68

AIX

June 2001

In this issue

update

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £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 1995 issue, are available separately to subscribers for £16.00 ($23.00) 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/aixupdate.html; you will need to supply a word from the printed issue.

Editors

Trevor Eddolls and Richard Watson

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Printed in England.

Report formatting script

When creating reports containing columns of data, it is nice to align the columns. To do this you can use the **printf** command or alternatively pipe the data into **awk**, and use **awk**'s own **printf** command. The problem with both these solutions is that often the data you are reporting on changes and the new data throws the columns out of line. The script *autoFormat* gets round this problem by calculating the required width of each column at run time. Hence by piping your data through *autoFormat* your columns will alway be correctly aligned.

This is the script:

```
#!/usr/bin/ksh
#
# Script: autoFormat
# Author: Roger Wickings
# Aim:    Format a report

awk="/usr/bin/awk"
basename="/usr/bin/basename"
cat="/usr/bin/cat"
rm="/usr/bin/rm"

TMPDIR="/tmp"

# Start of main processing

SCRIPT=`$basename $0 `
PARMS="$*"

TMPFILE="$TMPDIR/$SCRIPT.$$"

echo "$PARMS " > $TMPFILE

$cat $TMPFILE - |
$awk '
      BEGIN   {
                maxsub = 0
                maxfld = 0
                size[maxfld] = 0
              }
      NR == 1 {
                for ( fldsub = 1 ; fldsub <= NF ; ++fldsub )
                {
```

```
            if ( $fldsub == "l" )
            {
              align[fldsub] = "-"
            }
          }
        }
NR != 1 {
          if ( NF > maxfld )
          {
            while ( maxfld < NF )
            {
              maxfld++
              size[maxfld] = Ø
            }
          }

          for ( fldsub = 1 ; fldsub <= maxfld ; ++fldsub )
          {
            if ( $fldsub == "" )
            {
              break
            }
            field[maxsub] = $fldsub
            maxsub++
            if ( length($fldsub) > size[fldsub] )
            {
              size[fldsub] = length($fldsub)
            }
          }
          field[maxsub] = ""
          maxsub++
        }
END     {
          colsub = 1
          for ( fldsub = Ø ; fldsub < maxsub ; ++fldsub )
          {
            if ( field[fldsub] == "" )
            {
              printf "\n"
              colsub = 1
            }
            else
            {
        printf "%" align[colsub] size[colsub] "s " , field[fldsub]
              colsub++
            }
          }
        }
'
```

```
$rm $TMPFILE

exit 0
```

By default, script *autoFormat* right-justifies data in each column. For example *df* | *autoFormat* produces the following output:

```
          Filesystem 512-blocks    Free %Used  Iused %Iused             Mounted on
            /dev/hd4     131072   52608   60%   3212    10%                   /
            /dev/hd2    3538944  209080   95%  57950    14%                /usr
         /dev/hd9var     131072   73920   44%   2198    14%                /var
            /dev/hd3     131072  121064    8%    315     2%                /tmp
            /dev/hd1     720896  227736   69%   7480     9%               /home
     /dev/tftpbootlv     327680   37760   89%    169     1%           /tftpboot
       /dev/spdatalv   34832384 2414176   94%  71531     2%             /spdata
           /dev/lv00      65536   61968    6%     28     1%        /var/adm/csd
          /dev/ftilv     196608  143800   27%   2128     9%                /fti
          /dev/optlv     163840   25792   85%    289     2%                /opt
      /dev/refdatalv      32768   26616   19%    302     8%        /fti_refdata
          /dev/xmrec     163840   29984   82%     41     1%          /usr/xmrec
      /dev/historylv      32768   21784   34%   1251    31%       /history/user
     /dev/proddatalv      32768   27032   18%    296     8%          /prod_data
     /dev/tivoli32lv     294912  117288   61%   1361     4%    /opt/Tivoli_v3.2
        /dev/locallv      65536   58888   11%    319     4%          /usr/local
   /dev/ctrlmfixeslv     327680  196896   40%     86     1% /opt/controlm_fixes
          /dev/osslv      65536   53704   19%    372     5%                /oss
        /dev/rogerlv      98304   62848   37%   1512    13%              /roger
/home/spcws/operator     720896  227728   69%   7480     9%         /u/operator
    /dev/dolexportlv     655360  116768   83%     17     1%          /dolexport
```

In order to allow columns to be either right- or left-justified, *autoFormat* accepts a parameter list of ‘r’s and ‘l’s, each corresponding to a column and signifying the column’s justification.

For example *df* | *autoFormat l r r r r r l* produces:

```
Filesystem           512-blocks    Free %Used  Iused %Iused Mounted             on
/dev/hd4                 131072   52608   60%   3212    10% /
/dev/hd2                3538944  209080   95%  57950    14% /usr
/dev/hd9var              131072   73312   45%   2200    14% /var
/dev/hd3                 131072  121056    8%    315     2% /tmp
/dev/hd1                 720896  227728   69%   7480     9% /home
/dev/tftpbootlv          327680   37760   89%    169     1% /tftpboot
/dev/spdatalv          34832384 2414176   94%  71531     2% /spdata
/dev/lv00                65536   61968    6%     28     1% /var/adm/csd
/dev/ftilv               196608  143792   27%   2128     9% /fti
/dev/optlv               163840   25792   85%    289     2% /opt
/dev/refdatalv            32768   26616   19%    302     8% /fti_refdata
/dev/xmrec               163840   29984   82%     41     1% /usr/xmrec
/dev/historylv            32768   21784   34%   1251    31% /history/user
```

```
/dev/proddatalv          32768   27032   18%    296     8% /prod_data
/dev/tivoli32lv         294912  117288   61%   1361     4% /opt/Tivoli_v3.2
/dev/locallv             65536   58888   11%    319     4% /usr/local
/dev/ctrlmfixeslv       327680  196896   40%     86     1% /opt/controlm_fixes
/dev/osslv               65536   53704   19%    372     5% /oss
/dev/rogerlv             98304   62848   37%   1512    13% /roger
/home/spcws/operator    720896  227728   69%   7480     9% /u/operator
/dev/dolexportlv        655360  116768   83%     17     1% /dolexport
```

To get round the problem of the split 'Mounted on' heading I would code:

```
df | sed "s,ed on,ed@on," | autoFormat l r r r r r l | tr "@" " "
```

The **sed** command joining 'Mounted on' into a single string 'Mounted@on', with the tr translating the '@' character back to a space after the data has been formatted.

Roger Wickings
Systems Programmer
FT Interactive Data (UK)

Ensuring consistent mirrored logical volume data

Although implementing mirroring in AIX is simple, understanding and ensuring consistency of the different copies of the data requires a much closer examination of the structure of the Logical Volume Manager (LVM). Each physical volume, or disk, is made up of physical partitions, each equal in size, from 1 to 256 megabytes (default is typically 4 or 8MB). A volume group (containing multiple physical volumes) is divided into logical volumes, and each logical volume is made up of logical partitions. When no mirroring is implemented, a single logical partition points to a single physical partition. When mirroring is used, a single logical partition points to either two (single-mirroring) or three (double-mirroring) physical partitions.

Each physical partition making up a logical partition contains a copy of the same data. When a logical partition is changed, all physical partitions making up the logical partition are changed in the same manner, thus maintaining consistent copies of the data. It is of the utmost importance that the contents of the physical partitions always remain consistent. If not, it may be possible to read the same data from a filesystem twice and receive different results each time.

LVM ensures the consistency of mirrored data through three resources – Volume Group Status Area, Quorums, and Mirror-Write-Consistency Cache.

VOLUME GROUP STATUS AREA

Every physical volume contains a copy of the Volume Group Descriptor Area (VGDA), which maintains information about the volume group, logical volumes, and physical volumes in the volume group. It also contains the Volume Group Status Area (VGSA). The VGSA holds the status of the physical partitions in the volume group. This status indicates whether the partition is *syncd*, meaning that it is a consistent copy of the data, or *stale*, meaning that the partition contains a copy of a mirrored logical partition that is not current.

Each time a volume group is varied on, the VGDA is read from each disk, and, if not identical, the most recent VGDA is copied to all other disks.

If any disk containing a copy of a mirrored logical volume becomes unavailable, the VGSA on all remaining disks will be updated each time a logical partition is changed to indicate that the physical partition on the missing disk is now stale, adding back the missing physical volume when the volume group is varied on again. The stale physical partitions can be updated, or synced, to again become consistent with the most recent copies.

Unfortunately, this process is not foolproof in maintaining consistent copies. If, for instance, there are two disks in the volume group, and one becomes unavailable, the VGSA information on the remaining disk will be updated to reflect stale partitions. If the volume group is later varied on again, so the missing disk is made available and the previously-available disk is now missing, LVM no longer knows about the stale partitions on the available disk. You can see how this could become disastrous since the VGSA on each of the disks will reflect stale partitions on the other. How will LVM know which one to use in resyncing the mirrored data? This is where quorums come into play.

QUORUMS

LVM ensures that the most recent copy of the VGDA is always available by ensuring that a *quorum*, or majority, of VGDAs in the volume group are always available. If the loss of a disk in the volume group results in less than a quorum of VGDAs in the volume group, the entire volume group will be varied off. Note that, with the *rootvg* volume group, this may result in a system crash!

Because the loss of quorum is possible should either disk fail in a 2-disk volume group, LVM places a third VGDA on the first disk in a 2-disk volume group. If the first disk in the volume is lost, the VG is varied off since two of the three VGDAs are unavailable. If, however, the second disk in the volume group is lost, the VG remains available

since there are two of the three VGDAs remaining. This may not sound like a terrific solution, but at least it reduces the likelihood of the loss of a quorum by 50%.

Using quorums seems like a high price to pay just to prevent the unlikely circumstance described above. Therefore, most customers choose to turn off quorum checking. This can be done with the command *chvg –qn VGname*. If quorum checking is not used, then any time a volume group is varied on without the most recent VGDA copy, you must manually resync all partitions, whether they are stale or not, using a command *syncvg –f VGname*. This will make all copies consistent, although not necessarily using the most up-to-date copy of the data.

