46

AIX

update

August 1999

In this issue

Published by

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

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.

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).

Printed in England.

Integrating Windows NT with AIX

While Windows NT proliferates as a client and low-end server, Unix systems, such as IBM's AIX, remain the choice for technical workstations and high-end servers.

Given that NT and Unix each has its own advantages and disadvantages, many installations will choose to run mixed environments that include AIX servers, NT clients, and even NT servers, as well as NetPCs whose server operating system may well be AIX. Administrators are then left with the problem of integrating NT and AIX.

Cost is an important factor in choosing an enterprise solution. AIX offers better high-end server performance than NT and enables you to build clusters of up to 32 servers using HACMP and SP servers. On the other hand, Windows NT is part of the PC market, where up-front costs are usually significantly lower than in the Unix workstation market.

Companies that need high performance and also want to roll out workstations with a user-friendly interface and security features often implement AIX servers and Windows NT workstations. This article presents various options for this type of integration, concentrating on options delivered by IBM. However, I won't discuss NFS solutions offered by a variety of vendors.

INTEGRATION

In order to integrate AIX and NT, you'll probably need additional software, and this can require additional expertise. If you want, for example, to share data between AIX and NT, you have to make the file system from one operating system available to the other.

The standard packages do not support common network file systems and, therefore, you must add the appropriate software. The same is true if you intend to coordinate user and group information for both systems to preserve file system permissions and security.

TOTALNET ADVANCED SERVER (TAS)

TAS is from Syntax Inc, and is included in the AIX 4.3 Bonus Pack. The package allows an AIX system to act as file, print, or application server to PC clients in a multi-platform environment.

The client system can be Windows NT, Windows 95/98, Windows for Workgroups, MS-DOS, Apple Macintosh, or OS/2 Warp. Each client presents the familiar desktop interface (see Figure 1) while the AIX server acts in the background.

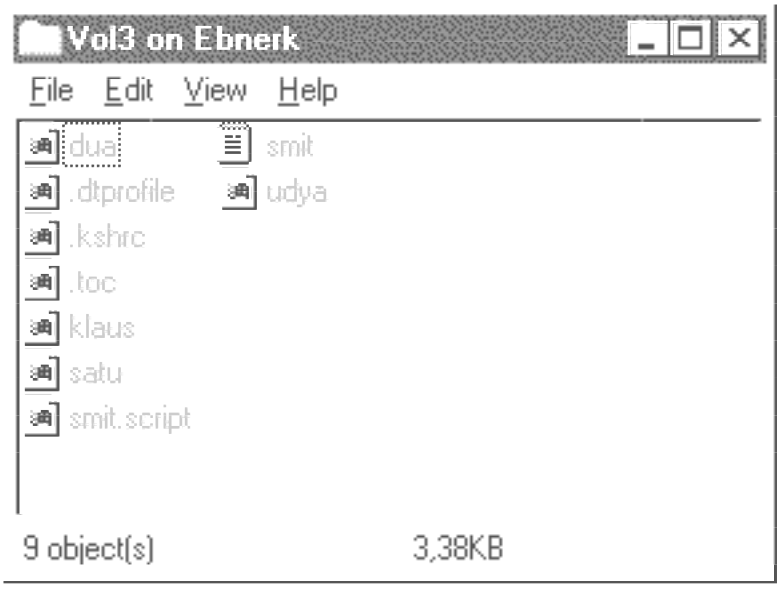

Figure 1: Accessing AIX filesystems using NT's Explorer

Note that the AIX Bonus Pack's version of TAS allows you to connect only one client to the TAS server – the full product is available from Syntax (see *http://www.syntax.com*).

In order to run TAS, you need AIX 4.1.4 or later and the following filesets: *bos.net.tcp.server*, *bos.txt.tfs*, *bos.rte.streams*, *bos.rte.tty*, and *bos.data*.

You can configure and administer TAS using a Web browser that supports tables, forms, Java, and JavaScript. When TAS is installed, start your Web browser and enter the following URL: *http://<hostname>:7777*. The browser then connects to TAS, and the TotalNET Administration Suite (TNAS) lets you log in.

TAS is divided into three ‘realms’. A realm is a part of the system that enables TAS to serve different kinds of client. The three realms are the ‘LM-NT-OS/2’ realm, the NetWare realm, and the AppleTalk realm.

The LM-NT-OS/2 realm supports native Windows networks using the SMB networking protocol and the NetBIOS interface.

The only other requirement is that the client must have access to the network. In the case of my own installation, Windows NT is a fully functional client and no additional software is needed.

AIX CONNECTIONS

IBM’s AIX Connections is based on the TotalNET Advanced Server. Like TAS, AIX Connections has different realms that allow the package to communicate with different types of client. AIX systems can act as file, print, or application servers for NetBIOS, NetWare, and Macintosh clients.

Software requirements are AIX 4.1.5 or higher, *bos.net.tcp.server* and *bos.data*, NetBIOS 2.1.4 or higher, and a Web browser that supports forms. Prior to installing AIX Connections you must install NetBIOS, which ships with the software.

You can install AIX Connections using **smit**, Web-based system management, or the command line. AIX Connections automatically creates a file service, a terminal service, and volume references to the home and *pccode* volumes.

As with TAS, the client machines don’t have special pre-requisites to meet in order to connect to AIX Connections. The LAN Manager part of Windows NT suffices.

Configuration and administration of AIX Connections can be done using the command **smit aconn** or via a Web browser by connecting to URL *http://hostname:7777* where *hostname* is your machine’s hostname.

The Quick Start option allows you to start with a default NetBIOS server configuration needed by Windows NT.

As there is no client software on the Windows NT machine, user

administration is in fact doubled. AIX Connections does not act as Windows NT server, and your clients can belong to an NT domain or workgroup. If the user name and password are right, the user can access resources for which he or she has permission.

In general AIX Connections is used as a file and print server that allows you to access AIX resources from a Windows NT machine. However, it is possible to use the software for access in the other direction. In this case, the AIX system is the client while Windows NT is the server.

In order to access NT volumes from an AIX machine you need to log on as *root* and to type **smit aconn**. If you have a user account, you can log into a Windows NT server via 'Client-Login/Logout'. Then select 'Mount/Unmount Volume' to mount the Windows NT drive as a volume on AIX. You also have to indicate the realm, which normally is NetBIOS (NB).

SAMBA

SAMBA is a freeware package developed by Andrew Tridgell in 1990. Today SAMBA is used widely for sharing resources and is available on numerous platforms.

SAMBA is primarily an SMB server that lets you share files and printers. Additionally, it allows AIX systems to access resources on other SMB servers. The software can act as a WINS server and participate in the Windows browsing mechanism.

SAMBA is available from *http://samba.anu.edu.au*. You can download it for free. If you don't want to recompile the source code, you can use one of the ready-to-use pre-compiled versions of SAMBA. Usually, you uncompress the file and use the **tar** command to install the binaries on your AIX machine.

Different client systems can be used with SAMBA. Among them are Windows NT, Windows 95, OS/2 Warp Connect, and OS/2 Warp 4. None of them needs any additional software as only TCP/IP is needed to access the SAMBA server.

