47

AIX update

September 1999

In this issue

AIX Update

Published by

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

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

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

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

Editor

Harold Lewis

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £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* is available from Xephon's Web page at *www.xephon.com/aixupdate* (you'll need the user-id shown on your address label to access it).

Using lsof and lslk

LSOF

lsof is a general-purpose, portable, public domain utility that lists files and sockets being accessed by the operating system and user programs. An open file may be a regular file, a directory, a block special file, a character special file, an executing text reference, a library, a stream, or a network file (Internet socket, NFS file, or Unix domain socket).

The latest version of the program is 4.42, which supports AIX 4.1, 4.2, and 4.3 (AIX version 3 is supported by **lsof** version 2.36).

INSTALLATION

The source code for the program is available by FTP from *ftp://vic.cc.purdue.edu* in the directory */pub/tools/unix/lsof*. After retrieving and unpacking the archive file, you must configure the software to match both your version of Unix (AIX in our case) and your compiler (in most cases this will be IBM's C or C++ compiler or GCC). To do this you must change directory to the root directory of **lsof** source code tree and then type the command:

```
    # ./Configure aix
```

(or **aixgcc**, if you are using GCC compiler).

The **Configure** script invokes three other scripts: **AFSConfig**, **Inventory**, and **Customize**.

AFSConfig locates AFS header files and establishes the version of AFS (if installed). The **Inventory** script checks the completeness of the source code in the **lsof** directory. The **Customize** script allows the installer to select values of the most important compile-time options for the compilation of **lsof** (unless experienced in the use of the compiler, you are expected to accept the defaults put forward by the script). After the **Configure** script terminates, the top-level directory of the source code tree contains the source files and the makefile needed to generate **lsof** for your version of AIX and the compiler you're using.

The next step is to invoke **make** command to compile and link the program:

```
# make
```

Please note that the generated program should be executed only on machines running the same level of the operating system as the development machine on which the program is compiled and made, as the program is closely tied to the structure of operating system internals that are frequently altered by the supplier of the operating system.

The makefile created doesn't contain code to perform the actual installation of the **lsof** program generated, which doesn't need to be installed in a specific location, though it does require read access to special memory device files, such as */dev/mem* and */dev/kmem*.

Another consideration is that, in order to allow non-priveleged users to execute **lsof**, the command's permissions should be set in such a way that the command obtains privileged user permissions during execution. This can be done by setting the ownership to 'superuser' and execution permission to have 'setgid' properties. For instance, on my system the command has the following permissions:

```
-rwxr-sr-x  1 root     system 140946 Mar 30 /urs/local/bin/lsof
```

Additionally it is necessary to update the scripts supplied with the program to point to the directory in which the program is installed. The manual page, which is distributed with the program, should be installed in the directory that contains manual pages of other public domain tools (these are typically found in */usr/local/man*).

USAGE

lsof has many options and, consequently, is accompanied by a very long and detailed manual entry that runs to no fewer than 26 pages. This section, however, contains examples only of common tasks that can be performed using the program.

It should be noted that **lsof** is unable to retrieve file path name components from AIX's kernel name cache, so the command will report the file system device name instead of the file's path name.

To list all open files and sockets, execute the following command:

```
# lsof
```

You should specify **lsof**'s **-X** flag to see all text (program) files and shared libraries open in the system. This flag is AIX-specific and was introduced as a work-around to a bug that used to cause **lsof** to hang. I should note that I've never encountered this problem on systems on which I have worked.

On my workstation, which is less heavily used than some, the **lsof** command produces 456 lines of output, while it produces 917 lines of output with **-X** flag. Therefore most users will probably want to limit the information produced by **lsof** by using various flags.

Use the following command to find all files open on a particular filesystem:

```
# lsof <filesystem-name>
```

This command is handy when you are looking for files that are being written to and risk filling the filesystem.

To find a file that's a running executable resident on the filesystem and is preventing you from unmounting the filesystem, use the following command:

```
# lsof  -X <filesystem-name>
```

Use the following command to find all network socket files that are open on your machine:

```
# lsof  -i
```

Use the following command to find servers running on the system and listening to socket files:

```
# lsof  -i | grep '*:'
```

Use the following command to find all network connections to a particular host (*server.acme.com* in our example):

```
# lsof  -i@server.acme.com
```

You can refine the query by supplying information about both the protocol (UDP or TCP) and port number or service name to the **-i** argument:

```
# lsof  -iTCP@server.acme.com:515
```

This option is also useful for finding a process connected to a particular network connection reported by the **netstat** command. For instance, if **netstat** reports following:

```
# netstat |fgrep login
tcp4  0  0  local.acme.i.login remote.acme..1023  ESTABLISHED
```

We can identify the process that is actually connected to the service named ‘login’ on local computer:

```
# lsof  -iTCP@local:login
```

The output will be similar to:

```
COMMAND   PID     USER   FD   TYPE   DEVICE       SIZE/OFF   NODE
➤  NAME
rlogind   13214   root   0u   IPv4   0x7008ced8   0t22       TCP
➤  local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
rlogind   13214   root   1u   IPv4   0x7008ced8   0t22       TCP
➤  local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
rlogind   13214   root   2u   IPv4   0x7008ced8   0t22       TCP
➤  local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
```

(Note the use of the continuation character, ‘➤’, in the output above to indicate that one line of output maps to more than one line of text.)

Alternatively, you can start by looking at the process control block (PCB) address reported by the command **netstat -A**:

```
# netstat -A|fgrep login
7008ced8 tcp4 0 0 local.acme.login remote.acme.1023 ESTABLISHED
```

You can then find the process by executing the following command:

```
# lsof -i|grep 7008ced8
```

Another nice trick is to find all files opened by a particular command:

```
# lsof -c <first_chacters_of_command_name>
```

Note that this will report all instances of the command that are currently executing. To find information about a particular instance of the command, first establish the process id (PID) associated with the instance of the command by using the output of the **ps** command, then execute following command line:

```
# lsof -p <PID>
```

Use **lsof**’s **-g** option to watch all files opened by processes that belong to a common process group (which all have the same parent process), as in the following example:

```
# lsof -g12345
```

Here ‘12345’ is the process group number.

lsof can be used to identify processes being run on the local system by remote users. The following steps perform this:

1 Find the processes with connections to a remote host:

```
$ lsof  -i@client
```

(The output is similar to one listed in the previous example.)

2 Find all open files that were opened by a process listed in the output of the previous command:

```
# lsof -p13214
```

This results in the following output:

```
COMMAND   PID USER     FD    TYPE   DEVICE       SIZE/OFF
➤  NODE NAME
rlogind   13214 root   cwd   VDIR   10,4         1024
➤  2 / (/dev/hd4)
rlogind   13214 root   0u    IPv4   0x7008ced8   0t22
➤  TCP local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
rlogind   13214 root   1u    IPv4   0x7008ced8   0t22
➤  TCP local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
rlogind   13214 root   2u    IPv4   0x7008ced8   0t22
➤  TCP local.acme.com:login->remote.acme.com:1023 (ESTABLISHED)
rlogind   13214 root   3r    VREG   10,5          1682
➤  15100 /usr (/dev/hd2)
rlogind   13214 root   4u    VCHR   26,2          0t0
➤  415 /dev/ptc/2
```

3 Identify the pseudo TTY device in the report from the previous command (in this case, */dev/ptc/2*).

4 Establish the process that uses the corresponding PTS device:

```
$ lsof /dev/pts/2
```

This produces output similar to the following:

```
COMMAND   PID  USER  FD   TYPE  DEVICE SIZE/OFF NODE NAME
csh     15396  alex  15u  VCHR  27,2   0t1212   670  /dev/pts/2
csh      15396  alex  16u  VCHR  27,2   0t1212   670  /dev/pts/2
csh      15396  alex  17u  VCHR  27,2   0t1212   670  /dev/pts/2
csh      15396  alex  18u  VCHR  27,2   0t1212   670  /dev/pts/2
csh      15396  alex  19u  VCHR  27,2   0t1212   670  /dev/pts/2
```

All the above steps are combined in a single script that you'll find located in the *Scripts* subdirectory of **lsof**'s source code distribution tree. The script is called **idrlogin**, with versions provided for both Perl 4 and Perl 5. The script assumes that the **lsof** command is located in the parent directory – you should customize the script to conform to the location of both **perl** and **lsof** on your system.

```
# idrlogin.perl
Login   Shell   PID     Via       PID     TTY     From
root    ksh     14090   telnetd   13468   pts/0   test.acme.com
alex    csh     15396   rlogind   13214   pts/2   remote.acme.com
```

To view NFS files opened by processes running on a local NFS client, you should invoke **lsof** with following argument:

```
# lsof -N
COMMAND  PID    USER  FD   TYPE  DEVICE  SIZE/OFF  NODE
➤  NAME
more     15108  alex  cwd  VDIR  NFS,13  1536      2
➤  /usr/local (remote:/usr/local)
more     15108  alex  3r   VREG  NFS,13  1731      20501
➤  /usr/local (remote:/usr/local)
csh      15396  alex  cwd  VDIR  NFS,13  1536      2
➤  /usr/local (remote:/usr/local)
```

Note that **lsof** is unable to report files opened by remote NFS clients from the local NFS server, as the program is unable to examine the processes running on remote machines.