Note: quorums are useful only in ensuring consistent copies of mirrored data. However, quorum checking is always made active by default, even when no mirroring is used. Therefore, you should always protect against loss of quorum by turning off quorum checking for volume groups that contain no mirrored logical volumes.

MIRROR-WRITE-CONSISTENCY

It is important to understand that not all copies of the data are updated at exactly the same time. Writes to a disk with outstanding I/O requests may take longer than those without. Because of this, there are intervals in which the copies of a logical partition will be inconsistent. A volume group or system failure that occurs during this period of time will result in inconsistent copies.

LVM ensures consistency of copies after a volume group failure by tracking when changes are being made to a logical partition. This is achieved by using the Mirror-Write-Consistency Cache (MWC).

There is a considerable performance cost to using MWC in most cases. MWC is maintained on a sector of each disk in the volume group. A bit is updated in the MWC to indicate when a mirror-write to a logical partition has begun and again when the write has completed. Should the system or volume group become unavailable while in the process of writing the data, the flag will indicate that the physical

partitions making up the logical partition may not be identical. This means there are actually three writes to a disk each time a partition is updated!

For disks containing mirrored logical volume copies with a low number of write requests, the use of MWC may have little or no impact on performance. However, if there is a high amount of I/O to a disk, updating the MWC for each mirror-write may have a considerable impact. This impact can be lessened by placing mirrored logical volumes with the highest number of writes nearest the MWC sector of the disk, or the outer edge.

Mirror-write-consistency can also be turned off if the impact on performance outweighs the benefit of maintaining consistent copies. This can be done for each logical volume by using the command *chlv –wn LVname*. If it is turned off, there is no way to know what partitions are being updated, so it becomes imperative that the logical volume be manually resynced any time a volume group failure occurs. To force a resync of a logical volume, use the command *syncvg –f -l LVname.*

In AIX 5L, there is a new MWC policy that may be applied to help ensure consistency of mirrors without updating the MWC area during each write. This is called *passive MWC.* Passive MWC tracks only when a logical volume is opened. If the system halts without properly closing a logical volume (ie unmounting a filesystem), a forced resync of the entire logical volume is automatically performed after the volume group is varied on again. Passive MWC may only be implemented for logical volumes in Big Volume Groups, and may be set with the command *chlv –wp LVname*.

Anthony Johnson
Software Developer
Storix Software (USA)

A back-up script with CGI code to view results

This back-up script uses **cpio** to back-up and verify individual filesystems, and then creates a results file that the given CGI script can use to view the results on an intranet, for example. If you do not have access to CGI or an intranet, then the results file can be printed instead.

Ideally you will need to have tapes labelled 02 – 31 and Jan – Dec. The results of the back-ups will need to be saved into a 'log' directory that contains sub-directories named 02 – 31 and Jan – Dec (these sub-directories are created if they do not already exist). On the first of each month, the monthly tape should be loaded. This will allow you to keep one month's worth of back-up results on-line and to view the monthly back-ups for a whole year.

The back-up performs the following tasks:

- A header is written to the tape in **dd** format and consists of the server name, tape number or month (ie day of month or Jan-Dec), the date and time, and the output of the **df** command so it is possible to see which filesystems were backed up on that particular tape. This header is also written to a file called **header** in the specified log directory, so you can always see what filesystems were backed up.
- A list of all mounted filesystems is obtained from a **df**. Each of these filesystems is then backed up sequentially, using **cpio**, to the non-rewind tape device. The list of files backed up in each filesystem is written to a file called **backup_<*filesystem*>** (eg the files backed up in the filesystem **/home/accounts/pinney** will be held in a file called **backup__home_accounts_pinney**. You will notice that an underscore replaces the '/' in the filesystem name).
- After all the filesystems have been backed up, the tape is rewound and each of the filesystem back-ups is verified. The list of files on the tape is written to a file called **verify_<*filesystem*>**. The naming conventions are the same as for the back-ups.
- For each filesystem, the **backup_<*filesystem*>** and

verify_<*filesystem*> files are compared (using **diff**) and any differences are written to a file called **errors__<*filesystem*>**.

- A file called **results** (in the log files directory) is then created and contains the following fields:
  - Filesystem name.
  - OK or FAILED.
  - Start time of back-up.
  - End time of back-up.
  - Number of files backed up.
  - Size of filesystem in either KB, MB, or GB.
  - Name of error log file.
- This file can then be either printed or viewed on an intranet page. If you view the results from the HTML script, yesterday's back-up results will be shown, by default, but you can select any day (2-31) from the pull-down list, or any month (Jan-Dec) to show the monthly back-up.
- The script is written to handle the more complicated case of where the Web server (hosting the CGI scripts) is separate from the server(s) that you are backing up. If you have the CGI script on the same server as the one you are backing up then read the notes in the back-up script to see what needs changing (more details later).

CHECKS BUILT INTO THE BACK-UP SCRIPT

Before the back-up script runs each day, checks are made on the tape that is loaded into the drive:

- It makes sure that the tape (if any) in the drive can be rewound. Reasons for failure are: no tapes in drive; drive not on-line; drive busy doing another back-up/restore; or a corrupt tape.
- It reads the header on the tape to check that the tape is valid. The first thing contained in the header on each tape is the server name

and the day of the month – eg **server1 10** for the Server1 tape used on the 10th of every month.

E-mails will be sent to a specified mail recipient list as soon as a problem is detected with the back-up.

A typical e-mail would be as follows:

```
------------------------------------------------------------------------
NOTE: This e-mail only applies to the Operator on shift on Mon 12/Ø3/Ø1
at 1Ø:42. If that isn't you then please ignore it.
------------------------------------------------------------------------


IMPORTANT:      <server> - back-up tape not loaded correctly

There is a problem rewinding the tape so the back-up CANNOT proceed.
Ensure the tape is correctly loaded and try running the back-up again.

You can do this as follows:

Open a terminal session to <server> and login as 'root':-

Type the following:

      /usr/local/backups/rerun_backup

If you do not get this e-mail again then the back-up will be running.
------------------------------------------------------------------------
```

In this case the tape could not be rewound so it is probably not loaded correctly. A similar e-mail will be sent if the tape cannot be written to.

You must re-run the back-up as soon as you have fixed the problem, by running a given script (**/usr/local/backups/rerun_backup** in this example, but this is configurable).

Additional checks have been added to generate an alert if the number of filesystem back-up failures is either equal to the expected number of back-ups or more than 1/3 of the number.

For example, if the server had 19 filesystem back-ups per night, you will be alerted if six or more back-ups fail.

An e-mail will be sent to the specified mail recipient list to alert them of these failures. Instructions are contained in the e-mail detailing

what checks can be made and how to re-run the back-up by running the script **/usr/local/backups/rerun_backup**.

HOW DO YOU KNOW IF THE BACK-UPS WORKED?

Each day an e-mail will be sent to a specified mail recipient list if any of the back-ups have FAILED. The e-mail contains a link to the intranet page so you can go and check the failures.

No e-mail means that the back-ups were successful.

The usual cause of back-up failures is temporary files. This happens when the **find** command lists a file that is subsequently removed before **cpio** can back it up to tape. If all the back-ups fail it is usually caused by a problem with the tape – eg faulty, not loaded, or write-protected.

The results file, if printed, looks like this:

```
/home            OK    23:30:09    23:30:12      76    320KB
/opt             OK    23:31:05    23:37:29    6370    324MB
/tmp         FAILED    17:52:04    17:52:33      20      3MB
/usr/local/log/23/errors__stand
/usr/local/log   OK    23:37:44    23:38:18    1205     50MB
/usr             OK    23:38:18    23:48:00   19907    531MB
/var             OK    23:48:06    00:02:06   22241    734MB
/stand           OK    00:02:08    00:02:34      20     19MB
/                OK    00:02:35    00:03:22    1683     47MB
```

WHAT THE HTML SCRIPT OFFERS

The HTML script provides an easy-to-use view of the back-up results – see Figure 1.

By default, yesterday's back-ups are shown because it is assumed that the back-up was run overnight and you are checking the results the next day. You can, however, select any day or month from the pull-down list.

While the back-up is running, the 'Status' column will indicate where the back-up is currently at.

You will see the following states (these may be GIFs or plain text):

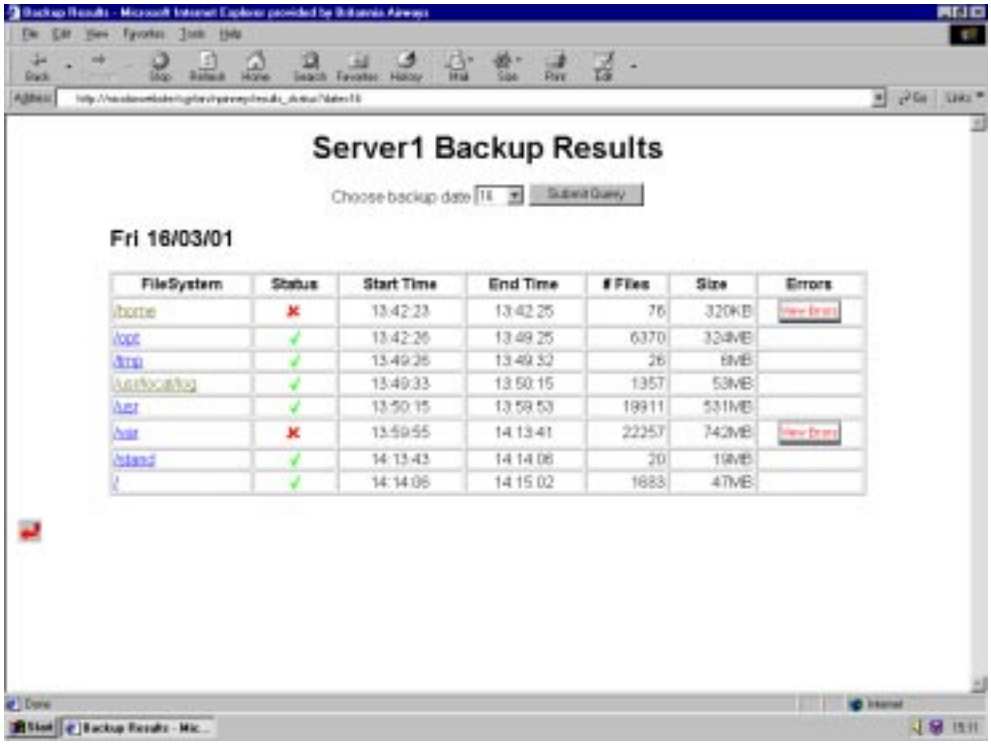

Figure 1: Back-up results

1 **Writing** – filesystem is being backed up.