SAMBA can be administered from the command line with the aid of

a text editor. Fortunately no complex procedures are involved. You need only edit one file, *smb.conf*, which is usually located in */usr/local/samba/lib/*.

ADVANCED SERVER FOR UNIX (AS/U)

AS/U is Group Bull's implementation of AT&T's networking operating system. It provides AIX with networking functionality equivalent to Windows NT Server. With AS/U installed, an AIX system acts as a server for PC clients using the SMB protocol. The software doesn't require any additional code on the client systems, which aren't even aware that they are being served by AS/U and not Windows NT.

The features of AS/U can be summarized as follows:

- SMB-based client support
- Logon validation
- Local and remote print serving
- File system access control
- Remote administration
- An AS/U system can act as a PDC or BDC
- AS/U supports trust relationships between domains
- An AS/U system can act as a WINS server (licensed from Microsoft)
- AS/U supports account and file replication.

As Advanced Server for Unix can be configured as either a primary or backup domain controller, the system is able to interoperate with Windows NT servers. A client or user won't see a difference between AS/U servers and NT servers as all systems look like Windows NT servers on the network. As AS/U is installed on an AIX system, this system can be used to manage Unix and PC users simultaneously.

Administering domain controllers is easy with Server Manager, which is also used for managing AIX-based domain controllers. Windows NT allows you to promote a backup domain controller

(BDC) to a primary domain controller (PDC), and demote it back to a BDC, using Server Manager on a Windows NT system. There is no difference between the promotion or demotion of a Windows NT server and an AIX server with AS/U installed. If you don't possess NT Server but only Workstation, Server Manager can be copied from the AS/U CD-ROM.

AS/U is delivered on a CD-ROM called 'OpenTeam for AIX' that contains the Advanced Server software, NetBIOS for Unix, some additional filesets, and the documentation. You need AIX version 4.1.1.2 or later, though Bull recommends that you install at least AIX 4.1.5; NetBIOS 3.0 for Unix, which is included on the CD-ROM, has to be installed prior to installing AS/U.

Standard Windows administration programs are used – you can use the System Policy Editor, User Manager for Domains, Server Manager, and WINS Manager. These tools are not provided with Windows NT Workstation but they come with AS/U and can be copied and installed locally. The only tool that is specific to AS/U is the AS/U Administrator Tool (**asuadm.exe**). This allows you to edit specific AS/U parameters that are stored in the AS/U registry (the parameters specify such settings as whether an AIX system account is generated whenever an AS/U account is created).

As the name suggests, the AS/U Server Administrator is used both for administering AS/U servers and for providing a launch pad for other administrative tools. These tools enable you to change the properties of the following services:

- Alerter service
- Computer browser service
- Connected clients
- File name space mapping service
- Netlogon service
- Server announcements
- Unix account mapping service
- Unix file system integration

- UPS services
- User alerts.

After installation, you can start the AS/U Administrator by typing the command **asuadm** at an NT command prompt.

The graphical interface asks you to select the server you want to administer. The version tab tells you the server's role and its operating system. You can start applications using the applications menu. By selecting the policy tab, you can change the policy settings for different items.

AS/U stores configuration information in a registry database. It's possible to open this file from Windows NT . Choose 'File', 'Select Computer', and type in the name or the IP address of the AIX machine running AS/U. Most settings that are relevant to AS/U server configuration are accessible from the following entry:

```
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/AdvancedServer
```

If you prefer the command line interface, you can use the well-known **net** command to administer AS/U. The command **net config server** provides you with much more information on an AS/U-based server than is available on real NT servers, allowing you also to modify the information.

Klaus Ebner (PSE)
Curriculum Manager for Windows NT
IBM Learning Services (Austria)

Disk space and bus architecture

After reading a short article in the February 1999 issue of *AIX Update* entitled *Memory size display* I'd like to contribute my own tips on finding the amount of usable physical memory and the bus architecture. The command:

```
# lsattr -E -l sys0 -a realmem
```

results in the following sample output:

```
realmem 131072 Amount of usable physical memory in Kbytes False
```

This works on both MCA and PCI boxes, at least starting from AIX v4.1.5.

Gathering information about system buses is equally simple – just use the following command:

```
# lsdev -Cc bus
bus0 Available 00-00 PCI Bus
bus1 Available 00-00 PCI Bus

# lsdev -Cc bus
bus0 Available 00-00 Microchannel Bus
```

The first example is from running the command on an RS/6000 model F40, the second from a model C10.

Rens Groenewegen
Unix/IT Specialist
ING Group (The Netherlands)

Disk usage report

dur.sh is a shell script that provides much information on physical volumes. It is able to display a list of values for a number of objects (system user, physical volume, mount point, etc) from which the user can select a specific value to focus on. The script can also send output to a named printer. The options available with the script are as follows:

5 Specific details of physical volumes

10 All details of the physical volume

15 Total disk usage for a specific user

20 Total disk usage for all users

25 Directory disk usage for a specific user

30 Specific logical to physical volume mapping

35 All logical to physical volume mappings

40 Specific physical to logical volume mapping

45 All physical to logical volume mapping

50 Filesystem defragmentation report

99 Exit.

Note the use of the continuation character, '➤', in the listing below to indicate that one line of code maps to several lines of print.