You can list files opened by processes that belong to a particular user by executing either of following commands:

```
# lsof -u<login_name>
```

or

```
# lsof -u<user_id>
```

If you want to omit certain users from a **lsof** report, you should prefix the user name in the **lsof** argument with a caret ('^'). For instance, to ignore all open files that were opened by processes owned by *root*, you should execute one of the following commands:

```
# lsof -u ^root

# lsof -u ^0
```

Normally **lsof** performs a logical 'OR' on the options that are supplied to it. To change this behaviour, use the **-a** argument. For instance, to observe files opened by process '1234' that perform FTP transfers, we may issue the following command:

```
# lsof -p1234 -a -d9,10
```

In the above example, '9' and '10' are file descriptors that are normally used for FTP transfers.

The **-r** option instructs **lsof** to repeat its inquiry every 15 seconds (by default), though a different interval may be specified after the flag. The example below is used to monitor RCP file transfers, which use descriptors '3' and '4'.

```
# lsof -p1234 -a -d3,4 -r
```

Another important option is **-F**, which directs **lsof** to produce field output that can easily be parsed by shell, Perl, or **awk** scripts.

LSLK

The lock file lister program, **lslk**, reports on locks on local files on the executing system, such as locks set by the functions **flock**(2), **fcntl**(2), and **lockf**(3). Either local or remote processes that run on NFS clients of executing system may set the locks.

The source code for the program is available by FTP from *ftp://vic.cc.purdue.edu* in */pub/tools/unix/lslk* directory. At the time of writing, the latest version of the program is 1.20. I was able successfully to configure and install it on both AIX 4.3.2 and AIX 4.1.5 systems. Installation and configuration are similar to **lsof**.

Running the command on an AIX 4.1.5 system produces the following report:

```
SRC       PID  DEV  INUM SZ TY M ST WH END        LEN NAME
dtlogin  3128 10,6   105  5  w 0  0  0   0 2147483647 /var (/dev/hd9var)
qdaemon 14724 10,6   266  0  w 0  0  0   0 2147483647 /var (/dev/hd9var)
qdaemon 15496 10,6   248  0  w 0  0  0   0 2147483647 /var (/dev/hd9var)
```

and the one below comes from an AIX 4.3.2 system:

```
SRC        PID  DEV  INUM SZ  TY  M  ST  WH  END        LEN NAME
dtlogin   3370 10,6   108  5   w  0   0   0    0 4294967295 /var (/dev/hd9var)
i4llmd   12386 10,7    24  0   r  1   0   0    0 4294967295 /tmp (/dev/hd3)
```

Below are some examples of how this command is commonly used.

To list all locks, issue the command without any flags:

```
    # lslk
```

To list all locks held by processes that are executing on remote host *client*, execute:

```
    # lslk -i client
```

To list all locks held by processes ‘1234’ and ‘5789’, execute:

```
    # lslk -p 1234,5789
```

To list all locks held by process ‘1234’, which runs on the host *client*, execute:

```
    # lslk -p 1234 -a -i client
```

Alex Polak
System Engineer
APS (Israel)

A function library for VT420 programming

Below is a series of library functions for controlling VT420 and lower terminals using escape sequences (the library is implemented as a Korn shell script). Using these functions should take much of the effort out of coding output for character-based terminals.