2 **Written** – filesystem has been backed up.

3 **Verifying** – the back-up of the filesystem is verifying.

4 **Verified** – the back-up of the filesystem has been verified.

5 A tick/OK or cross/FAILED to show whether the back-up was successful or not.

You will be able to click on the **FileSystem** name to view the files that were backed up or the **View Errors** icon to see what errors the back-up reported.

The status and error messages can be either in plain text or you can specify icons (obviously you will need to create these or download them from the Internet). The code will allow you to use either option

– it is fully documented on what to do. By default, you will see plain text because not many sites will have the GIFs available, but, once they are created, you can edit the code and use the GIFs instead (which looks a lot better).

GETTING THE SCRIPTS UP AND RUNNING

For the purpose of this article, the back-up script is called **backup.cpio**, the script to re-run the back-up is called **rerun_backup** and the two CGI scripts are called **display_files** and **results_<*server name*>**.

If you change these names then please note the following:

- **display_files** is called from **results_<*server name*>** so you will need to change the name in that script.
- **rerun_backup** is called from **backup.cpio** so you will need to change the name in that script.
- **backup.cpio** is called from **rerun_backup** so you will need to change the name in that script.

The **results_<*server name*>** script must have the server name as the second argument following the '_'. (For example, if your server is called **server1** you could call the script **cgiscript_server1**, **backup_server1**, etc). If you do not follow this naming convention then the **display_files** script will not correctly pick up the server name.

Now you need to change a few variables in the **backup.cpio** script. These are fully documented in the script but here is what needs changing:

- **LOGDIR** – this is the name of the directory on the server where the results files will be held. Change the pathname as necessary but leave the $DAY variable on the end. The sub-directories in **LOGDIR** will be created if necessary, eg **LOGDIR=/usr/local/log/$DAY**.
- **PRINTER** – if you want the results file printing then specify a printer name here and un-comment the relevant line of code later in the script.

- **TAPE** and **TAPENR** – define the tape device names. **TAPENR** must be the non-rewind tape device.
- **MAIL_LIST** – specify a list of users who will receive e-mails about back-up problems and failures.
- **WEB_SERVER** – define the name of the Web Server that is hosting the CGI scripts.
- **WEBDIR** – define the directory name on the **WEB_SERVER** that some of the results files will be copied to for viewing by the CGI scripts. These subdirectories are created as necessary if they do not exist. Change the pathname as necessary but leave the $SERVER and $DAY variables on the end, eg **WEBDIR=/usr/local/backup_results/$SERVER/$DAY**.

  If the CGI scripts are located on the same server as the back-up is run on then set this directory name to be the same as **LOGDIR**.

Next you need to change one variable in the **results_<*server name*>** script:

- **RESULTS_DIR** – this needs to set to the same values as **WEBDIR** (in **backup.cpio**), eg **RESULTS_DIR=/usr/local/backup_results/$SERVER/$DAY**.

Finally you need to change one variable in the **display_files** script:

- **RESULTS_DIR** – this needs to set to the same values as **WEBDIR** (in **backup.cpio**), eg **RESULTS_DIR=/usr/local/backup_results/$SERVER/$DAY**.

You now need to decide where to put all these scripts. A good choice for **backup.cpio** and **rerun_backup** would be a directory called **/usr/local/backups**. Make sure that you change the directory name in **rerun_backup** for the location of **backup.cpio**.

The CGI scripts (**display_files** and **results_<*server*>**) both need to go in the same directory on your Web Server where you would normally locate CGI programs (this is never a pre-defined place). If you start using GIF files, then place these in the relevant place (eg with Apache, they are located under the **htdocs** directory).

Finally set up a **cron** job to run the **backup.cpio** to run every evening before 12 pm (this will ensure that the day number on the tape matches the day of the month that the back-up is run on).

RERUN_BACKUPS

```
#!/bin/ksh
# Written by Nicola Pinney - March 2001
#
# This is the script to re-run the back-up. Change the pathname
# of the back-up script as necessary.
#
echo "\nBack-up job will now be re-submitted\n"
ulimit 4194304
at now << !
/usr/local/backups/backup.cpio
!
```

RESULTS_NIMBUS

```
#!/bin/ksh
##############################################################################
# Written by Nicola Pinney - March 2001
# Displays the results of back-ups on the server for the last 31 days
# and also the monthly back-ups for the last 12 months. Defaults to
# showing yesterday's back-up results as today's back-up won't have been
# run yet!
##############################################################################
# All this to get the server name and capitalize the first letter for
# displaying on the Web page. The script must have the server name as
# the second parameter separated by an underscore - eg results_server1
SERVER="`echo $0|cut -d "_" -f2`"
typeset -L1u FIRST=$SERVER
SERVERNAME=$FIRST$REST$(echo $SERVER | cut -c2-)

# Main section - start of HTML code
echo Content-Type: text/html
echo
echo "<font face=arial>"
echo "<head><title>Back-up Results</title>"
echo "<h1 align=center>"
echo "$SERVERNAME Back-up Results"
echo "</h1>"
echo "</head>"
echo "<html>"

# Need to know what day it was yesterday as the intranet page defaults
```

```
# to showing these back-up results when you open it.
YESTERDAY=`TZ=GMT+24 date '+%d'`
TODAY=`date '+%d'`
DAY=1
echo "$REQUEST_URI" | grep date  > /dev/null && DAY=`echo "$REQUEST_URI"
|cut -d = -f2` > /dev/null
[[ -z "$DAY" ]] && DAY=$YESTERDAY

# If the day is Ø1, then we actually want to display the month as the
# Monthly back-up is always done on the 1st of the month.

if [ "$DAY" = "Ø1" ]
then
      echo "JanFebMarAprMayJunJulAugSepOctNovDec" |grep -q "$DAY" ||
        DAY=`TZ=GMT+24 date '+%b'`
fi

# Allow the user to select any day Ø2 - 31 or any month Jan - Dec to
# view the back-up results.

echo "<form method=\"get\">"
echo "<center>Choose back-up date <select name=\"date\">"

while [ $DAYNO -lt 31 ]
do
      DAYNO=`expr $DAYNO + 1`

#    If the day is 2 - 9 then add a leading 'Ø' to keep the spacing nice

      if [ $DAYNO -lt 1Ø ]
      then
            TEMPDAY="Ø$DAYNO"
            DAYNO=$TEMPDAY
      fi

#      If this is the day we selected then leave it as the chosen day in
#      the pull-down list.

      if [ "$DAYNO" = "$DAY" ]
      then
            echo "<option value=\"$DAYNO\" selected>$DAYNO</option>"
      else
            echo "<option value=\"$DAYNO\">$DAYNO</option>"
      fi
done

for MONTH in Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
do

#      If this is the month we selected then leave it as the chosen
```

```
#      month in the pull-down list.

      if [ "$DAY" = "$MONTH" ]
      then
            echo "<option value=\"$MONTH\" selected>$MONTH</option>"
      else
            echo "<option value=\"$MONTH\" >$MONTH</option>"
      fi
done

echo "</select>"
echo "<input type=\"submit\">"
echo "</input>"
echo "</form>"

# Change this variable as necessary to point to the directory where the
# results files are kept.
RESULTS_DIR=/usr/local/backup_results/$SERVER/$DAY

# Display the table headings and only use 80% of the width of the screen
echo "<table align=center border=1 width=80%>"
echo "<tr><th align=center>FileSystem</th><th>Status</th><th>Start
Time</th><th>End Time</th><th># Files</th><th>Size</th><th>Errors</th></
tr>"

while read FILESYSTEM RESULTS START END FILES BLOCKS ERRORS
do
      if [ "$FILESYSTEM" = "$SERVER" ]
      then
            BACKUP_DATE="$RESULTS $START"

#        If you need to add a message about the back-up for any reason
#        then you can add comments on the first line of the results file
#        and they will get displayed on the intranet page.

            MESSAGE="$END $FILES $BLOCKS $ERRORS"
            echo "<h2 align=left>"
            echo "$BACKUP_DATE"
            echo $MESSAGE
            echo "</h2>"
            continue
      fi

#-----------------------------------------------------------------------
#  Uncomment the following lines if you have created GIFs for the
#  various states of the back-up. Comment out the corresponding lines
#  below as well.
#  Change the directory and GIF name as necessary to point to the
#  correct location.
#
```

```
#      case $RESULTS in
#            OK)      ICON="<img src=\"/gif/green_tick.gif\">" ;;
#            WRITING) ICON="<img src=\"/gif/writing.gif\">" ;;
#            WRITTEN) ICON="<img src=\"/gif/written.gif\">" ;;
#            VERIFYING) ICON="<img src=\"/gif/verifying.gif\">" ;;
#            VERIFIED) ICON="<img src=\"/gif/verified.gif\">" ;;
#            FAILED) ICON="<img src=\"/gif/cross.gif\">" ;;
#            KILLED) ICON="<img src=\"/gif/killed.gif\">" ;;
#      esac
#-------------------------------------------------------------------------
#   By default the status will be shown in plain text and not GIF files.
      case $RESULTS in
            OK)      ICON="OK" ;;
            WRITING) ICON="WRITING" ;;
            WRITTEN) ICON="WRITTEN" ;;
            VERIFYING) ICON="VERIFYING" ;;
            VERIFIED) ICON="VERIFIED" ;;
            FAILED) ICON="FAILED" ;;
            KILLED) ICON="KILLED" ;;
      esac

#       Convert the '/' in the filesystem name to an '_'
       DIR=`echo $FILESYSTEM | tr -s  '\Ø57' '\137'`
      if [ "$DIR" = "_" ]
        then
                DIR=root
        fi

#    Check that we have the results file from the verify and make this a
#    link you can click on to view the files that were backed up.
#    The shell script "display_files" is called with the day number, the
#    file name of the results file, the server name, and the filesystem
#    that you want to view the results for.

      if [ -f $RESULTS_DIR/verify_$DIR ]
      then

#   Change the pathname below to point to the relevant CGI directory and
#   change the filename as necessary.
            NEWVAR="<a
href=\"/cgi-bin/npinney/display_files?DAY=$DAY&file=verify_$DIR&server=$
SERVER\">$FILESYSTEM</a>"

#      If for some reason we have lost the verify results file then just
#      display the filesystem name but it will not be a link.
      else
            NEWVAR=$FILESYSTEM
      fi

#     If the filesystem back-up failed then check for an 'errors' file
```

```
as
#     we can then create a link to view the errors.

      if [ "$RESULTS" = "FAILED" ]
      then
            if [ -f $RESULTS_DIR/errors_$DIR ]
            then

#   Check that we have the error results file and make this a link you
#   can click on to view the errors. The shell script "display_files" is
#   called with the day number, the file name of the errors file and the
#   server name. Change the CGI directory and the filename as necessary.
#
#   Uncomment the following line if you are using a GIF as the Back
#   button. Comment out the corresponding line below.
#
#      ERRORSVAR="<a
href=\"/cgi-bin/npinney/display_files?DAY=$DAY&file=errors_$DIR&server=$
SERVER\"><img src=\"/gif/viewerror.gif\" border=Ø alt=\"View Errors\"></
a>"

# By default use plain text for the "View Errors"
      ERRORSVAR="<a
href=\"/cgi-bin/npinney/display_files?DAY=$DAY&file=errors_$DIR&server=$
SERVER\">View Errors<a border=Ø alt=\"View Errors\"></a>"
            else
#                  The errors file is missing for some reason so leave
#                  the field blank.

                  ERRORSVAR=" "
            fi
      else
#                  The back-up worked ok so leave the field blank.
                  ERRORSVAR=" "
      fi

# Display the rest of the data about the back-up - filesystem name,
# status icon, start time, end time, number of files backed up, size of
# the filesystem back-up, and a link, if necessary, to the errors file.

      echo "<tr><td>$NEWVAR</td><td align=center>$ICON</td><td
align=center>$START</td><td align=center>$END</td><td
align=right>$FILES</td><td align=right>$BLOCKS</td><td
align=center>$ERRORSVAR</td></tr>"
done < $RESULTS_DIR/results

# End of the above table
echo "</table>"

# Start a new table that contains a "Back Button" icon
```

```
echo "<table valign=bottom align=left border=Ø>"
echo "<tr><td><br></td></tr>"
echo "<tr><td>"

# Add the URL of where you want the "Back Button" to go to. Change the
# CGI directory and pathname as necessary for the Back button GIF.
# Uncomment the line below if you are using a GIF as the Back button and
# comment out the corresponding line below.
#
# echo "<a href=\"http://nicolaswebsite/cgi-bin/npinney/
default.html\"><img
src=\"/gif/back.gif\" border=Ø alt="Go back"></a>"

# By default plain text will be used for the Back button
echo "<a href=\"http://nicolaswebsite/cgi-bin/npinney/
default.html\">Back<a
border=Ø alt="Go back"></a>"

# Close the table
echo "</td></tr></table>"
```