DUR.SH

```
#! /usr/bin/ksh
################################################################################
#                                                                              #
# dur.sh (Disk Usage Report)                                                   #
#                                                                              #
# This script calculates and displays various statistics related               #
# to disk usage.                                                               #
#                                                                              #
# History                                                                      #
#                                                                              #
# Date     Author     Description                                              #
# ------------------------------------ #
# 01/03/99  A Zaman   Initial build                                            #
#                                                                              #
################################################################################

################################################################################
#                                                                              #
# InitialiseVariables                                                          #
#                                                                              #
# This function initializes all required variables.                            #
#                                                                              #
################################################################################
InitialiseVariables ()
{

# Define temporary files
REPORT_FILE=/tmp/dur_$$.rep
TEMP_FILE_1=/tmp/dur_$$_1.tmp
TEMP_FILE_2=/tmp/dur_$$_2.tmp
TEMP_FILE_3=/tmp/dur_$$_3.tmp
ERROR_FILE=/tmp/dur_$$.err
```

```
# Define return codes
TRUE=0
FALSE=1
SEC=0
FEC=1

# Define escape sequences
ESC="\0033["
RVON= [7m             # Reverse video on
RVOFF= [27m           # Reverse video off
BOLDON= [1m           # Bold on

BOLDOFF= [22m         # Bold off
BON= [5m              # Blinking on
BOFF= [25m            # Blinking off

# Define required variables
DUK=                  # Disk usage in kilobyte blocks
DUB=                  # Disk usage in bytes
SLEEP_DURATION=3      # Number of seconds for sleep command
ERROR="${RVON}${BON}dur.sh:ERROR:${BOFF}"
INFO="${RVON}dur.sh:INFO: "

# Define menu title
DURM="${RVON}Disk Usage Report Menu${RVOFF}"

# Messages
INTERRUPT="Program interrupted! Quitting.${RVOFF}"
WORKING="Working.........${RVOFF}"
INVALID_OPTION="\${OPTION}, is an invalid option${RVOFF}"
INVALID_ENTRY="Invalid entry${RVOFF}"
INVALID_MPOINT="Invalid mount point: \${MPOINT}${RVOFF}"
INVALID_PV="Invalid physical volume: \${PV}${RVOFF}"
INVALID_LV="Invalid logical volume: \${LV}${RVOFF}"
PRINT_OK="Successfully submitted print job${RVOFF}"
PRINT_NOT_OK="Failed to submit print job${RVOFF}"
INVALID_USER="\${UNAME} is an invalid user${RVOFF}"
OS_ERROR="\${SYSERROR}${RVOFF}"
REQ_USER="Must execute the script from root account${RVOFF}"

# Define signals
SIGNEXIT=0  ;  export SIGNEXIT   # Normal exit
SIGHUP=1    ;  export SIGHUP     # Session disconnected
SIGINT=2    ;  export SIGINT     # Ctrl-c
SIGTERM=15  ;  export SIGTERM    # "kill" command
}

#############################################################################
#                                                                           #
# HandleInterrupt                                                           #
```

```
#                                                                              #
# This function calls ProcessExit.                                            #
#                                                                              #
################################################################################
HandleInterrupt ( )
{
DisplayMessage I "${INTERRUPT}"

ProcessExit $FEC
}

################################################################################
#                                                                              #
# MoveCursor                                                                   #
#                                                                              #
# This function moves the cursor to the required (X, Y) location.             #
#                                                                              #
# Input   : Y and X coordinates                                                #
#                                                                              #
# Notes   1   The function must run in ksh for print to work.                  #
#                                                                              #
#         2   The print command must be used to move the cursor                #
#             as echo seems not to work.                                       #
#                                                                              #
################################################################################
MoveCursor  ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

YCOR=$1
XCOR=$2

echo "${ESC}${YCOR};${XCOR}H"
}

################################################################################
#                                                                              #
# DisplayMessage                                                               #
#                                                                              #
# This function displays a message                                            #
#                                                                              #
# Input   : Message type (E = Error, I = Information)                          #
#           Error code (defined in DefineMessages)                             #
#                                                                              #
################################################################################
DisplayMessage ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
MESSAGE_TYPE=$1
MESSAGE_TEXT=`eval echo $2`
```

```
clear
MoveCursor 24 1
if [ "${MESSAGE_TYPE}" = "E" ]
then
    echo "`eval echo ${ERROR}`${MESSAGE_TEXT}\c"
else
    echo "`eval echo ${INFO}`${MESSAGE_TEXT}\c"
fi
sleep ${SLEEP_DURATION}
return ${TRUE}
}

##############################################################################
#                                                                            #
# RootUser                                                                   #
#                                                                            #
# This function establishes whether the user is a root user.                 #
#                                                                            #
# Returns : TRUE if user is "root"                                           #
#           FALSE otherwise                                                  #
#                                                                            #
##############################################################################
RootUser ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

USER=`id | cut -d'(' -f2 | cut -d')' -f1`
if [ "${USER}" = "root" ]
then
    return $TRUE
else
    return $FALSE
fi
}

##############################################################################
#                                                                            #
# DisplayMenu                                                                #
#                                                                            #
# This function is used to display a menu.                                   #
#                                                                            #
##############################################################################
DisplayMenu ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

while true
do
    clear
    echo "    ############################################################"
```

```
    echo "    #           ${DURM}                                    #"
    echo "    #                                                      #"
    echo "    #   5  Specific details of physical volumes          #"
    echo "    #  10  All details of the physical volume            #"
    echo "    #  15  Total disk usage for a specific user          #"
    echo "    #  20  Total disk usage for all users                #"
    echo "    #  25  Directory disk usage for a specific user      #"
    echo "    #  30  Specific logical to physical volume mapping   #"
    echo "    #  35  All logical to physical volume mappings       #"
    echo "    #  40  Specific physical to logical volume mapping   #"
    echo "    #  45  All physical to logical volume mapping        #"
    echo "    #  50  Filesystem defragmentation report             #"
    echo "    #  99  Exit.                                         #"
    echo "    #                                                      #"
    echo "    ########################################################"
    echo "            Enter option ──>\c                "
    read OPTION
    ProcessOption
done
}

#############################################################################
#                                                                           #
# GetDateTime                                                               #
#                                                                           #
# This function captures date and time in the variable $DATETIME.           #
#                                                                           #
#############################################################################
GetDateTime ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
DATETIME='date "+%d/%m/%Y at %H:%M:%S"'
}

#############################################################################
#                                                                           #
# CheckError                                                                #
#                                                                           #
# This function checks for command execution errors.                        #
#                                                                           #
# Input    : Return code from the command                                   #
#            Command to check                                               #
#                                                                           #
# Notes    1   The function reads any errors in the error file              #
#              $ERROR_FILE and assigns them to the global variable          #
#              SYSERROR.                                                    #
#                                                                           #
#          2   The function also writes the failed command in the           #
#              output file $REPORT_FILE.                                    #
#                                                                           #
```

```
#############################################################################
CheckError ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
RC=$1   # Return code
FC=$2   # Command to check
if [ $RC -ne 0 ]
then
    DisplayMessage E “${APP_ERROR}”
    SYSERROR=””
    SYSERROR=‘head -1 ${ERROR_FILE}‘
    DisplayMessage E “${OS_ERROR}”
    echo “Failed to execute command, ${FC}\n” >> ${REPORT_FILE}
    return $TRUE
else
    return $FALSE
fi
}

#############################################################################
#                                                                           #
# DisplayListOfValues                                                       #
#                                                                           #
# This function displays a list of values for the given argument.           #
#                                                                           #
# Input   : PV    Physical volume                                           #
#           USER  System user                                               #
#           LV    Logical volume                                            #
#           MP    Mount point                                               #
#                                                                           #
#############################################################################
DisplayListOfValues ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
PARAM=”$1"
SELECTED_VALUE=     # Value selected by user
echo “         List of values for physical volumes “ > ${TEMP_FILE_1}
echo “         ========================== “ >> ${TEMP_FILE_1}
if [ “${PARAM}” = “PV” ]
then
    echo “         List of values for physical volumes “ >
    ➤  ${TEMP_FILE_1}
    echo “         ========================== “ >> ${TEMP_FILE_1}
    echo “\n       Select value by deleting the line and saving the
    ➤  file\n” >> ${TEMP_FILE_1}
    lspv | awk {‘print $1’} | sort >> ${TEMP_FILE_1}
elif [ “${PARAM}” = “USER” ]
then
    echo “         List of values for system users “ > ${TEMP_FILE_1}
    echo “         ======================== “ >> ${TEMP_FILE_1}
```

```
    echo “\n       Select value by deleting the line and saving the
    ➤  file\n” >> ${TEMP_FILE_1}
    cat /etc/passwd | cut -d’:’ -f1 | sort >> ${TEMP_FILE_1}
elif [ “${PARAM}” = “LV” ]
then
    echo “         List of values for logical volumes “ >
    ➤  ${TEMP_FILE_1}
    echo “         ========================== “ >> ${TEMP_FILE_1}
    echo “\n       Select value by deleting the line and saving the
    ➤  file\n” >> ${TEMP_FILE_1}
    lsvg -l ‘lsvg‘ >> ${TEMP_FILE_1}
elif [ “${PARAM}” = “MP” ]
then
    echo “         List of values for mount points” > ${TEMP_FILE_1}
    echo “         ========================” >> ${TEMP_FILE_1}
    echo “\n       Select value by deleting the line and saving the
    ➤  file\n” >> ${TEMP_FILE_1}
    df -k | sed 1d | awk {‘print $7’} >> ${TEMP_FILE_1}
fi
cp ${TEMP_FILE_1} ${TEMP_FILE_2}
view ${TEMP_FILE_1}
SELECTED_VALUE=‘diff ${TEMP_FILE_1} ${TEMP_FILE_2} | tail -1 |
➤  awk {‘print $2’}`
}

##############################################################################
#                                                                            #
# ValidateLogicalVolume                                                      #
#                                                                            #
# This function validates a specified logical volume.                        #
#                                                                            #
# Input   : A logical volume name                                            #
#                                                                            #
# Returns : $TRUE or $FALSE                                                  #
#                                                                            #
##############################################################################
ValidateLogicalVolume ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
P_LV=”$1"
if lslv -l ${P_LV} > /dev/null 2>&1
then
    return $TRUE
else
    DisplayMessage E “${INVALID_LV}”
    return $FALSE
fi
}

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