VT420.LIB

```
#!/bin/ksh

# Use these escape sequences with caution. You're encouraged to
# experiment with them before using them in production code.
```

```
############################################################
#               Miscellaneous routines                     #
############################################################

# OutputToScreen <text>
OutputToScreen ( )
{
    print -n "$*";
}

# OutputSequence <Ps1..Psn>
# Ps = Code to output
OutputSequence ( )
{
    while [[ $# -ne 1 ]]; do
        print -n "$1;";
        shift 1;
    done

    print -n "$1";
}

# TurnCursor ON|OFF
TurnCursor ( )
{
    TextCursorEnableMode $1;
}

# DisplayOnStatusLine [<Text>]
DisplayOnStatusLine ( )
{
    SelectStatusLineType $HostWritable;
    SelectActiveStatusDisplay $StatusLine;
    print -n "\n$*"
    SelectActiveStatusDisplay $MainDisplay;
}

############################################################
#                 Character encoding                       #
############################################################

# Control characters

# CO (7-Bit) control characters

# NUL
# Null
# NUL has no function.
Null ( )
{
    print -n "${NUL}";
}

# ENQ
# Enquiry
# Sends the "answer-back" message (communications set up).
Enquiry ( )
{
    print -n "${ENQ}";
}

# BEL
# Bell
# Sounds the bell (if enabled in the keyboard set-up).
Bell ( )
{
    print -n "${BEL}";
}

# BS
# BS
# Moves the cursor one character position to the left. If the cursor
# is at the left margin, no action occurs.
BS( )
{
    print -n "${BS}";
}

# HT
# HorizontalTab
# Moves the cursor to the next tab stop. If there are no more tab
# stops, the cursor moves to the right margin. HT does not cause
# text to auto wrap.
HorizontalTab ( )
{
    print -n "${HT}";
}

# LF
# LineFeed
# Generates a line feed or a new line, depending on the setting
# of line feed/new line mode.
LineFeed ( )
{
    print -n "${LF}";
}

# VT
# VerticalTab
# Treated as LF.
```

```
VerticalTab ( )
{
    print -n "${VT}";
}

# FF
# FormFeed
# Treated as LF.
FormFeed ( )
{
    print -n "${FF}";
}

# CR
# CarriageReturn
# Moves the cursor to the left margin on the current line.
CarriageReturn ( )
{
    print -n "${CR}";
}

# SO
# ShiftOut
# Maps the G1 character set to GL. You designate G1 using a select
# character set (SCS) sequence.
ShiftOut ( )
{
    print -n "${SO}";
}

# SI
# ShiftIn
# Maps the G0 character set to GL. You designate G0 using a select
# character set (SCS) sequence.
ShiftIn ( )
{
    print -n "${SI}";
}

# DC1
# DeviceControl1
# Also known as XON. Requires XON/XOFF flow control enabled in the
# communications set-up. DC1 clears DC3 (XOFF). This action causes
# the VT420 to continue sending characters.
DeviceControl1 ( )
{
    print -n "${DC1}";
}

# DC2
# DeviceControl2
DeviceControl2 ( )
{
    print -n "${DC2}";
}

# DC3
# DeviceControl3
# Also known as XOFF. Requires XON/XOFF flow control enabled in the
# communications set-up. DC3 stops the VT420 sending characters.
# The terminal cannot resume sending characters until it receives a
# DC1 control character.
DeviceControl3 ( )
{
    print -n "${DC3}";
}

# DC4
# DeviceControl4
# Introduces an SSU session management command. The VT420 and host
# use this control to separate SSU commands from both ANSI text and
# control functions.
DeviceControl4 ( )
{
    print -n "${DC4}";
}

# CAN
# Cancel
# Immediately cancels an escape sequence, control sequence, or device
# control string in progress. The VT420 does not display any error
# characters.
Cancel ( )
{
    print -n "${CAN}";
}

# SUB
# Substitute
# Immediately cancels an escape sequence, control sequence, or device
# control string in progress. The VT420 displays a reverse question
# mark as an error character.
Substitute ( )
{
    print -n "${SUB}";
}

# ESC
# Escape
# Introduces an escape sequence. ESC also cancels any escape sequence,
# control sequence, or device control string in progress.
Escape ( )
{
    print -n "${ESC}";
}

# DEL
# Delete
# Ignored when received, unless a 96-character set is mapped to GL.
# DEL is not used as a fill character - use NUL instead.
Delete ( )
{
    print -n "${DEL}";
}

# C1 (8-Bit) control characters

# IND
# Index
# Moves the cursor down one line in the same column. If the cursor
# is at the bottom margin, the page scrolls up.
Index ( )
{
    print -n "${IND}";
}

# NEL
# NextLine
# Moves the cursor to the first position on the next line. If the
# cursor is at the bottom margin, the page scrolls up.
Nextline ( )
{
    print -n "${NEL}";
}

# HTS
# HorizontalTabSet
# Sets a horizontal tab stop at the column where the cursor is.
HorizontalTabSet ( )
{
    print -n "${HTS}";
}

# RI
# ReverseIndex
# Moves the cursor up one line in the same column. If the cursor is
# at the top margin, the page scrolls down.
ReverseIndex ( )
{
    print -n "${RI}";
}

# SS2
# SingleShift2
# Maps the G2 character set to GL for the next graphic character
# only. You designate the G2 set using a select character set
# (SCS) sequence.
SingleShift2 ( )
{
    print -n "${SS2}";
}

# SS3
# SingleShift3
# Maps the G3 character set to GL for the next graphic character
# only. You designate the G3 set using a select character set
# (SCS) sequence.
SingleShift3 ( )
{
    print -n "${SS3}";
}

# DCS
# DeviceControlString
# Introduces a device control string. Used for loading either
# function keys or a soft character set.
DeviceControlString ( )
{
    print -n "${DCS}";
}

# SOS
# StartOfString
# Ignored.
StartOfString()
{
    print -n "${SOS}";
}

# DECID
# DECPrivateIdentification
# Makes the terminal send its Device Attribute, or 'DA', response
# to the host (same as an ANSI DA sequence). Programs should use
# the ANSI DA sequence.
DECPrivateIdentification ( )
{
    print -n "${DECID}";
}

# CSI
# Control_Sequence_Introducer
# Introduces a control sequence.
Control_Sequence_Introducer ( )
{
    print -n "${CSI}";
}

# ST
# StringTerminator
# Ends a device control string. You use ST in combination with DCS.
StringTermination ( )
{
    print -n "${ST}";
}

# OSC
# OperatingSystemCommand
# Introduces an operating system command. The VT420 ignores
# characters that follow until it receives a SUB, ST, or other C1
# control character.
OperatingSystemCommand ( )
{
    print -n "${OSC}";
}

# PM
# PrivacyMessage
# Introduces a privacy message string. The VT420 ignores
# characters that follow until it receives a SUB, ST, or other C1
# control character.
PrivacyMessage ( )
{
    print -n "${PM}";
}

# APC
# ApplicationProgramCommand
# Introduces an application program command. The VT420 ignores
# characters that follow until it receives a SUB, ST, or other C1
# control character.
ApplicationProgramCommand ( )
{
    print -n "${APC}";
}

# Using macros

# The VT420 lets you define and invoke macros to suit the needs of
# your application. A macro is a string of ANSI text and commands
# that are downloaded to the terminal. By invoking the macro, you
# execute a group of control functions with one operation.

# DECDMAC
# DefineMacro <Pid> <Pdt> <Pen> <D..D>
# Pid  = macro ID number (0-63).
# Pdt  = 0     => Delete all current macros.
# Pdt  = 1     => Delete all current macros.
# Pdt  = Other => terminal ignores the macro.
# Pen  = 0     => Standard ASCII characters.
# Pen  = 1     => Hex pairs for each ASCII character.
# Pen  = Other => Terminal ignores the macro.
# D..D = Control string data.
DefineMacro ( )
{
    print -n "${DCS}$1;$2;$3!z";
    shift 3;
    OutputSequence "$*";
    print -n "${ST}";
}

# DECINVM
# InvokeMacro <Pid>
# Pid = macro ID number.
InvokeMacro ( )
{
    print -n "${CSI}$1*z";
}

############################################################
#            Emulating VT series terminals                 #
############################################################