DISPLAY_FILES

```
#!/bin/ksh
# Written by Nicola Pinney - March 2ØØ1
# This script is called when you want to view either the list of files
# backed up or the errors generated.
#
# Capture the environment so we can look at variables
env > /tmp/env$$
FILE=`grep REQUEST_URI /tmp/env$$ | cut -d = -f4 | cut -d "&" -f1`
DAY=`grep REQUEST_URI /tmp/env$$ | cut -d = -f3 | cut -d "&" -f1`
SERVER=`grep REQUEST_URI /tmp/env$$ | cut -d = -f5`
FILESYSTEM=`echo $FILE | cut -c8-999 | tr -s '\137' '\Ø57'`

# Change the location of the results file as necessary.
RESULTS_DIR="/usr/local/backup_results/$SERVER/$DAY/"

# Capitalize the first letter of the server name
typeset -L1u FIRST=$SERVER
SERVERNAME=$FIRST$REST$(echo $SERVER | cut -c2-)

# Start of HTML coding
echo "Content-Type: text/html\n\n"
echo "<html>"
echo "<head><title>$SERVERNAME - $FILESYSTEM</title>"
echo "</head>"
echo "<body>"
echo "<A NAME=\"top\"></A>"
```

```
#-------------------------------------------------------------------------
# Uncomment the line below if you want to use a GIF instead of plain
# text. You may need to change the location of the "gif" directory and
# the name of the GIF as necessary. Comment out the corresponding line
# below as well.
# BACKREF="<a href=\"$HTTP_REFERER\"><img src=\"/gif/back.gif\" border=Ø
alt=\"Go back\"></a>"
#-------------------------------------------------------------------------

# By default we will use plain text to display links
BACKREF="<a href=\"$HTTP_REFERER\">Back<a border=Ø alt=\"Go back\"></a>"

echo "$BACKREF"
echo "<font face=arial>"
echo "<h2 align=left>"
echo "$SERVERNAME - $FILESYSTEM"
echo "</h2>"

# Display the first line of the results file as this has the day and
# date
BACKUP_DATE=`head -1 $RESULTS_DIR/results | awk '{print $2, $3'}`

echo "<h3 align=left>"
echo "$BACKUP_DATE"
echo "</h3>"

# Count the number of files that were backed up and if this is greater
# than 5Ø then assume that the browser won't be able to view them all on
# one page so add a "Go to Bottom" line.

LINES=`cat $RESULTS_DIR/$FILE | wc -l`
if [ "$LINES" -gt "5Ø" ]
then
      echo "<A href=\"#bottom\">Go to bottom</A>"
fi

# The <pre> tags are needed to space out the output of the text file
# correctly
echo "<pre>"
cat $RESULTS_DIR/$FILE
echo "</pre>"

# Remove temporary file as we do not need anymore parameters from it.
rm /tmp/env$$ > /dev/null 2>&1

# If the number of files that were backed up is greater than 5Ø
# then assume that the browser won't be able to view them all on one
# page. We have already added a "Go to Bottom" line so now add a link to
# the top of the page.
```



```
if [ "$LINES" -gt "5Ø" ]
then
      echo ""
      echo "<A NAME=\"bottom\"></A>"
      echo "<A href=\"#top\">Go to top</A>"
      echo "<P>"
      echo "$backref"
fi

# Closing HTML stuff
echo "</body>"
echo "</html>"
```