```
#                                                                              #
# ViewReport                                                                   #
#                                                                              #
# This function is for viewing and printing the report.                        #
#                                                                              #
# Input   : File Name                                                          #
#                                                                              #
################################################################################
ViewReport ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
P_FILE=”$1"
view ${P_FILE}

# Print the file
while true
do
    clear
    echo “Do you wish to print the file (Y/N)?:\c”
    read REPLY
    case $REPLY in
        n|N) return $TRUE ;;
        y|Y) while true
             do
                 clear
                 echo “Enter printer name:\c”
                 read PRINTER
                 case $PRINTER in
                     “”) DisplayMessage E “${INVALID_ENTRY}” ;;
                      *) lp -d${PRINTER} ${P_FILE} > ${ERROR_FILE}
                         ➤  2>&1 ;
                         if [ $? -ne 0 ]
                         then
                             DisplayMessage E “${PRINT_NOT_OK}” ;
                             return $FALSE
                         else
                             DisplayMessage I “${PRINT_OK}” ;
                             return $TRUE
                         fi ;;
                 esac
             done ;;
          *) DisplayMessage  E “${INVALID_ENTRY}” ;;
    esac
done
}

################################################################################
#                                                                              #
# ValidUser                                                                    #
#                                                                              #
```

```
# This function validates a userid using the /etc/passwd file.      #
#                                                                    #
######################################################################
ValidUser ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
UID=”$1"
cat /etc/passwd | while read LINE
                  do
                      if [ “${UID}” = “`echo $LINE | cut -d’:’ -f1`” ]
                      then
                          return $TRUE
                      fi
                  done
return $FALSE
}

######################################################################
#                                                                    #
# PrintHeaderLine                                                    #
#                                                                    #
# This function adds a “double line” character to variable           #
# $HEADER_LINE for each character in the variable $STRING.           #
#                                                                    #
# Input   : A string with the header text                            #
#                                                                    #
######################################################################
PrintHeaderLine ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
STRING=”$1"
HEADER_LINE=””
STRLEN=‘echo “$STRING\c” | wc -c‘
INDEX=0
while [ $INDEX -ne $STRLEN ]
do
    HEADER_LINE=”${HEADER_LINE}=”
    INDEX=‘expr $INDEX + 1‘
done
}

######################################################################
#                                                                    #
# GetHomeDirectory                                                   #
#                                                                    #
# This function reads a user’s home directory from the /etc/passwd   #
# file.                                                              #
#                                                                    #
# Input   : A userid                                                 #
#                                                                    #
```

```
##############################################################################
GetHomeDirectory ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
UID=”$1"
cat /etc/passwd | while read LINE
do
    if [ “${UID}” = “`echo $LINE | cut -d’:’ -f1`” ]
    then
        HOME_DIR=”‘echo $LINE | cut -d’:’ -f6‘“
        return $TRUE
    fi
done
}

##############################################################################
#                                                                            #
# SpecificUserTotalDiskUsage                                                 #
#                                                                            #
# This function calculates and displays the total disk usage for a  #
# specified user.                                                            #
#                                                                            #
##############################################################################
SpecificUserTotalDiskUsage ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
while true
do
    clear
    echo “Enter user name (l = list of values):\c”
    read UNAME
    case $UNAME in
         “” ) DisplayMessage E “${INVALID_USER}” ;
              continue;;
        l|L ) DisplayListOfValues “USER” ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                  UNAME=”${SELECTED_VALUE}” ;
              fi ;;
          * ) : ;;
    esac
    if ! ValidUser “${UNAME}”
    then
        DisplayMessage E “${INVALID_USER}”
        return $FALSE
    else
        break
    fi
```

```
done
GetHomeDirectory “${UNAME}”
DUK=‘du -sk  $HOME_DIR | awk {‘print $1’}‘
DUB=‘bc <<!
scale=2
$DUK * 1024
!‘
GetDateTime
HEADER=”Total disk usage for $UNAME on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “      Total disk usage for $UNAME on ${DATETIME}” >
➤  ${REPORT_FILE}
echo “      $HEADER_LINE” >> ${REPORT_FILE}
echo “$DUB bytes” >> ${REPORT_FILE}
ViewReport ${REPORT_FILE}
}

##############################################################################
#                                                                            #
# AllUserTotalDiskUsage                                                      #
#                                                                            #
# This function calculates and displays the total disk usage for    #
# all users.                                                                 #
#                                                                            #
##############################################################################
AllUserTotalDiskUsage ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
DisplayMessage I “${WORKING}”

# Get date and time
GetDateTime

# Print header in the report file
HEADER=”Total disk usage for all users on ${DATETIME}”
PrintHeaderLine  “${HEADER}”
echo “      $HEADER” > ${REPORT_FILE}
echo “      $HEADER_LINE\n” >> ${REPORT_FILE}

# Process all users from /etc/passwd file,
# ignoring users with home directories in “/”
cat /etc/passwd | while read LINE
do
    UID=”‘echo $LINE | cut -d’:’ -f1‘“
    HOME_DIR=”‘echo $LINE | cut -d’:’ -f6‘“
    if [ “${HOME_DIR}” = “/” ]
    then
        continue
    fi
```

```
    # Get summary in kilobyte blocks
    DUK=‘du -sk  $HOME_DIR | awk {‘print $1’}‘

    # Calculate summaries in bytes
    DUB=‘bc <<!
    scale=2
    $DUK * 1024
!‘
    LINE=‘echo “$UID $HOME_DIR $DUB” | \
    awk {‘printf(“%-10s—>%-45s—>%-13s”,$1,$2,$3)’}`
    echo “$LINE” >> ${REPORT_FILE}
done
ViewReport ${REPORT_FILE}
}