# DECSCL
# OperatingLevel <Level> [<Cntrl>]
# Level = 1       => VT100 mode.
# Level = 2, 3, 4 => VT400 mode.
# Cntrl = 0       => 8-bit controls.
# Cntrl = 1       => 7-bit controls (the default).
# Cntrl = 2       => 8-bit controls.
OperatingLevel ( )
{
    print -n "${CSI}6";
    OutputSequence $1 $2;
    print -n "\"p";
}

# DECNRCM
# CharacterSetMode h|l
# h = The terminal uses 7-bit characters from an NRC set.
# l = The terminal uses 7-bit and 8-bit characters from the DEC
#     multinational or ISO Latin-1 set.
```

```
CharacterSetMode ( )
{
    print -n "${CSI}?42$1";
}

############################################################
#                       Page memory                        #
############################################################

# Setting the page format

# DESCSCPP
# SetColumnsPerPage <Pn>
# Pn columns (80 or 132)
SetColumnsPerPage ( )
{
    print -n "${CSI}$1\$|";
}

# DESCCOLM
# ColumnMode h|l
# h = 132 columns
# l =  80 columns (the default)
ColumnMode ( )
{
    print -n "${CSI}?3$1";
}

# DECSLPP
# SetLinesPerPage <Pn>
# Pn lines per page. The number of pages depends on how many
# sessions you use.
# Pn    Dual sessions    Single sessions
# --    -------------    ---------------
# 24       3 pages           6 pages
# 25       2                 5
# 36       2                 4
# 48       1                 3
# 72       1                 2
# 144      -                 1
SetLinesPerPage()
{
    print -n "${CSI}$1t";
}

# DECSLRM
# SetLeftAndRightMargins <Pl> <Pr>
# Pl = Left column
# Pr = Right column
SetLeftAndRightMargins ( )
```

```
{
    print -n "${CSI}$1;$2s";
}

# DECSTBM
# SetTopAndBottomMargins <Pt> <Pb>
# Pt = Top column
# Pb = Bottom column
SetTopAndBottomMargins ( )
{
    print -n "${CSI}$1;$2r";
}

# DECOM
# OriginMode h|l
# h = Move within margins
# l = Move outside margins (the default)
OriginMode ( )
{
    print -n "${CSI}?6$1";
}

# DECVSSM
# VerticalSplitScreenMode h|l
# h = Left and right margins can be changed
# l = Left and right margins cannot be changed (the default)
VerticalSplitScreenMode ( )
{
    print -n "${CSI}?69$1";
}

# Moving through page memory

# NP
# NextPage <Pn>
# Move Pn pages forwards (C = home)
NextPage ( )
{
    print -n "${CSI}$1U";
}

# PP
# PrecedingPage <Pn>
# Move Pn pages backwards (C = home)
PrecedingPage ( )
{
    print -n "${CSI}$1V";
}

# PPA
# PagePositionAbsolute <Pn>
# Move to page Pn (C = same as previous page)
PagePositionAbsolute ( )
{
    print -n "${CSI}$1spP";
}

# PPB
# PagePositionBackward <Pn>
# Move Pn pages backwards (C = same as previous page)
PagePositionBackward ( )
{
    print -n "${CSI}$1spR";
}

# PPR
# PagePositionForward <Pn>
# Move Pn pages forward (C = same as previous page)
PagePositionForward ( )
{
    print -n "${CSI}$1spQ";
}

############################################################
#          Visual character and line attributes            #
############################################################

# Character and line attribute sequences

# SGR
# SelectGraphicRendition <Ps1..Psn>
# Ps = Character attribute value
SGR( )
{
    print -n "${CSI}$(OutputSequence $*)m"
}

SelectGraphicRendition (   )
{
    print -n "${CSI}";
    OutputSequence $*;
    print -n "m";
}

# DECSWL
# SingleWidthSingleHeightLine [<text>]
SingleWidthSingleHeightLine ( )
{
    print -n "${ESC}#5$*";
}
```

```
# DECDWL
# DoubleWidthSingleHeightLine [<text>]
DoubleWidthSingleHeightLine ( )
{
    print -n "${ESC}#6$*";
}

# DECDHL
# DoubleWidthDoubleHeightLine [<text>]
DoubleWidthDoubleHeightLine ( )
{
    SaveCursorState;
    print -n "${ESC}#3$*";
    RestoreCursorState;
    CursorDown 1;
    print -n "${ESC}#4$*";
    RestoreCursorState;
}

###########################################################
#                        Editing                          #
###########################################################

# Editing sequences

# IRM
# InsertReplaceMode h|l
# h = Insert characters
# l = Replace characters
InsertReplaceMode ( )
{
    print -n "${CSI}4$1";
}

# DECDC
# DeleteColumn <Pn>
# Pn columns
DeleteColumn ( )
{
    print -n "${CSI}$1'~";
}

# DECIC
# InsertColumn <Pn>
# Pn columns
InsertColumn ( )
{
    print -n "${CSI}$1'}";
}
```

```
# DL
# DeleteLine <Pn>
# Pn lines
DeleteLine ( )
{
    print -n "${CSI}$1M";
}

# IL
# InsertLine <Pn>
# Pn lines
InsertLine ( )
{
    print -n "${CSI}$1N";
}

# DCH
# DeleteCharacter <Pn>
# Pn characters
DeleteCharacter ( )
{
    print -n "${CSI}$1P";
}

# ICH
# InsertCharacter <Pn>
# Pn characters
InsertCharacter ( )
{
    print -n "${CSI}$1@";
}

# ED
# EraseInDisplay <Ps>
# Ps = 0 - Cursor to end (the default)
# Ps = 1 - Beginning to cursor
# Ps = 2 - Complete display
EraseInDisplay ( )
{
    print -n "${CSI}$1J";
}

# EL
# EraseInLine <Ps>
# Ps = 0 - Cursor to end (the default)
# Ps = 1 - Beginning to cursor
# Ps = 2 - Complete display
EraseInLine ( )
{
    print -n "${CSI}$1K";
}

# ECH
# EraseCharacter <Pn>
# Pn characters
EraseCharacter ( )
{
    print -n "${CSI}$1X";
}

# DECSCA
# SelectCharacterProtectionAttribute <Ps>
# Ps = 0 - DECSED and DECSEL can erase (the default)
# Ps = 1 - DECSED and DECSEL cannot erase
# Ps = 2 - DECSED and DECSEL can erase
SelectCharacterProtectionAttribute ( )
{
    print -n "${CSI}$1\"q"
}

# DECSED
# SelectiveEraseInDisplay <Ps>
# Ps = 0 - Cursor to end (the default)
# Ps = 1 - Beginning to cursor
# Ps = 2 - Complete display
SelectiveEraseInDisplay ( )
{
    print -n "${CSI}?$1J";
}

# DECSEL
# SelectiveEraseInLine <Ps>
# Ps = 0 - Cursor to end (the default)
# Ps = 1 - Beginning to cursor
# Ps = 2 - Complete display
SelectiveEraseInLine ( )
{
    print -n "${CSI}?$1K";
}

############################################################
#              Rectangular area operations                 #
############################################################

# Rectangular area control functions

# DECCRA
# CopyRectangularArea <Pts> <Pl> <Pbs> <Prs> <Pps> <Ptd> <Pld> <Ppd>
# Pts = Top edge
# Pl  = Left edge
# Pbs = Bottom edge
# Prs = Right edge
# Pps = Source page number
# Ptd = Destination top edge
# Pld = Destination left edge
# Ppd = Destination page number
CopyRectangularArea ( )
{
    print -n "${CSI}$1;$2;$3;$4;$5;$6;$7;$8\$v";
}

# DECERA
# EraseRectangularArea <Pt> <Pl> <Pb> <Pr>
# Pt = Top edge
# Pl = Left edge
# Pb = Bottom edge
# Pr = Right edge
EraseRectangularArea ( )
{
    print -n "${CSI}$1;$2;$3;$4\$z";
}

# DECFRA
# FillRectangularArea <Pch> <Pt> <Pl> <Pb> <Pr>
# Pch = Decimal code of fill character
# Pt  = Top edge
# Pl  = Left edge
# Pb  = Bottom edge
# Pr  = Right edge
FillRectangularArea ( )
{
    print -n "${CSI}$1;$2;$3;$4;$5\$x";
}

# DECSERA
# SelectiveEraseRectangularArea <Pt> <Pl> <Pb> <Pr>
# Pt = Top edge
# Pl = Left edge
# Pb = Bottom edge
# Pr = Right edge
SelectiveEraseRectangularArea ( )
{
    print -n "${CSI}$1;$2;$3;$4\${";
}

# DECSACE
# SelectAttributeChangeExtent <Ps>
# Ps = 0 - Stream of character positions affected
# Ps = 2 - Rectangular of character positions affected
SelectAttributeChangeExtent ( )
```

```
{
    print -n "${CSI}$1*x";
}

# DECCARA
# ChangeAttributesInRectangularArea <Pt> <Pl> <Pb> <Pr> <Ps1..Psn>
# Pt = Top edge
# Pl = Left edge
# Pb = Bottom edge
# Pr = Right edge
# Ps = Visual character attributes
ChangeAttributesInRectangularArea ( )
{
    print -n "${CSI}";
    OutputSequence $*;
    print -n "\$r";
}

# DECRARA
# ReverseAttributesInRectangularArea <Pt> <Pl> <Pb> <Pr> <Ps1..Psn>
# Pt = Top edge
# Pl = Left edge
# Pb = Bottom edge
# Pr = Right edge
# Ps = Visual character attributes
ReverseAttributesInRectangularArea ( )
{
    print -n "${CSI}";
    OutputSequence $*;
    print -n "\$t";
}