BACKUP_CPIO

```
#!/bin/ksh
#       author: Nicola Pinney
#       date:   March 2ØØ1
#       name:   backup.cpio
#
#   Uses cpio to sequentially back-up all filesystems to DLT/DAT and
#   then verify the back-up. The results are copied to a Web Server so
#   they can be viewed on the intranet.
#       Runs each night from cron at 11:3Ø p.m.
*******************************************************************************

#==============================================================================
# Define and initialize variables - these do not need changing
#==============================================================================

# Define servers
SERVER=`uname -n`

# Get date and time details
TIME=`date '+%H:%M'`
TODAY_DATE=`date '+%a %d/%m/%y'`
DAY=`date '+%d'`

# Check for day Ø1 as this is the monthly tape so change the $DAY
# variable to be the month name.
if [ "$DAY" = "Ø1" ]
then
    DAY=`date '+%b'`
fi

# General variables
FAILED=Ø
```

```
#=============================================================================
#   Variables that need configuring
#=============================================================================
# LOGDIR is the name of the directory on the server where the results
# files will be held. Change the pathname as necessary but leave the
# $DAY variable on the end. The subdirectory in LOGDIR will created if
# necessary.

LOGDIR=/usr/local/log/$DAY
[[ ! -d $LOGDIR ]] && mkdir -p $LOGDIR

# If you want the results file printing then specify a Printer name here
# and un-comment the relevant line of code - search for 'lp' in this
# script.

PRINTER=p00988

# Define tape devices - TAPENR must be the non-rewind tape device

TAPE=/dev/rmt/0m
TAPENR=/dev/rmt/0mn

# Define a list of Users who will receive e-mails about back-up problems
# & failures.

MAIL_LIST="techsupp operator"

# Define the name of the Web Server that is hosting the CGI scripts

WEB_SERVER=cgiserver

# Define the directory name on the WEB_SERVER that some of the results
# files will be copied to for viewing by the CGI scripts. These sub-
# directories are created as necessary if they do not exist. Change the
# pathname a necessary but leave the $SERVER and $DAY variables on the
# end.

WEBDIR=/usr/local/backup_results/$SERVER/$DAY
remsh $WEB_SERVER "ksh -c \"[[ ! -d $WEBDIR ]] && mkdir -p $WEBDIR\""
#=============================================================================
# Define file names
HEADER=$LOGDIR/header
RESULTS=$LOGDIR/results
#=============================================================================
# trap kill commands so that status file updated accordingly
#=============================================================================
trap cleanup 2
#=============================================================================
# Define umask
#=============================================================================
```

```
umask Ø22
#=============================================================================
# Function Declarations
#=============================================================================
# Function to perform cleanup when script killed
cleanup()
{
      KILL_TIME=`date '+%H:%M:%S'`
      sed 's/WRITING/KILLED/' $RESULTS | sed 's/VERIFYING/KILLED/' >
$RESULTS.tmp
      mv $RESULTS.tmp $RESULTS
      echo "BACKUP_INCOMPLETE KILLED --:-- $KILL_TIME -- --" >> $RESULTS
      remcopy_result_file
      exit 1
}

# Function to convert blocks to K, MB and GB
getsize()
{
      if [[ ! -s $BLOCKS ]]
      then
            SIZE=Unknown
      else
            BKP_SIZE_B=`cat $BLOCKS | awk '{print $1}'`
            BKP_SIZE_K=`expr $BKP_SIZE_B / 2`
            if [ $BKP_SIZE_K -gt 1Ø23 ] && [ $BKP_SIZE_K -lt 1Ø48576 ]
            then
                  BKP_SIZE_M=`expr $BKP_SIZE_K / 1Ø24`
                  SIZE="${BKP_SIZE_M}MB"
            else
                  if [ $BKP_SIZE_K -ge 1Ø48576 ]
                  then
        BKP_SIZE_G=`echo $BKP_SIZE_K | awk '{printf"%.2f", $1/1Ø48576}'`
                        SIZE="${BKP_SIZE_G}GB"
                  else
                        SIZE="${BKP_SIZE_K}KB"
                  fi
            fi
      fi
}

# Function to copy results file to web server
remcopy_result_file()
{
      rcp -p $RESULTS $WEB_SERVER:$WEBDIR/
}

#=============================================================================
# Main bit
#=============================================================================
```

```
#-------------------------------------------------------------------------
# Remove files from previous back-ups and initialize new results file
#-------------------------------------------------------------------------
remsh $WEB_SERVER "rm -f $WEBDIR/*"
rm -f $LOGDIR/*

# Create initial results file
echo $SERVER $TODAY_DATE > $RESULTS
remcopy_result_file

#-------------------------------------------------------------------------
# Build tape header after removing the existing one
#-------------------------------------------------------------------------
rm -f $HEADER
echo $SERVER $DAY > $HEADER
date >> $HEADER
df >> $HEADER

#-------------------------------------------------------------------------
# Check if tape OK and write header to tape
#-------------------------------------------------------------------------

# Try rewinding tape and mail error message to Operators if it fails.
mt -t $TAPE rew

if [ "$?" != "Ø" ]
then
      mailx -s "$SERVER backup tape needs attention" $MAIL_LIST << EOF

-------------------------------------------------------------------------
NOTE: This e-mail only applies to the Operator on shift on $TODAY_DATE
at $TIME.
If that isn't you then please ignore it.
-------------------------------------------------------------------------

IMPORTANT:      $SERVER - back-up tape not loaded correctly

There is a problem rewinding the tape so the back-up CANNOT proceed.
Ensure the tape is correctly loaded and try running the back-up again.

You can do this as follows:

Open a terminal session on your Workstation and login to $SERVER:-

            rlogin $SERVER -l root

(enter root's password)

Type the following:
```

```
      /usr/local/backups/rerun_backup

If you do not get this e-mail again then the back-up will be running.
EOF
      exit 2
fi

# write header and e-mail Operators if it fails
dd if=$HEADER of=$TAPENR bs=32k > /dev/null 2>&1
if [ "$?" != "Ø" ]
then
      mailx -s "$SERVER back-up tape needs attention" $MAIL_LIST << EOF

------------------------------------------------------------------------
NOTE: This e-mail only applies to the Operator on shift on $TODAY_DATE
at
$TIME.
If that isn't you then please ignore it.
------------------------------------------------------------------------

IMPORTANT:      $SERVER - back-up tape cannot be written to.

There is a problem writing to the tape so the back-up CANNOT proceed.
Ensure the tape is correctly loaded and is not write-protected and then
run the back-up again.

You can do this as follows:

Open a terminal session on your Workstation and login to $SERVER:-

            rlogin $SERVER -l root

(enter root's password)

Type the following:

      /usr/local/backups/rerun_backup

If you do not get this e-mail again then the back-up will be running.
EOF
      exit 2
fi

#------------------------------------------------------------------------
# Get list of mounted filesystems
#------------------------------------------------------------------------
FS_LIST=`df | awk '{print $1}'`
NUMBER=`df | wc -l`
```

```
#-------------------------------------------------------------------------
# Back up filesystems with relative pathnames
#-------------------------------------------------------------------------
for DEV in $FS_LIST
do
    # Swap the '/' character in the filesystem name for an '_' to create
    # a legal filename to be used.
      FS=`echo $DEV | tr -s '\Ø57' '\137'`
      if [ "$FS" = "_" ]
      then
            FS=root
      fi
      cd $DEV

      BLOCKS=$LOGDIR/blocks_$FS
      CPIO_OUTPUT=$LOGDIR/backup_$FS
      START_TIME=`date '+%H:%M:%S'`
      #START_SECS=$(/usr/local/bin/epoch_secs)

      echo "$DEV WRITING $START_TIME --:-- -- --" >> $RESULTS
      remcopy_result_file

#      Do the cpio back-up
      find . -xdev -print | cpio -ocvB > $TAPENR 2> $CPIO_OUTPUT
      RET=$?

#      Check exit status of cpio command
      if [[ $RET -ne Ø ]]
      then
           echo "\nCpio command returned error code $RET" >>$CPIO_OUTPUT
      fi
#      END_SECS=$(/usr/local/bin/epoch_secs)
      END_TIME=`date '+%H:%M:%S'`

# Capture the number of blocks backed up and then remove it from
# the file
      grep "[Ø-9]* blocks" $CPIO_OUTPUT > $BLOCKS
      sed '/[Ø-9]* blocks/d' $CPIO_OUTPUT > $CPIO_OUTPUT.tmp
      mv $CPIO_OUTPUT.tmp $CPIO_OUTPUT

#      Get the number of files that were backed up
      NUM_FILES=`cat $CPIO_OUTPUT| wc -l`

      getsize
      if [[ $SIZE = "Unknown" ]]
      then
            echo "\nCpio command may not have completed successfully -
no block count written" >> $CPIO_OUTPUT
      fi
```

```
      sed "s%$DEV WRITING $START_TIME --:-- -- --%$DEV WRITTEN
$START_TIME
$END_TIME $NUM_FILES ${SIZE}%" $RESULTS > $RESULTS.tmp
      mv $RESULTS.tmp $RESULTS
done

#-------------------------------------------------------------------------
# Rewind tape and skip header
#-------------------------------------------------------------------------
mt -t $TAPE rew
dd if=$TAPENR  bs=32k > /dev/null 2>&1

#-------------------------------------------------------------------------
# Verify the back-up
#-------------------------------------------------------------------------
for DEV in $FS_LIST
do
      # Swap the '/' character for an '_'.
      FS=`echo $DEV | tr -s '\Ø57' '\137'`
      if [ "$FS" = "_" ]
      then
            FS=root
      fi

      BLOCKS=$LOGDIR/blocks_$FS
      CPIO_OUTPUT=$LOGDIR/backup_$FS
      ERRORS=$LOGDIR/errors_$FS
      FULL_VERIFY=$LOGDIR/full_verify_$FS
      VERIFY=$LOGDIR/verify_$FS

      sed "s%$DEV WRITTEN%$DEV VERIFYING%" $RESULTS > $RESULTS.tmp
      mv $RESULTS.tmp $RESULTS
      remcopy_result_file

      cd $DEV
      cpio -itcvB < $TAPENR > $FULL_VERIFY 2> /dev/null

      sed "s%$DEV VERIFYING%$DEV VERIFIED%" $RESULTS > $RESULTS.tmp
      mv $RESULTS.tmp $RESULTS
      remcopy_result_file
#     Cut out just the filenames from the full list of files backed up

      cat $FULL_VERIFY | cut -d: -f3- | cut -c1Ø-999 > $VERIFY

#     Now compare the back-up and verify lists of files
      diff $CPIO_OUTPUT $VERIFY > $ERRORS 2>&1
      if [ $? -eq Ø ]
      then
            if [[ -s $ERRORS ]]
            then
```

```
                  rm -f $ERRORS
            fi
            sed "s%$DEV VERIFIED%$DEV OK%" $RESULTS > $RESULTS.tmp
            mv $RESULTS.tmp $RESULTS
      else
            sed "s%$DEV VERIFIED%$DEV FAILED%" $RESULTS | sed "s%$DEV
FAILED .*%& $ERRORS%" > $RESULTS.tmp
            mv $RESULTS.tmp $RESULTS
            FAILED=`expr $FAILED + 1`
      fi
      remcopy_result_file
done

#-------------------------------------------------------------------------
# Print results - uncomment this line if you want the results file
# printing out
#-------------------------------------------------------------------------
#lp -d $PRINTER $RESULTS

#-------------------------------------------------------------------------
# rewind tape
#-------------------------------------------------------------------------
mt -t $TAPE rew

# Copy to remote web server
rcp $LOGDIR/[!bh]* $WEB_SERVER:$WEBDIR/

#-------------------------------------------------------------------------
# Handle any failures
#-------------------------------------------------------------------------
# E-mail Mail List recipients if there are any failures.
if [ "$FAILED" -gt "Ø" ]
then
      mailx -s "$SERVER: $FAILED back-ups have failed on $TODAY_DATE"
$MAIL_LIST << EOF
$FAILED back-ups have failed on $SERVER. This may be caused by temporary
files so please check the results on the intranet page:

            http://nicolaswebsite/cgi-bin/npinney/results_$SERVER
EOF
fi

# If the number of FAILED back-ups is more than 1/3 of the number of
# expected back-ups then e-mail Operators to have a look and possibly
# re-run back-up.
threshold=`expr $NUMBER / 3`

if [ "$FAILED" = "$NUMBER" ]
then
      mailx -s "$SERVER: All back-ups have failed!" $MAIL_LIST << EOF
```

```
------------------------------------------------------------------------
NOTE: This e-mail only applies to the Operator on shift on $TODAY_DATE
at
$TIME.
If that isn't you then please ignore it.
------------------------------------------------------------------------

ALL back-ups have failed on $SERVER. This could be because of a corrupt
tape or the tape drive may need cleaning.

FIRSTLY: Please go and check the drive to see if it needs cleaning. You
may want to clean it anyway.

SECONDLY: If you suspect that the tape is at fault then use another one
(if you can find one).

FINALLY: Try re-running the back-up:

You can do this as follows:

Open a terminal session on your Workstation and login to $SERVER:-

            rlogin $SERVER -l root

(enter root's password)

Type the following:

      /usr/local/backups/rerun_backup

If you do not get this e-mail again then the back-up will be running.
EOF

      exit 2
fi

if [ "$FAILED" -ge "$threshold" ]
then
      mailx -s "$SERVER: $FAILED back-ups have failed!" $MAIL_LIST <<
EOF

------------------------------------------------------------------------
NOTE: This e-mail only applies to the Operator on shift on $TODAY_DATE
at $TIME.
If that isn't you then please ignore it.
------------------------------------------------------------------------

$FAILED back-ups have failed out of a total $NUMBER on $SERVER.
```

```
These could be geniune failures or there may be a problem with either
the tape drive or the tape itself.

FIRSTLY: Check out the failures on the Intranet. Click on the "View
Errors" icon to get an idea of why the back-ups are failing.

If you see error messages such as:

            write failed: I/O error
            Can't open /dev/tty.

then it is likely that there is either a problem with the tape drive or
the tape.

If the error messages are along the lines of:

      <filename>: No such file or directory
      Cannot stat .

then it is nothing to worry about and you will not need to re-run the
back-up.

SECONDLY: Please go and check the drive to see if it needs cleaning. You
may want to clean it anyway.

THIRDLY: If you suspect that the tape is at fault then use another one
(if you can find one).

FINALLY: Try re-running the back-up:

You can do this as follows:

Open a terminal session on your Workstation and login to $SERVER:-

            rlogin $SERVER -l root

(enter root's password)

Type the following:

      /usr/local/backups/rerun_backup

If you do not get this e-mail again then the back-up will be running.
EOF

      exit 2
fi
```