#############################################################################
#                                                                           #
# SpecificUserDirectoryDiskUsage                                           #
#                                                                           #
# This function calculates and displays disk usage in all                   #
# directories for a specified user.                                         #
#                                                                           #
#############################################################################
SpecificUserDirectoryDiskUsage ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP

# Get user name
while true
do
    clear
    echo “Enter user name (l = list of values):\c”
    read UNAME
    case $UNAME in
         “” ) DisplayMessage E “${INVALID_USER}” ;
              continue;;
        l|L ) DisplayListOfValues “USER”  ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                  UNAME=”${SELECTED_VALUE}” ;
                  break ;
              fi ;;
          * ) if !  ValidUser “${UNAME}”
              then
                  DisplayMessage E “${INVALID_USER}” ;
              else
                  break ;
              fi ;;
```

```
    esac
done
DisplayMessage I "${WORKING}"

# Get user's home directory
GetHomeDirectory "${UNAME}"

# Get sorted summaries of disk usage in kilobyte blocks
du -k $HOME_DIR | sort -k1n > ${TEMP_FILE_1}

# Get summaries in bytes
> ${TEMP_FILE_2}
cat ${TEMP_FILE_1} | while read LINE
do
    DUK=`echo "$LINE" | awk {'print $1'}`
    DIR=`echo "$LINE" | awk {'print $2'}`
    DUB=`bc <<!
    scale=2
    $DUK * 1024
!`
    LINE=`echo "$DIR $DUB" | awk {'printf("%-50s———>%-13s",
    ➤  $1,$2)'}`
    echo "$LINE" >> ${TEMP_FILE_2}
done

# Get date and time
GetDateTime

# Print header in report file
HEADER="Directory Disk usage for $UNAME on ${DATETIME}"
PrintHeaderLine "${HEADER}"
echo "      Directory disk usage for $UNAME on ${DATETIME}" >
➤  ${REPORT_FILE}
echo "      $HEADER_LINE" >> ${REPORT_FILE}

# Append disk usage summaries to report file
cat ${TEMP_FILE_2} >> ${REPORT_FILE}

# View report file
ViewReport ${REPORT_FILE}
}

#############################################################################
#                                                                           #
# ValidatePhysicalVolume                                                    #
#                                                                           #
# This function validates a physical volume name.                           #
#                                                                           #
# Input   : A physical volume name                                          #
#                                                                           #
```

```
#############################################################################
ValidatePhysicalVolume ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
P_PV=”$1"
if lspv | grep “${P_PV}” > /dev/null 2>&1
then
    return $TRUE
else
    DisplayMessage E “${INVALID_PV}”
    return $FALSE
fi
}

#############################################################################
#                                                                           #
#  SpeficPhysicalVolumeDetails                                              #
#                                                                           #
#  This function retrieves details of a specified physical volume.  #
#                                                                           #
#############################################################################
SpecificPhysicalVolumeDetails ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
while true
do
    clear
    echo “Enter required physical volume (l = list):\c”
    read PV
    case $PV in
        l|L ) DisplayListOfValues “PV” ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                  PV=”${SELECTED_VALUE}” ;
                  break ;
              fi ;;
         “” ) DisplayMessage E “${INVALID_ENTRY}” ;;
          * ) if ValidatePhysicalVolume “${PV}”
              then
                  break ;
              fi ;;
    esac
done
DisplayMessage I “${WORKING}”
GetDateTime
HEADER=”Physical volume details for ${PV} on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “        ${HEADER}” > ${REPORT_FILE}
```

```
echo “        ${HEADER_LINE}” >> ${REPORT_FILE}
COMMAND=”lspv ${PV}”
lspv ${PV} 1>> ${REPORT_FILE} 2> ${ERROR_FILE}
if CheckError $? “${COMMAND}”
then
    return $FALSE
fi
echo “ \n” >> ${REPORT_FILE}
ViewReport ${REPORT_FILE}
}

###############################################################################
#                                                                             #
# AllPhysicalVolumeDetails                                                    #
#                                                                             #
# This function retrieves all physical volume details. It also                #
# calculates the total disk capacity.                                         #
#                                                                             #
###############################################################################
AllPhysicalVolumeDetails ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
DisplayMessage I “${WORKING}”
GetDateTime
HEADER=”Physical volume details on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “        ${HEADER}” > ${REPORT_FILE}
echo “        ${HEADER_LINE}” >> ${REPORT_FILE}
COMMAND=”lspv | awk {‘print \$1’}”
lspv | awk {‘print $1’} 1> ${TEMP_FILE_1} 2> ${ERROR_FILE}
if CheckError $? “${COMMAND}”
then
    return $FALSE
fi
cat ${TEMP_FILE_1} | while read HD
do
    HEADER=”${HD}”
    PrintHeaderLine “${HEADER}”
    echo “                      $HEADER”          >> ${REPORT_FILE}
    echo “                      ${HEADER_LINE}\n” >> ${REPORT_FILE}
    COMMAND=”lspv ${HD}”
    lspv ${HD} 1>> ${REPORT_FILE} 2> ${ERROR_FILE}
    if CheckError $? “${COMMAND}”
    then
        return $FALSE
    fi
    echo “ \n” >> ${REPORT_FILE}
done

# Calculate TDC (Total Disk Capacity)
```

```
# Assign all the disk capacities to LODC (List Of Disk Capacity)
LODC=‘grep “TOTAL PPs” ${REPORT_FILE} | \
    cut -d’(‘ -f2 | cut -d’)’ -f1 | awk {‘print $1’}`
TDC=0
for DC in $LODC
do
    TDC=‘expr $TDC + $DC‘
done
echo “\n” >> ${REPORT_FILE}
echo “Total disk capacity = $TDC (megabytes)\n” >> ${REPORT_FILE}
ViewReport ${REPORT_FILE}
}

###############################################################################
#                                                                             #
# SpecificLogicalToPhysicalVolumeMapping                                      #
#                                                                             #
# This function prints physical volume mappings for a specific               #
# logical volume.                                                             #
#                                                                             #
###############################################################################
SpecificLogicalToPhysicalVolumeMapping ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
while true
do
    clear
    echo “Enter logical volume name (l = list):\c”
    read LV
    case $LV in
        l|L ) DisplayListOfValues LV ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                  LV=”${SELECTED_VALUE}” ;
                  if ValidateLogicalVolume “${LV}”
                  then
                      break;
                  else
                      continue ;
                  fi
              fi ;;
         “” ) DisplayMessage E “${INVALID_ENTRY}” ;;
          * ) if ValidateLogicalVolume “${LV}”
              then
                  break ;
              else
                  : ;
              fi ;;
```