############################################################
#             Cursor movement and panning                  #
############################################################

# Enabling the cursor

# DECTCEM
# TextCursorEnableMode h|l
# h = Visible cursor (the default)
# l = Invisible cursor
TextCursorEnableMode ( )
{
    print -n "${CSI}?25$1";
}

# Moving the cursor

# DECBI
# BackIndex
BackIndex ( )
{
    print -n "${ESC}6";
}

# DECFI
# ForwardIndex
ForwardIndex ( )
{
    print -n "${ESC}9";
}

# CUP
# CursorPosition <Pl> <Pc>
# Line Pl, column Pc.
CursorPosition ( )
{
    print -n "${CSI}$1;$2H";
}

# HVP
# HorizontalAndVerticalPosition <Pl> <Pc>
# Line Pl, column Pc (using CUP instead is recommended)
HorizontalAndVerticalPosition ( )
{
    print -n "${CSI}$1;$2f";
}

# CUF
# CursorForward <Pn>
# Pn columns right
CursorForward ( )
{
    print -n "${CSI}$1C";
}

# CUB
# CursorBackward <Pn>
# Pn columns left
CursorBackward ( )
{
    print -n "${CSI}$1D";
}

# CUU
# CursorUp <Pn>
# Pn lines up
CursorUp ( )
{
    print -n "${CSI}$1A";
}

# CUD
# CursorDown <Pn>
# Pn lines down
CursorDown ( )
{
    print -n "${CSI}$1B";
}

# SU
# PanDown <Pn>
# Pn lines down
PanDown ( )
{
    print -n "${CSI}$1S";
}

# SD
# PanUp <Pn>
# Pn lines up.
PanUp ( )
{
    print -n "${CSI}$1T";
}

# DECVCCM
# VerticalCursorCouplingMode h|l
# h = Coupled (the default)
# l = Uncoupled
VerticalCursorCouplingMode ( )
{
    print -n "${CSI}61$1";
}

# DECPCCM
# PageCursorCouplingMode h|l
# h = Coupled (the default)
# l = Uncoupled
PageCursorCouplingMode ( )
{
    print -n "${CSI}64$1";
}

###########################################################
#        Keyboard, printing and display commands          #
###########################################################

############################
```

```
# Keyboard control sequences

# AM
# KeyboardAction h|l
# h = Locked
# l = Unlocked (the default)
KeyboardAction ( )
{
    print -n "${CSI}2$1";
}

# DECBKM
# BackarrowKey h|l
# h = Backspace
# l = Delete (the default)
BackarrowKey ( )
{
    print -n "${CSI}?67$1";
}

# LNM
# LineFeedNewLine h|l
# h = New Line
# l = Line feed (the default)
LineFeedNewLine ( )
{
    print -n "${CSI}20$1";
}

# DECARM
# Autorepeat h|l
# h = Repeat (the default)
# l = No repeat
Autorepeat ( )
{
    print -n "${CSI}?8$1";
}

# DECAWM
# Autowrap h|l
# h = Autowrap
# l = No autowrap (the default)
Autowrap ( )
{
    print -n "${CSI}?7$1";
}

# DECCKM
# CursorKeys h|l
# h = Application
# l = Cursor (the default)
CursorKeys ( )
{
    print -n "${CSI}?1$1";
}

# DECPAM
# KeypadApplicationNumeric =|>
# h = Application
# l = Numeric (the default)
KeypadApplicationNumeric ( )
{
    if [[ ".$1" = ".h" ]]; then
        print -n "${ESC}=";
    else
        print -n "${ESC}>";
    fi
}

# DECNKM
# NumericKeypadMode h|l
# h = Application
# l = Numeric (the default)
NumericKeypadMode ( )
{
    print -n "${CSI}?66$1";
}

# DECKBUM
# KeyboardUsageMode h|l
# h = Data processing.
# l = Typewriter (the default)
KeyboardUsageMode ( )
{
    print -n "${CSI}?68$1";
}

# DECKPM
# KeyPosition h|l
# h = Position reports
# l = Character codes (the default)
KeyPosition ( )
{
    print -n "${CSI}81$1";
}

# SRM
# SendReceiveMode h|l
# h = Local echo off (the default)
# l = Local echo on
```

```
SendReceiveMode ( )
{
    print -n "${CSI}12$1";
}

# DECSCNM
# ScreenMode h|l
# h = Light background
# l = Dark background (the default)
ScreenMode ( )
{
    print -n "${CSI}?5$1";
}

# DECSCLM
# ScrollingMode h|l
# h = Smooth scroll (the default)
# l = Jump scroll
ScrollingMode ( )
{
    print -n "${CSI}?4$1";
}

# DECSNLS
# SelectNumberOfLinesPerScreen <Pn>
# Pn = Number of lines
SelectNumberOfLinesPerScreen ( )
{
    print -n "${CSI}$1*|";
}

# DECSASD
# SelectActiveStatusDisplay <Ps>
# Ps = 0 - Main display
# Ps = 1 - Status line
SelectActiveStatusDisplay ( )
{
    print -n "${CSI}$1\$}";
}

# DECSSDT
# SelectStatusLineType <Ps>
# Ps = 0 - None
# Ps = 1 - Indicator (the default)
# Ps = 2 - Host-writable
SelectStatusLineType ( )
{
    print -n "${CSI}$1\$~";
}
```

```
###########################################################
#                     VT420 reports                       #
###########################################################

# DECSC
# SaveCursorState
SaveCursorState ( )
{
    print -n "${ESC}7";
}

# DECRC
# RestoreCursorState
RestoreCursorState ( )
{
    print -n "${ESC}8";
}

###########################################################
#           Resetting and testing the terminal            #
###########################################################

#################################
# Resetting and testing sequences

# DECSTR
# SoftTerminalReset
SoftTerminalReset ( )
{
    print -n "${CSI}!p";
}

# RIS
# HardTerminalReset
# Not recommended
HardTerminalReset ( )
{
    print -n "${ESC}c";
}

# DECSR
# SecureReset <Pr>
# Pr is any number from 0 to 16383
SecureReset ( )
{
    print -n "${CSI}$1+p";
}

# DECSRC
# SecureResetConformation <Pr>
# Pr is any number from 0 to 16383
SecureResetConformation ( )
{
    print -n "${CSI}$1*q";
}

# TBC
# TabulationClear h|l
# h = Clear tab at cursor position
# l = Clear all tabs
TabulationClear ( )
{
    if [[ ".$1" = ".h" ]]; then
        print -n "${CSI}0g";
    else
        print -n "${CSI}3g";
    fi
}

# DECALN
# ScreenAlignmentDisplay
ScreenAlignmentDisplay ( )
{
    print -n "${ESC}8";
}

# DECTST
# InvokeConfidenceTest <Ps1..Psn>
# Ps = 0 - All tests
# Ps = 1 - Power-up self-test
# Ps = 2 - RS-232 port data loopback
# Ps = 3 - Printer port loopback
# Ps = 6 - RS-232 modem control line loopback
# Ps = 7 - DEC-423 port loopback
# Ps = 9 - Repeat tests
InvokeConfidenceTest ( )
{
    print -n "${CSI}4;"
    OutputSequence $*;
    print -n "y";
}

###########################################################
#                    Session management                   #
###########################################################

# EnableSessions
EnableSessions ( )
{
    print -n "${CSI}&x";
}

###########################################################
#                  Constant definitions                   #
###########################################################

SET=h
RESET=l
ON=$SET
OFF=$RESET
COUPLED=$SET
UNCOUPLED=$RESET

# Operating Levels
VT100=1;
VT200=2;
VT300=3;
VT400=4;

Seven_Bit=1
Eight_Bit=2

# Visual character attribute values
AllAttributesOff=0;
Bold=1;
Underline=4;
Blinking=5;
ReverseVideo=7;
BoldOff=22;
UnderlineOff=24;
BlinkingOff=25;
ReverseVideo_Off=25;

# Editing attributes
CursorToEnd=0;
BeginningToCursor=1;
CompleteLine=2;
CompleteDisplay=2;
CanErase=0;
CannotErase=1;

# Screen display control sequence parameters
LocalEchoOff=$SET;
LocalEchoOn=$RESET;
LightBackground=$SET;
DarkBackground=$RESET;
SmoothScroll=$SET;
JumpScroll=$RESET;
MainDisplay=0;
StatusLine=1;
```

```
None=0;
Indicator=1;
HostWritable=2;

# The control code sequences for 7-bit characters
NUL="\0000";
SOH="\0001";
STX="\0002";
ETX="\0003";
EOT="\0004";
ENQ="\0005";
ACK="\0006";
BEL="\0007";
BS="\0010";
HT="\0011";
LF="\0012";
VT="\0013";
FF="\0014";
CR="\0015";
SO="\0016";
SI="\0017";
DLE="\0020";
DC1="\0021";
DC2="\0022";
DC3="\0023";
DC4="\0024";
NAK="\0025";
SYN="\0026";
ETB="\0027";
CAN="\0030";
EM="\0031";
SUB="\0032";
ESC="\0033";
FS="\0034";
GS="\0035";
RS="\0036";
US="\0037";
SP="\0040";
DEL="\0177";

IND="${ESC}D";
NEL="${ESC}E";
SSA="${ESC}F";
ESA="${ESC}G";
HTS="${ESC}H";
HTJ="${ESC}I";
VTS="${ESC}J";
PLD="${ESC}K";
PLU="${ESC}L";
RI="${ESC}M";
```

```
SS2="${ESC}N";
SS3="${ESC}O";
DCS="${ESC}P";
PU1="${ESC}Q";
PU2="${ESC}R";
STS="${ESC}S";
CCH="${ESC}T";
MW="${ESC}U";
SPA="${ESC}V";
EPA="${ESC}W";
SOS="${ESC}X";
DECID="${ESC}Z";
CSI="${ESC}[";
ST="${ESC}\\";
OSC="${ESC}]";
PM="${ESC}^";
APC="${ESC}_";
```