A simple blend of AIX, Oracle, and TSM

We have a rather large investment in Oracle and TSM (formerly ADSM) on our AIX servers. One of the things we have always struggled with is a simple approach to Oracle datafile back-ups with TSM. We have tried both EBU (for Oracle 7.*x*) and RMAN (Oracle 8 and 8*i*), as well as the TSM add-on for Oracle in an AIX environment, but have never been satisfied with the un(ease) of use, or the interaction between AIX/ADSM and Oracle. For those of you that have tried these approaches I'm sure you're familiar with how cumbersome and sometimes difficult it is to clean up past EBU/RMAN back-ups that were done to TSM.

We have also tried managing the tapes ourselves in scripts, but found this to be also tough to maintain.

What we have decided on instead is a few simple Korn shell scripts in AIX that, we think, suit our requirements just fine. The basic flow is as follows:

1 Run an SQL script that creates a list of database files to be backed up, in TSM macro format.

2 Bring down the database.

3 Run the cold back-up with the TSM archive command.

4 Bring the database back up.

These four steps are scheduled via **cron**, with ample time in between the third and fourth steps to assure time to finish. These scripts assume you have a window of opportunity to perform cold back-ups. If not the scripts and SQL could easily be modified to issue alter tablespace begin back-up commands to turn this into a hot back-up approach (consult your DBA for specifics).

SCRIPT SUMMARY

All scripts are run as the Oracle DBA user.

build_cold_backup.sh <SID>

The Oracle SID is passed into this script to keep it generic enough for multiple database use. It will produce output that looks like this for SID oltp2:

```
archive -ARCHmc=ARCHIVE4 /u04/oradata/oltp2/system01.dbf
archive -ARCHmc=ARCHIVE4 /u02/oradata/oltp2/tools01.dbf
archive -ARCHmc=ARCHIVE4 /u06/oradata/oltp2/rbs01.dbf
archive -ARCHmc=ARCHIVE4 /u05/oradata/oltp2/temp01.dbf
……….
archive -ARCHmc=ARCHIVE4  /u01/app/oracle/product/817/dbs/initoltp2.ora
```

There is one line for each Oracle database file, control file, and the parameter file.

Notes

We allow archived redo logs to be picked up with normal nightly TSM processing, which is why you do not see them as part of this script.

ARCHIVE4 is a TSM management class that keeps four versions of an archived file. (We run this cold back-up weekly, so this yields one month.) Archived REDO logs are kept at least this long.

cold_backup.sh <SID>

This script issues the TSM **dsmc** script to back-up the files. It also produces a fairly detailed log file of its activity, and a 'stripped-down' version of the log for mailing to admins, then checks that the appropriate number of files was backed up, then e-mails the log to the appropriate admin people.

The log file is the detailed output that results from a TSM archive command. The stripped-down log file for mailing looks like this:

```
Begin Cold Back-up of  oltp2   2001-03-18 04:12

Check the state of Oracle processes for oltp2 :
  oracle 29028 52694   1 04:12:01      -  0:00 grep oltp2
  oracle 52694  7306  12 04:12:00      -  0:00 ksh /u01/home/oracle/
admin/backup
s/cold_backup.sh oltp2

Record list of Oracle files to back-up, as per getdbfiles.sql:
```

```
archive -ARCHmc=ARCHIVE4 /u04/oradata/oltp2/system01.dbf
archive -ARCHmc=ARCHIVE4 /u02/oradata/oltp2/tools01.dbf
archive -ARCHmc=ARCHIVE4 /u06/oradata/oltp2/rbs01.dbf
…….
archive -ARCHmc=ARCHIVE4 /u05/oradata/oltp2/control03.ctl
archive -ARCHmc=ARCHIVE4  /u01/app/oracle/product/817/dbs/initoltp2.ora

Begin dsmc (ADSM) back-up of data files
status and errors are being recorded in  /u01/home/oracle/admin/backups/
logs/olt
p2.coldbu.log.20010318

End of dsmc ADSM archive requests  2001-03-18 04:21
Number of files to be backed up:  14
Number of files successfully backed up:  14

Catalog of files archived:
Normal File-->       283,123,712 /u04/oradata/oltp2/system01.dbf [Sent]
Normal File-->        25,174,016 /u02/oradata/oltp2/tools01.dbf [Sent]
Normal File-->       629,153,792 /u06/oradata/oltp2/rbs01.dbf [Sent]
…….
Normal File-->       734,011,392 /u05/oradata/oltp2/temp01.dbf [Sent]
Normal File-->     1,048,584,192 /u03/oradata/oltp2/users01.dbf [Sent]
Normal File-->         3,432,448 /u05/oradata/oltp2/control03.ctl [Sent]
Normal File-->             2,622 /u01/app/oracle/product/817/dbs/
initoltp2.ora [
Sent]

End of Processing  2001-03-18 04:21
```

SCRIPTS

build_cold_backup.sh

```
#!/usr/bin/ksh
. $HOME/.profile

# DB/INSTANCE/NODE and other SPECIFICS

ORACLE_SID=`echo $1`
export ORACLE_SID

if [[ $ORACLE_SID = '' ]]; then
        echo "NO SID supplied, usage is cold_backup.sh <SID> "
        exit
fi
```

```
BUDIR=$HOME/admin/backups;export BUDIR
LOGDIR=$HOME/admin/backups/logs;export LOGDIR
MAIL=$LOGDIR/$ORACLE_SID.coldbackup.mail.log;export MAIL
DT=`date +%Y%m%d`;export DT;
LOG=$LOGDIR/$ORACLE_SID.coldbu.log.$DT

# clear log and lst files

touch $LOG
> $MAIL
rm -f $BUDIR/oracle_dbfiles.$ORACLE_SID.lst

# run sqlplus script that queries the database to find all database
# files
# output them in another file with adsm archive syntax attached.

sqlplus internal/ @$BUDIR/getdbfiles.sql >> $LOG

# rename the file list with SID name to keep unique

mv $BUDIR/oracle_dbfiles.lst $BUDIR/oracle_dbfiles.$ORACLE_SID.lst

# make sure we pick up the parameter file

echo "archive -ARCHmc=ARCHIVE4 " $ORACLE_HOME/dbs/init$ORACLE_SID.ora >>
$BUDIR/
oracle_dbfiles.$ORACLE_SID.lst
```

