAN_BACKUPS.KSH

```
#!/bin/ksh
#
# Program for purging back-ups of the tape database
#
# Written 9/19/2001 Bill V.
#
home="/var/tapesys"
log="$home/log/db_cleanbackuplog.log"
otherlogfiles="$home/log/dblocklog.log  $home/log/tapesyslog"
logfilelevel=80  # This will be divided by 100 to get a percentage.
tmp="/tmp.$$"
tapes="$home/db/tapes.db"
tape_history="$home/db/tape_history.db"
label_log="$home/bin/label_log.ksh"
default=750        # Nbr of backups to save - backups, not days
cleanlogs=0
cleanbkups=0
logmsg=
if [ "$#" -eq "0" ] ; then
   logmsg="Cleaning backup files and log files - default settings"
   cleanlogs=1
   cleanbkups=1
   nbr="$default"
else
   while [ "$#" -gt "0" ] ; do
      if [ "$1" = "-all" ] ; then
         logmsg="Cleaning backup files and log files ${logmsg}"
         cleanlogs=1
         cleanbkups=1
         nbr="$default"
      elif [ "$1" = "-logs" ] ; then
         logmsg="Cleaning backup log files only ${logmsg}"
         cleanlogs=1
      elif [ "$1" = "-backups" ] ; then
         logmsg="Cleaning backups only ${logmsg}"
         cleanbkups=1
      elif [ "$1" = "-num" ] ; then
         if [ "$cleanlogs" -eq Ø -a "$cleanbkups" -eq Ø ] ; then
            cleanlogs=1
            cleanbkups=1
         fi
         shift
         nbr="$1"
         logmsg="Setting log value to $nbr ${logmsg}"
      fi
      shift
   done
fi
if [ "$cleanlogs" -eq "Ø" -a "$cleanbkups" -eq "Ø" ] ; then
   echo "Unable to clean logs - cannot determine config settings"
   exit 1
fi
$home/bin/lockdb.ksh "Clean DB Backups"
$label_log "Database locked for DB clean"
$label_log "$logmsg"
exec 1>$log 2>&1
   date
   echo "$logmsg\n"
   cd $home
   if [ "$cleanbkups" -eq "1" ] ; then
      nbkups='ls -la $home/backups|wc -l|awk {'print $1'}'
      $label_log "Number of backups before cleaning: $nbkups"
      echo "Number of backups before cleaning: $nbkups"
      echo "\nPurging backup tape database and cleaning logs\n"
      echo "# = $nbr"
      cd backups
      ls -tr |grep "backup-"|tail -$nbr > $tmp
      for file in 'ls -tr' ; do
         grep $file $tmp 1>/dev/null 2>&1
         if [ "$?" -eq "1" ] ; then
            echo "Purging $file"
            rm $file
            $label_log "Removing file $file"
         else
            echo "Not purging $file"
         fi
      done
      nbkups='ls -la $home/backups|wc -l|awk {'print $1'}'
      $label_log "Number of backups after cleaning: $nbkups"
      echo "\nNumber of backups after cleaning: $nbkups"
      rm $tmp
   fi
   cd $home
   if [ "$cleanlogs" -eq "1" ] ; then
      logpct=".${logfilelevel}"
      echo "Log factor = $logpct"
      $label_log "Log factor = $logpct"
      for logfile in $otherlogfiles ; do
         echo "Logfile: $logfile"
         echo "Before: `ls -l $logfile`"
         lines='wc -l $logfile|awk {'print $1'}'
         cp $logfile $logfile.save
         loglines='echo "$lines*${logpct}"|bc|cut -f1 -d"."'
         tail -${loglines} $logfile > $logfile.tmp
         mv $logfile.tmp $logfile
         echo " After: `ls -l $logfile`\n"
      done
   fi
   cat $log | mail -s "Backup Log Purge" root
   $home/bin/unlockdb.ksh "Clean DB Backups"
   $label_log "Database unlocked for DB clean"
```

DAILY_REPORT.KSH

```
#!/usr/bin/ksh
#
myname="'basename $Ø'"
home="/var/tapesys"
base=1
Report="Daily"
TAPE_REPORT_FILE="dailytapereport.txt"
label_log="$home/bin/label_log.ksh"

if [ "$myname" = "weekly_report.ksh" ] ; then
   $label_log "Running weekly report"
   Report="Weekly"
   base="7"
   export TAPE_REPORT_TITLE="Weekly Tape Movement Report"
   export TAPE_REPORT_FILE="weeklyreport.txt"
elif [ "$myname" = "weekend_report.ksh" ] ; then
   $label_log "Running weekend report"
   Report="Weekend"
   base="2.75"
   export TAPE_REPORT_TITLE="Weekend Tape Movement Report"
   export TAPE_REPORT_FILE="weekendreport.txt"
elif [ "$myname" = "daily_report.ksh" ] ; then
   $label_log "Running daily movement report"
fi

echo "$base" | $home/bin/tape_report.ksh foo daily 1>/dev/null 2>&1
DT='date +%D-%T'
```

```
cat $home/reports/$TAPE_REPORT_FILE|mail -s "$Report Tape Movement
Report for $DT" ops,root
```

Editor's note: this article will be continued next month.

Bill Verzal
Project Leader
Komatsu America (USA)

AIX news

Atempo has announced Version 4.0 of Time Navigator, its data protection software that provides unified data management across all tiers of storage.

Time Navigator 4.0 integrates the latest snapshot management, replication, and advanced disk-to-disk-to-tape (D2D2T) virtual library technologies into its back-up and restore architecture. With Time Navigator, storage managers are able to manage data protection between all storage layers.

As well as AIX, Time Navigator supports all major Unix platforms, Mac OS X, Microsoft Windows, and Linux platforms.

For further information contact:
URL: www.atempo.com/products/overview.php.

* * *

Appgen Software Solutions has announced that Version 6.2 of Appgen now runs under AIX.

The Appgen product line consists of the Appgen 4GL Development System, Appgen Custom Suite applications, the MyBooks Professional small business accounting and finance software, and the new Executive Dashboard.

Appgen's applications are cross-platform, natively supporting Microsoft Windows, various distributions of Linux (including Red Hat, Novell and Linspire), and Mac OSX. With the updated support for the AIX platform, users are able to co-mingle AIX with other OS installations in their infrastructure and still ensure application and data compatibility.

For further information contact:
URL: www.appgen.com/.

* * *

Jacada has announced Version 2.0 of Jacada Fusion, its software for producing process-centric composite applications. The latest release of Jacada Fusion offers enhancements to all core components of the platform including infrastructure, system management, and its non-invasive, service-enabling tools – Jacada HostFuse, Jacada WebFuse, and Jacada WinFuse.

Jacada Fusion enables organizations to change and improve business processes without rewriting or replacing any existing applications, the company claims.

In the latest release, customers have a new option to run the Jacada HostFuse engine as a JCA resource adapter natively under J2EE application servers, on AIX, Windows, Solaris, HP-UX, and zSeries, and iSeries platforms.

For further information contact:
URL: www.jacada.com/Products/Jacada_fusion/welcome.htm.

* * *

BindView has announced Version 8.00 of Password Self Service, which provides an integral element for global identity management strategies, helping enterprises balance password security and high-level productivity. It provides 24x7 Web-enabled password reset requests across multiple operating systems, databases, and applications.

Users have password management capabilities for Active Directory, Windows NT, Novell eDirectory, Microsoft SQL Server, and many Unix versions including AIX, SUN iPlanet, HP, SUSE Linux, and Red Hat Linux.

For further information contact:
URL: www.bindview.com/Products/DirAdminMig/PasswordManagement/.