Arif Zaman
DBA/System Administrator
High-Tech Software Ltd (UK)

Report script for administrators

This month's instalment concludes this article on a report script for administrators.

The script **look_all** calls **look_all.ctrlc** to establish whether the user has used the key sequence <Ctrl> + C or <Ctrl> + D to abort.

LOOK_ALL.CTRLC

```
# Name script    : /home/oper/look_all.ctrlc
# Description    : Detect  <CTRL> C or <CTRL>D used in  look_all
# -------------------------------------------------------------
# Set special output.
bold=`tput smso`
offbold=`tput rmso`
under=`tput smul`
offunder=`tput rmul`
echo "\n\nProces ${under}really${offunder} stop ?
➤  (Y/${bold}N${offbold}) \c"
```

```
# Set variabele to uppercase.
typeset -u ans
read ans
# Replace previous string and output on screen
echo "\0215Really stop proces ${under}${offunder}?
➤  (Y/${bold}N${offbold})\c"
if [ "$ans" = "Y" ]
    then
        # Create file to show the user really requested termination
        >/tmp/look_all.ans.$1
        echo " Yes"
    else
        echo " No"
fi
```

Below is the help file that accompanies the utility.

```
Look_all is a utility that enables you to:
    * Check mail
    * Check filesystems
    * See the status of databases
    * See the results of saves
    * Check for space problems
    * View this help file.
The available options are (in order) mail, dfi, db, save, space,
and '?' (help).
Except for help, no parameters are required.
Examples:     look_all
              look_all dfi save



                            (More to come)

Notes:

-------------------------------  mail  ----------------------------

Before mail starts, a line displays the sender of the mail.
You can use n<Enter> or [number of the message]<Enter>
to look at the mail.
Use x<ENTER> to stop.

Futher description of mail or actions that have to be made.
.
.

The file /home/data/look_all.allservers contains all servers that send
mail to this machine.
```

```
                            (More to come)

-------------------  dfi  ( use of file-systems )  -----------------
The default threshold for filesystems is 90% free space. If this
is breached, action has to taken. This can be:
    * Expanding the filesystem
    * Cleaning up the filesystem.
If no action is taken, all free space eventually becomes exhausted
and processes may crash.

This option produces two types of output:
    * A list of filesystems on which action is necessary.
    * A list of filesystems that are skipped and on which no action
      is necessary. Examples include databases, CD-ROMs, etc.
      These filesystems are static, either because they don't grow
      or because no action is necessary even if they do grow (for
      instance, a database, a software installation etc).

The filesystems below are always skipped:
    orasys                   (Oracle installation files)
    /dev/cd0                 (the standard CD-ROM)
    /infocd                  (information CDs)
    Starting with ora_       (database files)    (More to come)

------------  dfi  ( use of file-systems )  ------------ continuing -

    Starting with /db        (database files)

The file /home/data/look_all.skipfs.[srv], where [srv] is the name of
the server, contains all filesystems that are to be skipped other than
the standard ones that are skipped.

----------------------  db   ( databases ) -----------------------

The first line is the site on which the databases are running.
If a database has a status other than active, then action needs to
be taken on that database.


The file /home/data/look_all.dbservers contains all servers that
are to be checked.



                            (More to come)
 ------------------------------  save  ----------------------------

This part of the script look_all is for examining a summary of the
save log.
There are two options:
    - The save is started.
    - The save isn't scheduled (only for Saturday), which is not a
      problem and doesn't have to be reported.

If any save went wrong (at the end of the return code is 0 or the
save hasn't ended) action is required.


The file /home/data/look_all.allservers contains all servers that
are to be checked.





                            (More to come)

---------------------  space ( space problems )  ------------------

This part of the look_all script checks sites where you expect
space problems in future.

You can examine:
    - Space in the rootvg volume group.
      This volume group may request space if you restore a BosBoot
      tape and the restore needs to expand the /tmp filesystem.
    - Space in other volume groups.
      This space may be needed for a variety of reasons:
      (in addition to the ones below, others can be added)
          * A filesystem has to be increased in size
          * A database needs more space
          * Software has to be installed.

If the filesystem hasn't enough free space (10%), the script shows
the filesystems involved (rootvg and others).



                            (End of help)
```

FORMATS OF FILES

/HOME/DATA/LOOK_ALL.*

```
* /home/data/look-_all.xxxservers:
server1
server2
...
```

Example: *look_all.dbservers* (servers that support one or more databases).

```
afts1
afts2
afts7
bgns1
ecodet
rlts1
sons1
```

/HOME/DATA/LOOK_ALL.SKIPFS.XXXXXX:

```
file_system1 |  file_system2 | filesystem3 |   ......
```

Example: *look_all.skipfs.rlts1* (filesystems to skip at site *rtls1*).

```
/dev/data |
```

/HOME/DATA/LOOK_ALL.SPACE.XXXXXXX

```
<action that has been taken>
```

Example: *look_all.space.elss1* (action to take when potential space problems are detected at site *elss1*).

```
Action has been taken. A new disk has been ordered and will
be added to the server. The disk will be delivered in two weeks.
```

Example of screen output:

```
Part : check_sites:
    *****************************************************
    *     No information received ( mail ) of:          *
    *                                                   *
    *                --> bgns1                          *
    *                                                   *
    *         Inform the administrator                  *
    *****************************************************

Part : mail

    *****************************************************
    *     There is no mail for root in the next sites:  *
    *                  aftopt                           *
    *                  afts1                            *
    *                  afts2                            *
    *                  afts3                            *
    *                  afts7                            *
    *                  ecodet                         *
    *                  elss1                          *
    *                  gtns1                          *
    *                  kbds1                          *
    *                  netview                        *
    *                  rlts1                          *
    *                  sons1                          *
    ***************************************************

Mail van bgns1:


Mail [5.2 UCB] [AIX 4.1]  Type ? for help.
"/var/spool/mail/root": 2 messages 2 new
>N  1 root         Mon Dec 21 06:30  22/795  "bgns1 epsyscheck_log"
 N  2 root         Mon Dec 21 07:00  11/341  "bgns1 daily log."
?
 Part: use of file-systems:
    ***************************************************
    *   The free space in the file-systems is ok for:   *
    *                  aftopt                         *
    *                  afts4                          *
    *                  afts7                          *
    *                  bgns1                          *
    *                  ecodet                         *
    *                  elss1                          *
    *                  gtnigb                         *
    *                  gtns1                          *
    *                  hrns1                          *
    *                  hvns1                          *
    *                  kbds1                          *
    *                  netview                        *
    *                  rlts1                          *
    *                  sons1                          *
    *                                                 *
    ***************************************************

Filesystems of gtns1, which are using more than 90%:

Filesystem               Total Kb  Free Kb  %used
/epochdata                  61440    40680   99%


Skipped filesystems in afts1:
/infocd
/orasys
/ora_ctrlog
/ora_db9902
/ora_IEFITDB
/ora_db9908
```

```
/db9901_01
/ora_db9906

Part: databases:

Servername: afts2
Database db9905     status: active
Database ABSDB      status: ?????? INFORM THE DATABASE ADMINISTRATOR
Database ABSDB3     status: active
Database DBIEFD     status: active
Database db9907     status: active
Database dbtest     status: active

Part: save:

Save of: afts2:

10/11/98 20:30:01 root     /home/oper/dagtape1 - gather_saves - Begin
10/11/98 21:08:30 root     /home/oper/dagtape1 - gather_saves - Create
save1 has an returncode 0
10/11/98 21:08:56 root     /home/oper/dagtape1 - gather_saves - End
10/11/98 22:00:01 root     /home/oper/dagtape2 - gather_saves - Begin
10/11/98 22:01:52 root     /home/oper/dagtape2 - gather_saves - Create
save2 has an returncode 3
10/11/98 22:50:28 root     /home/oper/dagtape2 - gather_saves - Eind
10/11/98 23:00:00 root     /home/oper/dagtape3 - gather_saves - Begin
11/11/98 01:22:45 root     /home/oper/dagtape3 - gather_saves - End

Part: space:

    *********************************************************
    *      Space problems detected on machine: elss1        *
    *                 rootvg      : 24 Mb                   *
    *                                                       *
    *********************************************************

    Action has been taken. A new disk has been ordered and will be
    added to the server. The disk will be delivered in two weeks.
```

Teun Post
Unix Specialist
Schuitema NV (The Netherlands)

SSA disk problems

This article concerns SSA disks in normal configuration (that is, not in SSA RAID or multi-initiator configuration), and examines ways of reducing risks arising from unresolved problems.

With this type of disk subsystem installed, you may occasionally encounter entries like the ones below in an **errpt** report.

```
8BDD5B42   0424043099 T H pdisk53        DISK OPERATION ERROR
8BDD5B42   0421040399 T H pdisk53        DISK OPERATION ERROR
8BDD5B42   0416052399 T H pdisk53        DISK OPERATION ERROR
8BDD5B42   0416040099 T H pdisk53        DISK OPERATION ERROR
```

This indicates a problem, so carry out a disk certification test. If the test is positive, then no bad clusters were found, which could indicate a temporary fault. In this case, keep an eye on the **errpt** report for further errors.

PREREQUISITES

The procedure outlined in this article requires the following:

- *root* access to the system.
- Access the SSA disk affected by the problem.
- Either mirrored logical volumes or an additional hard disk.
- Documentation about volume groups, logical volumes, filesystems, and disks structure at the site.

The procedure below has been tested with AIX 4.2.1.

The problem that this article addresses is essentially how to reduce the risk of failure following reported errors while maximizing overall availability. The steps that are necessary to achieve this are:

1 Discovering the affected logical disk.

2 Checking the logical volumes on it.

3 Establishing the right repair procedure.

4 Executing the procedure.

THE STEPS IN DETAIL

1 *Discover the affected logical disk*

Run the command **ssaxlate** using the physical disk name from the **errpt** output. For example:

```
ssaxlate -l pdisk53
```

This reveals the logical disk name of the disk referred to in the report. In our example, this is *hdisk43*. The number of the logical disk may be the same as the physical disk, but this is not necessarily the case.

The command below displays all the logical volumes affected. In our example, they are: *lvm_ab* and *lvm_cd*, as shown by the output of the command.

```
lspv -l hdisk43

hdisk43:
LV NAME       LPs   PPs   DISTRIBUTION         MOUNT POINT
lvm_ab        2     2     02..00..00..00..00   /fs_ab
lvm_cd        100   100   00..27..26..27..20   /fs_cd
```

2 *Check the logical volumes on the affected disk*

Run **lslv -l** for each *lvm*.