```
    esac
done

DisplayMessage I "${WORKING}"
GetDateTime
HEADER="Logical to physical volume mapping for ${LV} on ${DATETIME}"
PrintHeaderLine "${HEADER}"
echo "        ${HEADER}" > ${REPORT_FILE}
echo "        ${HEADER_LINE}" >> ${REPORT_FILE}
lslv -l ${LV} | sed 1,2d | awk {'print "Disk-> "$1"
➤  Distribution-> "$4'} >> ${REPORT_FILE}
echo "\n\n" >> ${REPORT_FILE}
ViewReport ${REPORT_FILE}
}

###############################################################################
#                                                                             #
# AllLogicalToPhysicalVolumeMapping                                           #
#                                                                             #
# The function prints physical volume mappings for every logical              #
# volume in each volume group.                                                #
#                                                                             #
# Input   : FROM_MENU (if called directly from the menu) or                   #
#           INTERNAL  (if called by another function)                         #
#                                                                             #
###############################################################################
AllLogicalToPhysicalVolumeMapping ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
CALL="$1"
DisplayMessage I "${WORKING}"
GetDateTime
HEADER="Logical to physical volume mapping on ${DATETIME}"
PrintHeaderLine "${HEADER}"
echo "        ${HEADER}" > ${REPORT_FILE}
echo "        ${HEADER_LINE}" >> ${REPORT_FILE}
COMMAND="lsvg"
lsvg 1> ${TEMP_FILE_1} 2> ${ERROR_FILE}
if CheckError $? "${COMMAND}"
then
    return $FALSE
fi

# Process each volume group in turn
cat ${TEMP_FILE_1} | while read VG
do
    # For each volume group, get all the logical volumes
    echo "       \n"                    >> ${REPORT_FILE}
    echo "       Volume group = ${VG}" >> ${REPORT_FILE}
    lsvg -l ${VG} | awk {'print $1'} > ${TEMP_FILE_2}
```

```
    # Save reading from ${TEMP_FILE_1}
    exec 5<&0

    # For each volume, get all physical volume details
    # (name and distribution) from ${TEMP_FILE_2}.
    # Delete the top two lines, which are headings.
    sed 1,2d ${TEMP_FILE_2} > ${TEMP_FILE_3}
    cat ${TEMP_FILE_3} | while read LV
    do
        echo "       \n" >> ${REPORT_FILE}
        if [ "${CALL}" = "FROM_MENU" ]
        then
            # Apply format
            echo "          Volume = ${LV} " >> ${REPORT_FILE}
            lslv -l ${LV} | sed 1,2d | awk {'print "Disk-> "$1"
            ➤  Distribution-> "$4'} >> ${REPORT_FILE}
        else
            # Prepare file for calling function to assign all
            # physical volumes to LOHD (List Of Hard Disks)
            LOHD=`lslv -l ${LV} | sed 1,2d | awk {'print $1'}`
            for HD in ${LOHD}
            do
                echo "$HD  $LV" >> ${REPORT_FILE}
            done
        fi
    done
    # Resume reading from ${TEMP_FILE_1}
    exec 0<&5
done
echo "\n\n" >> ${REPORT_FILE}
if [ "${CALL}" = "FROM_MENU" ]
then
    ViewReport ${REPORT_FILE}
else
    return $TRUE
fi
}

################################################################################
#                                                                              #
# SpecificPhysicalToLogicalVolumeMapping                                       #
#                                                                              #
# This function prints logical volume mapping for a specified                 #
# physical volume.                                                             #
#                                                                              #
################################################################################
SpecificPhysicalToLogicalVolumeMapping ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
while true
```

```
do
    clear
    echo “Enter physical volume name (l = list):\c”
    read PV
    case $PV in
        l|L ) DisplayListOfValues PV ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                  PV=”${SELECTED_VALUE}” ;
                  if ValidatePhysicalVolume “${PV}”
                  then
                      break ;
                  else
                      continue ;
                  fi
              fi ;;
         “” ) DisplayMessage E “${INVALID_ENTRY}” ;;
          * ) if ValidatePhysicalVolume “${PV}”
              then
                  break ;
              else
                  : ;
              fi ;;
    esac
done
DisplayMessage I “${WORKING}”

# Call LogicalToPhysicalVolumeMapping with parameter “INTERNAL”
# to obtain a file with physical and logical volume details
AllLogicalToPhysicalVolumeMapping “INTERNAL”
cp ${REPORT_FILE} ${TEMP_FILE_1}
GetDateTime
HEADER=”Physical to logical volume mapping for ${PV} on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “        ${HEADER}” > ${REPORT_FILE}
echo “        ${HEADER_LINE}” >> ${REPORT_FILE}

# Get all the logical volumes for this physical volume
if grep “${PV}” ${TEMP_FILE_1} > /dev/null 2>&1
then
    # Volume is assigned, so obtain all logical
    # volumes for this physical volume
    grep “${PV}” ${TEMP_FILE_1} | awk {‘print “Logical Volume
    ➤  —>”$2'} >> ${REPORT_FILE}
else
    echo “Physical volume is not assigned “ >> ${REPORT_FILE}
fi
ViewReport ${REPORT_FILE}
}
```

```
#############################################################################
#                                                                           #
# AllPhysicalToLogicalVolumeMapping                                         #
#                                                                           #
# This function prints logical volume mappings for each physical            #
# volume.                                                                   #
#                                                                           #
# Notes   1   This function makes the following function call:              #
#                LogicalToPhysicalVolumeMapping “INTERNAL”                  #
#                                                                           #
#############################################################################
AllPhysicalToLogicalVolumeMapping ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
DisplayMessage I “${WORKING}”

# Call LogicalToPhysicalVolumeMapping with parameter “INTERNAL”
# to obtain a file with physical and logical volume details.
AllLogicalToPhysicalVolumeMapping “INTERNAL”
cp ${REPORT_FILE} ${TEMP_FILE_1}
GetDateTime
HEADER=”Physical to logical volume mapping on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “        ${HEADER}” > ${REPORT_FILE}
echo “        ${HEADER_LINE}” >> ${REPORT_FILE}
COMMAND=”lspv”
lspv | awk {‘print $1’} 1> ${TEMP_FILE_2} 2> ${ERROR_FILE}
if CheckError $? “${COMMAND}”
then
    return $FALSE
fi

# Process each physical volume in turn
cat ${TEMP_FILE_2}  |  while  read  PV
do
    # For each physical volume, get all logical volumes
    echo “        \n”                          >> ${REPORT_FILE}
    echo “        Physical volume = ${PV}\n” >> ${REPORT_FILE}
    if grep “${PV}” ${TEMP_FILE_1} > /dev/null 2>&1
    then
        # Volume is assigned, so obtain all logical
        # volumes for this physical volume.
        grep “${PV}” ${TEMP_FILE_1} | awk {‘print “Logical volume
        ➤  —>”$2'} >> ${REPORT_FILE}
    else
        echo “Physical volume is not assigned “ >> ${REPORT_FILE}
    fi
done
ViewReport ${REPORT_FILE}
}
```