cold_backup.sh

```
#!/usr/bin/ksh

. $HOME/.profile
# DB/INSTANCE/NODE and other SPECIFICS
PNAME=`echo $1`
export PNAME
ORACLE_SID=`echo $1`
export ORACLE_SID
if [[ $ORACLE_SID = '' ]]; then
        echo "NO SID supplied, usage is cold_backup.sh <SID> "
        exit
fi
BUDIR=$HOME/admin/backups;export BUDIR
LOGDIR=$HOME/admin/backups/logs;export LOGDIR
MAIL=$LOGDIR/$ORACLE_SID.coldbackup.mail.log;export MAIL
DT=`date +%Y%m%d`;export DT;
LOG=$LOGDIR/$ORACLE_SID.coldbu.log.$DT

# make sure log is empty (in case of rerun)
>$LOG
```

```
# store date/time for echoing into log file

XDT=`date "+%Y-%m-%d %H:%M"`;export XDT
echo "Begin Cold Back-up of " $ORACLE_SID " " $XDT >> $MAIL
echo " " >> $MAIL

echo "Check the state of Oracle processes for " $ORACLE_SID ":" >> $MAIL
ps -ef | grep $ORACLE_SID >> $MAIL
echo " " >> $MAIL

echo "Record list of Oracle files to back-up, as per getdbfiles.sql:" >>
$MAIL
echo " " >> $MAIL

cat $BUDIR/oracle_dbfiles.$ORACLE_SID.lst >> $MAIL
echo " " >> $MAIL

echo "Begin dsmc (ADSM) back-up of data files" >> $MAIL
echo "status and errors are being recorded in " $LOG >> $MAIL
echo " " >> $MAIL

# Use ADSM archive function to back-up the data files.
# The list of datafiles was created in a previous script.

dsmc MACRO $BUDIR/oracle_dbfiles.$ORACLE_SID.lst -password=client >>$LOG

XDT=`date "+%Y-%m-%d %H:%M"`;export XDT
echo "End of dsmc ADSM arcive requests " $XDT >>$MAIL

# check the number of files created bu Oracle getdbfiles sql against
# the number of files successfully archived with ADSM

IN=`grep -c ARCHmc $BUDIR/oracle_dbfiles.$ORACLE_SID.lst`;export IN;
OUT=`grep -c "finished without failure" $LOG`;export OUT;

echo "Number of files to be backed up: " $IN >> $MAIL
echo "Number of files successfully backed up: " $OUT >> $MAIL

echo " " >> $MAIL; echo "Catalog of files archived:" >> $MAIL
grep 'Normal File' $LOG >> $MAIL

XDT=`date "+%Y-%m-%d %H:%M"`;export XDT
echo " " >>$MAIL; echo "End of Processing " $XDT >>$MAIL

if [[ $IN -ne $OUT ]] then
   echo "#*#*#*# WARNING NUMBER OF FILES TO BE BACKED UP DOES NOT MATCH
#*#*#*#"
>> $MAIL
fi
```

```
# mail the log

mail -s$ORACLE_SID.coldbu.$DT @ntmail_server:yourself@bhs.org < $MAIL
mail -s$ORACLE_SID.coldbu.$DT @ntmail_server:your.dba@bhs.org < $MAIL
```

SQL used by build_cold_backup.sh (getdbfiles.sql)

```
set heading off
set feedback off
set wrap off
spool /u01/home/oracle/admin/backups/oracle_dbfiles
select 'archive -ARCHmc=ARCHIVE4 ' || name from v$datafile;
select 'archive -ARCHmc=ARCHIVE4 ' || member from v$logfile;
select 'archive -ARCHmc=ARCHIVE4 ' || name from v$controlfile;
spool off
exit
```

start_db.sh <SID>

These are the two scripts we use to start up and shutdown the database – yours may already exist or vary, of course.

```
#!/usr/bin/ksh
. $HOME/.profile
ORACLE_SID=$1; export ORACLE_SID
svrmgrl << END
connect internal
startup
exit
END
```

stop_db.sh <SID>

```
#!/usr/bin/ksh
. $HOME/.profile
ORACLE_SID=$1; export ORACLE_SID
svrmgrl << END
connect internal
shutdown immediate
startup mount
shutdown
exit
END
```

David Miller
Database Architect
Baystate Health Systems (USA)

Implementing I/O multi-pathing using AutoPath

The widespread availability of modern storage devices providing multiple connections to internally-defined storage units (LUNs), such as IBM's MSS, ESS ('Shark') and HP's XP256 and XP512, prompted our implementation of I/O multi-pathing support for AIX. This capability, previously available only with SSA-based devices, enables two important features:

1 Load balancing – multi-path load balancing of data traffic prevents a single I/O path from becoming overloaded when many I/O operations are directed to common devices along the same I/O path. Normally, this is performed on a global rotating basis.

2 Failover – enables the server to perform automatic failover to an alternative I/O path, in case the I/O path used by the server stops working because of a failure in the I/O adapters or cables. Restoring the physical connection does not automatically restore the connection to the lost I/O path.

This article will discuss I/O multi-pathing implemented by AutoPath XP software Version 1.00.01, supplied by HP for IBM's Unix servers connected to XP256 and XP512 storage devices.

COMPATIBILITY AND INSTALLATION REQUIREMENTS

Compatibility and installation requirements are:

- IBM RISC/6000 system with AIX Version 4.2.1 with PTF IX62304, Version 4.3.2, or Version 4.3.3 with APAR IY04634 and IY05369. Ensure that the **bos.adt** package is installed. The host system can be a uniprocessor (UP) or a multiprocessor (SMP).
- HP SureStore E disk array configured for Unix/NT (not OS/400), Version 52-46-11-00/00 or later.
- The following are types of SCSI or Fiber Channel adapters (minimum of two to a maximum of eight):

- IBM SCSI-2 Differential Fast/ Wide PCI Bus Adapter.
- IBM Enhanced SCSI-2 Differential Fast/ Wide Adapter.
- IBM FC6227 Fiber Channel Adapter:
    - AIX Version 4. 3.3 with APAR U470141, U470126, and U467115.
    - HBA device driver 4.3. 3.0 for AIX Version 4.3.3 are required.
    - HBA firmware version must be SF2.23 or later.

- You must configure the SCSI/Fiber Channel adapters prior to installing and using Auto Path XP for AIX. (Auto Path XP also supports a single SCSI/ Fiber Channel adapter on the host system. However, load balancing and failover are not provided for a single path.)
- SCSI/ Fiber Channel cables to connect each SCSI/ Fiber Channel host adapter to a storage system controller port or FC Switch/ Hub port.
- AIX system administrator privileges.

INSTALLATION INSTRUCTIONS

The installation instructions are:

1 Log in as **root** user.

2 Insert the Auto Path XP installation CD- ROM into the CD- ROM drive.

3 From your desktop window, enter smitty install_update. The **Install and Update Software** menu displays.

4 Highlight **Install and Update from Latest Available Software** and press *Enter*.

5 Press F4. The **Input device/ directory for software** screen displays.

6 Select the CD-ROM drive you are using for the installation (for

example */dev/ cd0*) and press *Enter*. The **Install and Update from Latest Available Software** screen displays.

7 Highlight **Software to Install** and press F4. The **Software to Install** screen displays.

8 Select *dpo. ibmssd* and press *Enter*. The **Install and Update from Latest Available Software** screen displays with the name of the software you selected to install. The fileset name is *dpo.ibmssd.rte* with an AIX version level (for example *dpo.ibmssd.ret.432*). The description of the fileset is *Auto Path XP runtime for AIX*.

9 Check the default option settings to ensure that they are correct.

10 Press *Enter* to install. The following message displays:

ARE YOU SURE?

Continuing may delete information you may want to keep.

This is your last chance to stop before continuing.

11 Press *Enter* to continue. The installation takes several minutes.

12 When the installation is finished, press F10 to exit from SMIT. Remove the installation CD-ROM.

CONFIGURATION INSTRUCTIONS

The configuration instructions are:

1 Make sure that the disk array is running, is configured to support the AIX system, and is physically connected to the server.

2 Make sure that *dpo.ibmssd.rte* software is installed on the server.

3 From your desktop window, enter smitty devices. The **Devices** menu displays.

4 Highlight **Data Path Devices** and press *Enter*. The **Data Path Devices** screen displays.

5 Highlight **Define and Configure all Data Path Devices** and press *Enter*. Configuration begins.

6 Use AutoPath XP-specific LVM commands to create or change volume groups that use virtual paths (vpath) devices provided by Auto Path XP as explained in later sections.

Another option to perform Auto Path XP device configuration is to reboot the server using the **shutdown –Fr** command.

To verify configuration of the Auto Path XP on an AIX host system, follow the steps below:

1 Enter smitty from your desktop window. The **System Management Interface Tool** Menu displays.

2 Highlight **Devices** and press *Enter*. The **Devices** menu displays.

3 Highlight **Data Path Device** and press *Enter*. The **Data Path Device** screen displays.

4 Highlight Display **Data Path Device Configuration** and press *Enter*. A list is displayed of the condition (either **Defined** or **Available)** of all Data Path displayed pseudo devices, in addition to the multiple paths of each device. If any device is listed as **Defined,** the configuration was not successful. Check the configuration procedure again.

To verify multiple-attached paths to each adapter connected to a disk array port, follow these steps:

1 Enter smitty from your desktop window. The **System Management Interface Tool** menu displays.

2 Highlight **Devices** and press *Enter*. The **Devices** menu displays.

3 Highlight **Data Path Devic**e and press *Enter*. The screen displays.

4 Highlight **Display Data Path Device Adapter Statu**s and press *Enter*. The screen displays all attached paths of each adapter.

UNCONFIGURATION INSTRUCTIONS

In order to unconfigure Auto Path XP devices you should perform the following procedure. First, unmount all the file systems from the volume groups that are using the Auto Path XP devices. Then, run the

vp2hd volume group conversion script to convert the volume group from Auto Path XP devices **(vpathN**) to disk array physical devices (hdisks).

Using the Software Management and Interface Tool (SMIT), you can unconfigure the Auto Path XP devices in two ways. Either you can unconfigure without deleting the device from the Object Database Management (ODM) database, or you can delete information device information from the ODM database. If you unconfigure without deleting the device information, the device remains in the **Defined** condition. Using SMIT, you can return it to an **Available** condition.

If you delete the device information from the ODM database, that device is removed from the system. To return it, follow the procedure described under *Configuration* instruction heading.

Follow these steps to unconfigure the Data Path devices:

1 Enter smitty from your desktop window. The **System Management Interface Tool** menu displays.

2 Highlight **Device**s and press *Enter*. The screen displays.

3 Highlight **Data Path Devic**e and press *Enter*. The screen displays.

4 Highlight **Remove a Data Path Device** and press *Enter*. A list of all Data Path devices and their condition (either **Defined** or **Available**) is displayed.

5 Select the device that you want to unconfigure. Select whether or not you want to delete the device information from the ODM database.

6 Press *Enter*. The device is unconfigured to the condition that you selected.

Note: to remove all devices from your system and deinstall the Auto Path XP software, all the Data Path devices must be removed from your host system. Select **Yes** on the **Deleting Device Information from ODM database** field. You can then remove the Auto Path XP software from your AIX host system.

Follow the steps below to remove (deinstall) the Auto Path XP software:

1 Enter smitty **deinstall** from your desktop window to go directly to the **Remove Installed Software** panel. The screen displays.