In our example the commands to run are:

```
lslv -l lvm_ab
```

and

```
lslv -l lvm_cd
```

This shows you the distribution of logical volumes. After running these command you'll be able to ascertain whether the logical volumes are mirrored. Take a look at the field *COPIES* in the first column – if the value shown in this field is greater than '1', the *lvm* is mirrored.

```
lslv -l lvm_cd

lvm_cd:/fs_cd
```

```
PV            COPIES         IN BAND   DISTRIBUTION
hdisk43       100:000:000    27%       000:027:026:027:020

lslv  lvm_cd

LOGICAL VOLUME:   lvm_cd               VOLUME GROUP:   vg_zta
LV IDENTIFIER:    0008823de8a56796.1   PERMISSION:     read/write
VG STATE:         active/complete      LV STATE:       opened/syncd
TYPE:             jfs                  WRITE VERIFY:   off
MAX LPs:          512                  PP SIZE:        16 MBs
COPIES:           1                    SCHED POLICY:   parallel
LPs:              100                  PPs:            100
STALE PPs:        0                    BB POLICY:      relocatable
INTER-POLICY:     minimum              RELOCATABLE:    yes
INTRA-POLICY:     middle               UPPER BOUND:    32
MOUNT POINT:      /fs_cd               LABEL:          /fs_cd
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

3 *Establish the right repair procedure*

In this step we have two choices:

If the logical volumes are not mirrored and the error arises frequently, we have to react as quickly as possible, as there is a risk of data loss should the problem worsen. We could even lose the whole logical volume and be forced to recover from the last back-up.

On the other hand, if all logical volumes affected are mirrored, then there is no urgency in dealing with the problem. In this case, go to step ‘4b’ below.

4 *Execute the repair procedure*

a If you have a spare disk to hold the physical partitions of the logical volumes, then add the physical volume to the volume group of the affected SSA disk. If not, then document the entire volume group, logical volume, and filesystem structure. Back up all data, delete the logical volumes, replace the defective disk, recreate the old *lvm*s and filesystems, and restore the data. To carry out this step, you don’t have to mirror your logical volumes, though you do lose system availability for one or two hours (a discussion of this is beyond the scope of this article).

The output of the **lslv lsvm_cd** command shows that this logical volume belongs to the volume group *vg_zta* (as shown by the value of field 'VOLUME GROUP').

Take a look at the new *hdisk* by issuing the following command:

```
lspv hdisk11
```

This is our new interim disk – if the output of the command looks like this:

```
0516-320 : Physical volume 004001738c45594d is not assigned
           to a volume group.
```

then the *hdisk* is not a member of another *vg* and the disk is available for use, so add it to *vg_zta*:

```
extendvg vg_zta hdisk11
```

For each logical volume on the old problem disk we have to **mklvcopy** and **syncvg**:

```
mklvcopy lvm_ab 2 hdisk11
mklvcopy lvm_cd 2 hdisk11
syncvg -l lvm_ab
syncvg -l lvm_cd
```

b This is the entry point if we have already mirrored our data, which assumes that all affected logical volumes are mirrored on another physical hard disk.

Check the mirror with:

```
lsvg -l vg_zta
```

Our two sample logical volumes should have the status 'open/synced' – if not, then do a **syncvg -l lvm_name** on the 'stale' logical volumes. Then remove the logical volumes from the old problem disk using the command:

```
rmlvcopy lvm_ab 1 hdisk43
rmlvcopy lvm_cd 1 hdisk43
```

After issuing the command **lspv -l hdisk43**, we should see no logical volumes on this disk. By issuing the command:

```
reducevg vg_zta hdisk43
```

the volume group *vg_zta* is removed from the problem disk. After issuing the command **lspv hdisk43**, we should get the message: *0516-320: Physical volume is not assigned to a volume group*. If we don't receive this message, then there are still logical volumes on the *hdisk* and we have either to delete them (if they are mirrored) or migrate them to another disk.

Now that the disk for our test procedure is isolated, carry out the following procedure:

1 Certify the disk:

If media-related problems occur, the following command will try to reassign soft error blocks:

```
ssa_certify -l pdisk53
```

The output of the **ssa_certify** command is '0' unless a non-media-related problem occurs. If a non-media-related problem occurs, the command sends a message to *stderr*. To check the return value, use the command **echo $?** immediately after the command returns.

If the attempt to reassign soft error blocks fails, or if the block has a hard media error, then the **ssa_certify** command returns '0' and also prints the LBA of the failing block to *stdout*, followed by the word 'Failed'.

If the operation to certify the disk is successful, the **ssa_certify** command returns no output.

2 The second alternative is to diagnose the disk.

The **ssa_diag** command is located in */usr/lpp/diagnostics/bin*. Issue the command as follows:

```
ssa_diag -l pdisk53
```

If an error occurs, the **ssa_diag** command generates an error message, such as:

```
ssa2 SRN 42500
```

and sends it to *stdout*. If no error occurs, the command sends no output. A non-zero return code indicates an error, which is accompanied by an error message to *stdout*. Note the use of optional flags with this command:

```
ssa_diag -l pdisk53 -s
```

This requests the output of the power status, which could be:

0 – disk power is good.

1 – disk has lost redundant power.

2 – disk has lost power.

3 We have now diagnosed that the disk really has a hardware problem, so call the hardware maintenance contract desk to send a new SSA disk!

Before physically removing the hard disk, remove the physical and logical disk from the system configuration:

```
rmdev -l pdisk53 -d
rmdev -l hdisk43 -d
```

To find the right disk to remove, follow the steps below.

Before you physically replace the SSA disk, check it using **umayf**, which is part of the **diag** menu, though you can invoke it directly from */usr/lpp/diagnostics/bin* – below is the screen selection.

```
SSA SERVICE AIDS
Move cursor to the required option, then press <Enter>.
  Set Service Mode
  Link Verification
  Configuration Verification
  Format Disk
  Certify Disk
  Display/Download Disk Drive Microcode
```

Select ‘Link Verification’ and you get an output like this:

```
LINK VERIFICATION
SSA Link Verification for:
osiris:ssa0          04-06   IBM SSA Enhanced RAID Adapter

To Set or Reset Identify, move cursor onto selection, then
press <Enter>
```

```
Physical            Serial#     Adapter Port
                                A1  A2  B1  B2      Status
osiris:pdisk0       294E1672     0   3              Good
osiris:pdisk3       294E2589     1   2              Good
osiris:pdisk5       29C85600     2   1              Good
osiris:pdisk53      29C8583B     3   0              Good
osiris:pdisk6       29C857E9             0   3      Good
osiris:pdisk4       29C855FC             1   2      Good
osiris:pdisk1       294E1787             2   1      Good
osiris:pdisk2       294E23CA             3   0      Good
```

Check that the rest of the loop is still closed, otherwise you'll encounter substantial problems with the remaining active data connections, perhaps leading to parts of the loop becoming isolated.

4 To identify the SSA disk in the cabinet, use the **umayf** option above or the command:

```
ssaidentify -l pdisk53 -y
```

which puts the disk in 'identify' mode. While the disk is in this mode, its amber *Ready* light flashes approximately once a second. The **-n** flag turns off the mode, but if you replace the disk, you don't need to do this.

SSA devices can be maintained concurrently. In other words, they can be removed, installed, and tested on an SSA loop while other devices on the loop continue to work normally. If a disk drive has its *Check* light on, you can remove that disk drive from the SSA loop without taking any preparatory measures.

5 Replace the disk that has its service indicator light on with the new disk.

- Run the **cfgmgr** command.
- Check that the new SSA drive has been detected using **lspv**.
- Run the **umayf** command again to check the connectivity of the loop.
- Add the disk to the volume group.