```
#############################################################################
#                                                                           #
# ReportFileSystemDefragmentation                                           #
#                                                                           #
# This function prepares a filesystem fragmentation report.                 #
#                                                                           #
#############################################################################
ReportFileSystemDefragmentation ( )
{
trap “HandleInterrupt” $SIGINT $SIGTERM $SIGHUP
while true
do
    clear
    echo “Enter filesystem mount point (l = list):\c”
    read MPOINT
    case $MPOINT in
         “” ) DisplayMessage E “${INVALID_ENTRY}” ;
              continue ;;
        l|L ) DisplayListOfValues “MP” ;
              if [ “${SELECTED_VALUE}” = “” ]
              then
                  continue ;
              else
                   MPOINT=”${SELECTED_VALUE}” ;
              fi ;;
          * ) : ;;
    esac

    # Validate mount point
    if df -k | sed 1d | awk {‘print $7’} | grep ${MPOINT} >
    ➤  /dev/null 2>&1
    then
        break
    else
        DisplayMessage E “${INVALID_MPOINT}”
    fi
done
GetDateTime
HEADER=”Fragmentation report for filesystem on $MPOINT on ${DATETIME}”
PrintHeaderLine “${HEADER}”
echo “  ${HEADER}”      >  ${REPORT_FILE}
echo “  ${HEADER_LINE}” >> ${REPORT_FILE}
echo “\n”               >> ${REPORT_FILE}
DisplayMessage I “${WORKING}”
defragfs -r ${MPOINT}   >> ${REPORT_FILE}
ViewReport ${REPORT_FILE}
}

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

```
# ProcessExit                                                          #
#                                                                      #
# This function processes various menu options.                        #
#                                                                      #
########################################################################
ProcessExit ( )
{
EXIT_CODE="$1"

rm -f $REPORT_FILE
rm -f $TEMP_FILE_1
rm -f $TEMP_FILE_2
rm -f $TEMP_FILE_3

exit ${EXIT_CODE}
}

########################################################################
#                                                                      #
# ProcessOption                                                        #
#                                                                      #
# This function processes various menu options.                        #
#                                                                      #
########################################################################
ProcessOption ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
case $OPTION in
     5) SpecificPhysicalVolumeDetails ;;
    10) AllPhysicalVolumeDetails ;;
    15) SpecificUserTotalDiskUsage ;;
    20) AllUserTotalDiskUsage ;;
    25) SpecificUserDirectoryDiskUsage ;;
    30) SpecificLogicalToPhysicalVolumeMapping ;;
    35) AllLogicalToPhysicalVolumeMapping "FROM_MENU" ;;
    40) SpecificPhysicalToLogicalVolumeMapping ;;
    45) AllPhysicalToLogicalVolumeMapping ;;
    50) ReportFileSystemDefragmentation ;;
    99) ProcessExit $SEC ;;
     *) DisplayMessage E "${INVALID_OPTION}" ;;
esac
}

########################################################################
#                                                                      #
# main                                                                 #
#                                                                      #
# This function invokes all other functions.                           #
#                                                                      #
########################################################################
```

```
main ()
{
InitialiseVariables
if ! RootUser
then
    DisplayMessage E “${REQ_USER}”
    clear
    exit $FEC
fi
DisplayMenu
}

# Invoke main
main
```

Overleaf are some sample reports. Note that the reports are truncated for the sake of brevity (the remainder of each report is the same as that which we’ve printed).

SAMPLE OUTPUT 1

```
          List of values for system users
          ===============================

        Select value by deleting the line and saving the file

adm
bfmgr
bin
bor4mgr
daemon
```

Note: this file is displayed using the **view** command and, therefore, you must use ‘w!’ (forced write) to write and save the file.

SAMPLE OUTPUT 2

```
     Physical volume details for hdisk0 on 28/03/1999 at 15:59:49
        ============================================
PHYSICAL VOLUME:    hdisk0                  VOLUME GROUP:     rootvg
PV IDENTIFIER:      00001069d0049b74        VG IDENTIFIER
00001069d004a326
PV STATE:           active
STALE PARTITIONS:   0                       ALLOCATABLE:      yes
PP SIZE:            8 megabyte(s)           LOGICAL VOLUMES:  8
TOTAL PPs:          518 (4144 megabytes)    VG DESCRIPTORS:   2
FREE PPs:           27 (216 megabytes)
USED PPs:           491 (3928 megabytes)
```

```
FREE DISTRIBUTION:  27..00..00..00..00
USED DISTRIBUTION:  77..104..103..103..104
```

SAMPLE OUTPUT 3

```
      Directory disk usage for kanna_d on 28/03/1999 at 16:01:09
      ==========================================
/home/kanna_d/EuroCAT/v1.00/ctl                  ———>1024
/home/kanna_d/EuroCAT/v1.00/func                 ———>1024
/home/kanna_d/EuroCAT/v1.00/log                  ———>1024
/home/kanna_d/EuroCAT/v1.00/wrk                  ———>1024
/home/kanna_d/lns/v1.00/todo                     ———>1024
/home/kanna_d/EuroCAT/v1.00/archive              ———>2048
/home/kanna_d/lns/v1.00/wrk/euro                 ———>2048
```

SAMPLE OUTPUT 4

```
        Physical to logical volume mapping on 28/03/1999 at 16:03:56
        ============================================

         Physical Volume  = hdisk0

Logical Volume —>hd6
Logical Volume —>hd5
Logical Volume —>hd8
Logical Volume —>hd4

         Physical Volume  = hdisk1

Logical Volume —>lv01
Logical Volume —>loglv00

         Physical Volume  = hdisk2

Logical Volume —>lv02

         Physical Volume  = hdisk4
```

SAMPLE OUTPUT 5

```
        Logical to physical volume mapping on 28/03/1999 at 16:07:18
        ============================================

       Volume Group  = rootvg

          Volume = hd6
Disk—> hdisk0 Distribution—> 037:104:024:000:091
```

```
          Volume = hd5
Disk—> hdisk0 Distribution—> 001:000:000:000:000

          Volume = hd8
Disk—> hdisk0 Distribution—> 000:000:001:000:000

          Volume = hd4
Disk—> hdisk0 Distribution—> 010:000:001:002:013

       Volume Group  = usrvg

          Volume = lv01
Disk—> hdisk1 Distribution—> 104:104:103:103:103

          Volume = lv02
Disk—> hdisk2 Distribution—> 104:104:103:103:103
```

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

Report script for administrators

Our company has several AIX systems that must be managed and tuned continually, with problems reported to the administrator. This requires the administrator to login at the systems concerned to check unread mail for *root*, error logs, etc. This can be a burden to the people concerned, so, to ease their job, we've developed scripts to automate this task. The scripts split systems into two groups:

- One system on which you can view data about other systems (in this example *afts1*). This is the 'result machine'.
- All the other systems.