2 Enter **dpo. ibmssd.rte** in the **SOFTWARE** name field.

3 Press *Enter*.

4 Press the **Tab** key in the **PREVIEW Only?** field to toggle between Yes and No.

Select **No** to remove the software package from your AIX host system.

Note: if you select **Yes**, the deinstall process stops at this point and previews what you are removing (deinstalling). The results of your deinstall precheck are displayed without removing the software. If the condition for any Data Path device is either **Available** or **Defined,** this deinstall will fail.

5 Select **No** for the remaining fields on this screen.

6 Press *Enter*. SMIT responds with the following message:

ARE YOU SURE??

Continuing may delete information you may want to keep.

This is your last chance to stop before continuing.

7 Press *Enter* to begin the deinstall process. This might take a few minutes.

8 When the deinstall process completes, the Auto Path XP software package is removed from your system.

USING THE AUTOPATH XP SOFTWARE

After you configure the Auto Path XP it creates DPO device special files (vpath device) for disk array logical units (LUNs). These are accessible through the connection between the AIX host server FC adapter and the disk array ports. Since the common AIX disk driver always creates the original disk array device special files (hdisk), applications now have two ways in which to access disk array devices.

To use the load balancing and failover features of the Auto Path XP and access disk array devices, your application must use the DPO device vpath special files rather than the disk array device hdisk special files.

Two types of application use disk array storage. One type, such as Oracle or DB2, might access disk array devices directly by DPO devices vpath. For this type of application, use the Auto Path XP device vpath (raw device). The other type of application uses disk array devices through AIX Logical Volume Management (LVM). For this type of applications, you must create the volume group with the Auto Path XP vpath device.

To provide failover protection, a pseudo device must include a minimum of two paths. Both the pseudo device and the hdisk devices must also be **Available**. In the example below, vpath0 has a single path and, therefore, will not provide fail-over protection because there is no alternative path to the disk array LUN. The other pseudo devices each have two paths and, therefore, can provide failover protection.

To display which pseudo devices are available to provide failover protection, use either the Auto Path XP **Display Data Path Device Configuration** SMIT panel, or enter the **lsvpcfg** command. Following is an example of output that might be displayed using either of these methods:

```
vpath0 (Available pv) 00035791_0336__ = hdisk13 (Available )
vpath1 (Available pv) 00035791_0342__ = hdisk2 (Available ) hdisk3 (Available )
vpath17 (Available pv) 00035791_0260__ = hdisk19 (Available ) hdisk34 (Available)
vpath18 (Available pv) 00035791_0266__ = hdisk20 (Available ) hdisk35 (Available)
vpath19 (Available pv) 00035791_0271__ = hdisk21 (Available ) hdisk36 (Available)
vpath20 (Available pv) 00035791_0277__ = hdisk22 (Available ) hdisk37 (Available)
vpath21(Available pv) 00035791_0278_ = hdisk23(Available ) hdisk38  (Available)
```

The output displays the following information:

- The name of each pseudo device; for example, vpath17.
- The defined condition of the pseudo device. It is either Defined or Available. There is no failover protection if one path is Defined. All paths to each pseudo device must be Available to have failover protection. This condition also indicates whether or not the pseudo device is defined to AIX as a physical volume (**pv**

flag). If **pv** is displayed for both devices, you might not have failover protection.

- The unit serial number of the disk array LUN, for example 00035791_0277.
- The names of the AIX disk devices that comprise the pseudo devices, their configuration, and the physical volume status.

To display similar information you can also use the Auto Path XP **datapath query device** command, which will be described later.

To create a volume group with Auto Path XP pseudo devices, use SMIT to select the vpath pseudo devices that are included in the volume group from those that can provide failover protection. You can create a volume group from vpath pseudo devices that have only a single path and then reconfigure the connections to add paths later; however, you will not have failover protection.

Follow these steps to create a new volume group with Auto Path XP vpaths:

1 Enter **SMIT** from your desktop window. The System Management Interface Tool displays.

2 Highlight **System Storage Management (Physical & Logical Storage**) and press *Enter*. The screen displays.

3 Highlight **Logical Volume Manage**r and press *Enter*. The menu displays.

4 Highlight **Volume Group** and press *Enter*. The screen displays.

5 Highlight **Add Volume Group with Data Path Device**s and press *Enter*.

Press F4 while highlighting the **PHYSICAL VOLUME name**s field to list all available Auto Path XP vpaths.

If you use a script file to create a volume group in order to create a DPO volume group, you must modify your script file and replace the **mkvg** command with the **mkvg4vp** command.

Once you create the volume group, AIX creates the pseudo device,

‘physical volumes’ (**pv**). To list all the physical volumes known to AIX, enter the **lspv** command. Any vpath pseudo devices that were created into physical volumes are included in the output. To display which devices comprise a volume group, enter the **lsvg -p *vgname*** command. For example, the command **lsvg - p vg00** might produce the following output:

```
vg00:
PV_NAME    PV STATE   TOTAL PPs     FREE PPs    FREE DISTRIBUTION
vpath18    active      437          18          00..00..00..00..18
vpath19    active      437          19          00..00..00..00..19
vpath20    active      437          19          00..00..00..00..19
vpath21    active      437          19          00..00..00..00..19
```

This indicates that the **vg00** volume group uses physical volumes vpath18 through vpath21.

All the functions that apply to a regular volume group also apply to a volume group that is built using Data Path devices. Use SMIT to create logical and file systems as usual.

AUTOPATH XP COMMANDS

Installation of Auto Path XP software introduces the following new AIX commands:

- datapath query adapter
- datapath query device
- datapath set adapter
- datapath set device
- mkvg4vp
- hd2vp
- vp2hd
- lsvpcfg
- extendvg4vp
- dpovgfix
- xpinfo (supplied by HP with XP256 or XP512 disk array).

You must login as superuser (**root**) in order to be able to use these command.

The datapath query adapter command displays information about single or all available adapters.

Syntax:

```
datapath query adapter [adapter number]
```

If the adapter number is absent, the command displays information about all available adapters.

For example:

```
datapath query adapter

Active Adapters :2

Adpt#  Adapter Name   State     Mode     Select    Errors  Paths  Active
    Ø  fscsiØ         NORMAL    ACTIVE   8396587      Ø     16      16
    1  fscsi1         DEGRAD   ACTIVE    6771625     1Ø     19      17
```

The fields are:

- Adpt # – adapter number.
- Adapter name – adapter name. In our case IBM's FC 6227 Adapters.
- State – operational state of the adapter:

  NORMAL – the adapter is in use.

  DEGRAD – one or more paths are not functioning.

  FAILED – the adapter is not used by Auto Path XP software.
- Mode – either ACTIVE or OFFLINE.
- Select – the number of times the adapter was selected for input or output.
- Errors – the number of errors on a path connected to the adapter.
- Paths – the number of paths attached to the adapter. This is also the number of physical and logical devices attached to the adapter.

- Active – the number of functioning paths attached to the adapter. This number is equal to the number of paths attached, less the number of FAILED or OFFLINE paths.

Editor's note: this article will be concluded in next month's issue.

Alex Polyak
System Engineer
APS Israel

Contributing to *AIX Update*

In addition to *AIX Update*, the Xephon family of *Update* publications now includes *CICS Update*, *MVS Update*, *TCP/SNA Update*, *VSAM Update*, *DB2 Update*, *Domino Update*, *MQ Update*, *NT Update*, *Oracle Update*, *RACF Update*, and *TSO/ISPF Update*. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with AIX, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it. A copy of our *Notes for Contributors*, which explains the terms and conditions under which we publish articles, is available from our Web site at www.xephon.com/contnote.html. Articles can be sent to Trevor Eddolls at any of the addresses shown on page 2, or e-mailed to trevore@xephon.com

AIX news

LSI Logic Storage Systems has announced Version 7.10 of its SANtricity Storage Manager software, with expanded host and volume support, along with enclosure upgrades to the MetaStor E2400 and E4400 that increase storage density per square foot of floor space by 40%.

SANtricity 7.10 features include host support for AIX, Solaris, Windows NT, Windows 2000, Linux, HP-UX, IRIX, and NetWare.

Also included are double the number of storage partitions and a quadrupling of volumes per storage system. SANshare partitioning enables a single system to function as up to 16 separate logical systems, each capable of supporting an independent host.

For further information contact:
LSI Logic Storage Systems, 12110 Sunset Hills Rd, Suite 450, Reston, VA 20190, USA.
Tel: (703) 390 9040.
URL: http://www.lsilogicstorage.com.

* * *

Mainsoft is shipping its Visual MainWin, which lets developers use the Microsoft Visual Studio development environment to rapidly deploy applications simultaneously on Windows, AIX, Solaris, HP-UX and Linux.

It allows the entire multi-platform development process to be completed from any PC with no direct interaction with a Unix machine.

It is available for AIX, Solaris, HP-UX, and Red Hat Linux.

For further information contact:
Mainsoft, 3850 North First Street, San Jose, CA 95134, USA.
Tel: (408) 544 1400.
URL: http://www.mainsoft.com.

* * *

IBM has announced AIX 5L Version 5.1, promising the ability to run on both its own POWER systems and Intel Itanium platforms.

The Unix 98-branded operating system supports existing 32-and 64-bit hardware systems, integrates Java and IP multipath routing, and has a range of development tools, including a Performance Toolbox for system profiling and tuning.

It enables the use of certificate revocation lists with the Internet Key Exchange (IKE) protocol for authenticating remote users or devices, enhancing the AIX IPsecurity function for virtual private networking support.

AIX Developer Kit, Java 2 Technology Edition is an interface between the AIX kernel and Java technology to help improve application scalability and performance over the range of IBM Unix servers. It also improves AIX for Java runtime and applications development.

Also new is an improved version of Journal File System, JFS2, which allows data to be stored in a more contiguous manner.

For further information contact your local IBM representative.
URL: http://www-1.ibm.com/servers/aix/news.