```
extendvg vg_zta hdisk43
```

- Recreate the logical volumes (or their copies) on the disk using the commands:

  ```
  mklvcopy lvm_ab 2 hdisk43
  mklvcopy lvm_cd 2 hdisk43
  syncvg -l lvm_ab
  syncvg -l lvm_cd
  ```

- Update the system documentation to include the disk replacement.

CONCLUSION

To achieve high system availability, all critical logical volumes should be mirrored. It is helpful to have a spare disk available so that defective disks can be replaced quickly.

Michael Imhotep
AIX System Administrator (Australia)

Saving volume group information

Have you ever suffered a disk crash or other disaster that resulted in lost data? You can normally recover data from a back-up quite easily, but what about recreating the volume group structure? Have you got all the relevant volume group information to hand? The sort of information you need is which disks your volume group resides on, the PP size, how big each logical volume is, and the filesystem mount points. In addition, you may also have your logical volumes mapped to specific regions of the disk.

The following script saves all the important information required for recreating a volume group after a disaster. It stores the information in the */var/adm/vgdata* directory, so that the information is backed up as part of your standard **mksysb**. It also creates some plain text files that are stored in the */var/adm/vgdata/${VG}* directory and can be used as reference information about your volume group and disk layout.

The script requires only one parameter, which is either a volume group name or the word ‘ALL’, which results in information about all volume groups that are currently varied on being saved.

The script has been tested at AIX Version 4.2.

To recreate your volume group on the disks from which it was backed up, use the command:

```
restvg -f /var/adm/vgdata/${VG}.backup
```

To recreate your volume group on different disks, use the command:

```
restvg -f /var/adm/vgdata/${VG}.backup hdisk5 hdisk6 hdisk7 ...
```

For more information about the **restvg** command, please refer to the AIX Info Explorer or ‘**man**’ pages.

Note the use of the continuation character, ‘➤’, to indicate that one line of code maps to several lines of print.

SAVE_VG_INFO.KSH

```
#!/usr/bin/ksh
#
# Script: save_vg_info.ksh
#
# Author: Steve Diwell - Jedi Technology Ltd.
#
# Date:   June 1999
#
# Aim:    Save the volume group information required to recreate a
#         volume group after a disk crash.
#
# NOTE:
#       To run this script in debug mode type:
#          export DEBUG=yes
#       on the command line before running this script.
#
#       To turn off debug mode type:
#          unset DEBUG
#       on the command line before running this script.
#

#
# Function to display the scripts usage
#
```

```
usage()
{
[[ -n ${DEBUG} ]] && set -x
clear

cat <<EOF

This script saves information required to recreate a volume group
after a disaster or disk crash. The volume group information is
stored in the "${SAVEDIR}" directory.

You may specify 'ALL' as the volume group name, in which case the
script saves information on all volume groups that are currently
on-line.

This script will not create the information on the rootvg
filesystem.

NOTE: You must be the root user to run this script.

Usage: ${SCRIPT} VolumeGroup|ALL

EOF

exit 1
}

#
# Some commands take a while to run and may produce no output, so
# this function puts a . on the screen every five seconds.
#
progress()
{
while :
do
    echo ".\c"
    sleep 5
done
}

#
# Are we in debug mode?
#
[[ -n ${DEBUG} ]] && set -x
clear

#
# Set environment for the script
#
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin:
```

```
SCRIPT=`basename ${0}`
SAVEDIR="/var/adm/vgdata"
VG_INPUT=$*

#
# Only root can run the script.
#
[[ `whoami` != "root" ]] && usage

#
# If user input is null or rootvg, display usage.
# If user input is ALL, do all volume groups except rootvg.
#
[[ -z ${VG_INPUT} ]]      && usage
[[ ${VG} = "rootvg" ]]    && usage
[[ ${VG_INPUT} = "ALL" ]] && VG_INPUT=`lsvg -o | grep -v rootvg`

#
# For every volume group listed, save information
#
for VG in `echo ${VG_INPUT}`
do
   echo "\nWorking on volume group ${VG}..."

#
# Let's check the ${VG} is on-line
#
   lsvg -o | grep -q ${VG} || {
      echo "\nThe Volume Group ${VG} is not on-line on this system."
      break
      }

#
# Get all the disks the ${VG} volume group is on.
#
   DISKS=`lsvg -p ${VG} | awk '/active/ { ORS=" "
                          print $1}'`

#
# Create the Exclude file to stop the savevg backing up any files
#
   [[ -f /etc/exclude.${VG} ]] && mv /etc/exclude.${VG}
   ➤  /etc/exclude.${VG}.saved

   echo "/" > /etc/exclude.${VG}

#
# Now, mount all the filesystems in ${VG}, otherwise, savevg will
# miss them!!
#
```

```
   echo "\nEnsuring all filesystems in the ${VG} are mounted,
   ➤  please wait."
   progress &
   PID=$!

   lsvg -l ${VG} | awk '/closed/ {
      if ( $2 == "jfs" )
         { if ( system ("mount " $7 ) != 0 )
            print "Mount command failed for " $7
         }
      }'

   echo ""
   kill ${PID} 1>/dev/null 2>&1

#
# Create volume group information
#
   echo "\nCreating the volume group data, please wait..\c"
   progress &
   PID=$!

   mkvgdata -m ${VG} || {
      echo "\nMkvgdata Command Failed for ${VG}"
      kill ${PID}
      exit 1
      }

   echo ""
   kill ${PID} 1>/dev/null 2>&1

#
# Backup the ${VG} data file and put copy in ${SAVEDIR}
#
   [[ ! -d ${SAVEDIR}/${VG} ]] && mkdir -p ${SAVEDIR}/${VG}

   cp /tmp/vgdata/${VG}/${VG}.data ${SAVEDIR}/${VG}/${VG}.data

   echo "" > ${SAVEDIR}/${VG}/${VG}.disks

   for PV in `echo ${DISKS}`
   do
      lspv -l ${PV} >> ${SAVEDIR}/${VG}/${VG}.disks
      echo ""       >> ${SAVEDIR}/${VG}/${VG}.disks
      lspv ${PV}    >> ${SAVEDIR}/${VG}/${VG}.disks
      echo ""       >> ${SAVEDIR}/${VG}/${VG}.disks
   done

#
```

```
# Save the volume group structure, which is used by restvg
#
   echo "\nSaving the volume group structure, please wait..."

   savevg -ef ${SAVEDIR}/${VG}.backup ${VG} || {
      echo "Savevg command failed for ${VG}"
      exit 1
      }

   echo "\nInformation required to recreate the ${VG} volume group"
   echo "has been saved in plain text files in \"${SAVEDIR}/${VG}\"."

   echo "\nThe command \"restvg -f ${SAVEDIR}/${VG}.backup\""
   echo "can be used to recreate the volume group.\n"

#
# Put back the original exclude file, if it existed.
#
   if [[ -f /etc/exclude.${VG}.saved ]]
     then
         mv /etc/exclude.${VG}.saved /etc/exclude.${VG}
     else
         rm -f /etc/exclude.${VG}
   fi

#
# Clean up temporary files and directories
#
   [[ -f /tmp/${VG}.backup ]]        && rm -f /tmp/${VG}.backup
   [[ -f /tmp/vgdata/vgdata.files ]] && rm -f /tmp/vgdata/vgdata.files
   [[ -d /tmp/vgdata/${VG} ]]        && rm -fr /tmp/vgdata/${VG}

done            # For the for VG in `echo ${VG_INPUT}`

exit 0
```

Steve Diwell
Senior Consultant
Jedi Technology Ltd (UK)

AIX news

IBM has announced some new capabilities for MQSeries 5.1 for AIX, the main ones being support for XML, Java APIs, a new MQSeries APIs, and new versions of the MQSeries Integrator message broker and MQSeries Workflow.

The new APIs include the Application Messaging Interface (AMI), which moves message handling code from the application into the middleware, so applications can focus on business logic. Java Message Service (JMS) API is the foundation services API for message queueing in the IBM Application Framework for e-business. Applications written to it can communicate with applications written to the Message Queue Interface (MQI) and the AMI.

The new features will be available on MQSeries Version 5.1 for AIX in Q3 1999.

For further details contact your local IBM representative.

* * *

SCO this week announced Tarantella 1.4, a Web-enabling package whose new features include faster development and customization of intranet and extranet portals and a ‘follow-me’ printing facility that redirects print requests to the most appropriate place.

IBM is to ship Tarantella on AIX to its own customers. Available now, Tarantella also runs on HP-UX, Solaris, Siemens Reliant Unix, UnixWare 7, and SCO UnixWare system and costs US$395 per concurrent user.

For further information contact:
SCO, Encinal Street, Santa Cruz, CA 95060, USA
Tel: +1 831 425 7222
Fax: +1 831 458 4227
Web: http://www.sco.com

SCO, Conqueror House, Vision Park, Cambridge CB4 9ZR, UK
Tel: +44 1223 518000
Fax: +44 1223 518001

* * *

Merant, formerly Micro Focus and Intersolv, has announced PVCS Version 6.5, the company’s software configuration management package. New are a browser-based client, HTML-based help system, and hierarchical project structure. The new Version supports AIX, Windows, Solaris, and HP-UX. It’s out now, and prices start at US$650.

For further information contact:
Merant, 2465 E Bayshore Road, Palo Alto, CA 94303, USA
Tel: +1 650 856 4161
Fax: +1 650 856 3724
Web: http://www.microfocus.com

Merant, Speen Court, 7 Oxford Road, Newbury, Berks, RG14 1PB, UK
Tel: +44 1635 32646
Fax: +44 1635 33966

xephon