FOCUSING ON THE PROBLEMS

What we require of the system is the ability to:

- Monitor filesystem usage. If a filesystem grows too large, you have to decide whether to remove some files or expand the filesystem. Note that some filesystems are static (for example database and CD-ROM filesystems) and should be skipped.
- Generate mail for *root* (or whoever is responsible for receiving alerts) to report problems encountered during the day. To send mail to *root*, use the following command:

  ```
  mail root </file_to_report
  ```

- React to space problems that affect *root* and other filesystems. You need to establish what constitutes a problem – in our case we decided that we have a problem when space in a critical filesystem falls below 200 MB.
- Monitor space in *Rootvg*. When you have to restore a BosBoot tape, it is sometimes necessary to increase the size of the */tmp* filesystem. If you don't have the required space, the restore fails.
- Monitor the space in other volume groups, which (aggregated together) must also have a minimum of 200 MB of free space. You have the option of either adding extra disk(s) or cleaning up your disk(s) before the shortage of space causes serious problems.
- Carry out error reporting on disks and tapes.
- Report on databases (for instance, whether they are running or not).
- Report on the status of backups (for instance, did they complete successfully?).
- Ensure there is enough space to accommodate a dump in */var*, should it be necessary.

GETTING INTO DETAILS

The script that gathers the necessary information is called **gatherall**, and a copy of it should be placed on all Unix systems that have to be managed. The script that's used to inform the administrator is called **look_all** and it's located on the 'result machine'.

Note the use of the continuation character (‘➤’) in the scripts below to indicate that one line of code maps to several lines of print.

GATHERALL

```
# Name script    : /home/oper/gatherall
# Description    : Send all data to the result machine afts1
#————————————————————————————
logmsg $0 “Begin script $0”

# Remove all work files when you’re not on the result machine (output
# has to be deleted using crontab commands).
if [ ! -s /home/oper/look_all ]
  then rm /tmp/gather*
fi

# Produce an error report on tape devices and disks, redirect it
# to a file, and mail it to root.
if [ -s /tmp/gather.tmp ]
  then mail root </tmp/gather.tmp
    sleep 2
fi
rm /tmp/gather.tmp

# Get the hostname of the machine.
SITE=‘hostname‘

# Check whether /var can accommodate a system dump.
#
# sysdumpdev -e gives the amount of space (in bytes) needed to
# store the dump.
bytes_needed=‘sysdumpdev -e  | cut -f6 -d “ “‘

# Check the amount of free space in /var.
string_free=‘df -Ik | grep var‘
kbytes=‘echo $string_free | cut -f4 -d” “‘
let bytes_free=$kbytes*1024

# Check that there are sufficient kilobytes free.
let bytes_to_expand=$bytes_needed-$bytes_free
if [ $bytes_to_expand -gt 0 ]
  then    # The /var filesystem needs to be expanded.
    let bytes_to_expand=$bytes_to_expand/1024
    if [ bytes_to_expand -eq 0 ]
      then    # By dividing you’ve lost some bytes.
        bytes_to_expand=100
    fi
    # Create a dialogue box to display.
```

```
    while [ `expr "$bytes_to_expand" : ".*"` -lt 6 ]
      do
        bytes_to_expand=' '$bytes_to_expand
      done
    let bytes_needed=$bytes_needed/1024
    while [ `expr "$bytes_needed" : ".*"` -lt 6 ]
      do
        bytes_needed=' '$bytes_needed
      done
    echo "    **************************************************"
    ➤  >/tmp/dump_space.tmp
    echo "    *     /var has to be expanded with $bytes_to_expand Kb
    ➤       *" >>/tmp/dump_space.tmp
    echo "    *  because a crash dump can't be stored into /var *"
    ➤  >>/tmp/dump_space.tmp
    echo "    *         The next crash dump needs $bytes_needed Kb
    ➤      *" >>/tmp/dump_space.tmp
    echo "    **************************************************"
    ➤  >>/tmp/dump_space.tmp
    # Mail it to root, so it can be processed in the next step.
    mail root </tmp/dump_space.tmp
    rm /tmp/dump_space.tmp
fi

# Report filesystem space usage (detect when a filesystem has to
# be expanded or cleaned). The OS level is needed as AIX 3.2.5 does
# not support the -k flag
Os_Level=`oslevel`
if [ "$Os_Level" = "<>3250" -o "$Os_Level" = ">3250" ]
  then echo "Output of the command df -I on $SITE:\n\n"
    ➤  >/tmp/gather_dfI
    df -I >>/tmp/gather_dfI
  else echo "Output of the command df -Ik on $SITE:\n\n"
    ➤  >/tmp/gather_dfI
    df -Ik >>/tmp/gather_dfI
fi

# Report on the root user's mail and clean it.
cp /var/spool/mail/root /tmp/gather_mail
>/var/spool/mail/root

# If there is no mail, fill the file (you can't ftp files of
# zero bytes).
if [ ! -s /tmp/gather_mail ]
  then echo "No mail for root" >/tmp/gather_mail
fi

# Use ftp to send the files to the result machine. The file
# /home/data/rootpwd contains the root password.
# Using the site command means that everyone can see the results.
```

```
# This also means that you have to use chmod on each file, as the
# site command is seeking (for example) for the file gather_mail*,
# which doesn't exist.
exec >/tmp/gather_afts1.netrc
echo "machine afts1.aft.schuitema login root password \c"
echo `cat /home/data/rootpwd`
echo "macdef init"
echo "type binary"
# The next files are sent to the result machine.
echo "put /tmp/gather_dfI /tmp/gather_dfI.$SITE"
echo "site chmod 664 /tmp/gather_dfI.$SITE"
echo "put /tmp/gather_mail /tmp/gather_mail.$SITE"
echo "site chmod 664 /tmp/gather_mail.$SITE"
echo "put /home/logging/gather_saves.log /tmp/gather_saves.$SITE"
echo "site chmod 664 /tmp/gather_saves.$SITE"

# This section of code is very specific to our own installation,
# and so needs to be modified or discarded when you implement the
# code at your own site. It checks whether Oracle is installed, and,
# if it is, it checks all databases to ensure that they are
# running. check_alldb is a home-made script that checks all
# the databases named in /etc/oratab that should be started (for
# more details on how this works, please contact me).
if [ -s /etc/oratab ]
  then check_alldb >/tmp/gather_cadb
    echo "put /tmp/gather_cadb /tmp/gather_cadb.$SITE"
    echo "site chmod 664 /tmp/gather_cadb.$SITE"
fi

# Produce a space usage report
# A minimum of 200 MB is required for the following reasons:
#  - When you have to restore a BosBoot tape, it is sometimes
#    necessary to expand the /tmp filesystem. If the filesystem
#    isn't sufficiently large, the restore fails.
#  - A threshold warning gives you time to clean up disks or add
#    extra capacity.
Min_Free_Space=200

# Calculate the total free space in rootvg.
#
# Initialize the variables.
Free_Rootvg=0
Free_Rest=0
# List all rootvg disks, remove the header, and grep Free PPs.
lsvg -p rootvg | grep hdisk | cut -c40-50 | while read FreeSpace
    do
        let Free_Rootvg=Free_Rootvg+$FreeSpace
    done
# Calculate total free space of other volume groups.
# List all volume groups other than rootvg.
```

```
Rest_Vg=‘lsvg -o | grep -v rootvg‘
if [ -z “$Rest_Vg” ]
    then    # Only one volume group, rootvg.
        Free_Rest=$Min_Free_Space
    else
        lsvg -p $Rest_Vg | grep hdisk | cut -c40-50 | while
        ➤  read Free_Space
            do
                let Free_Rest=$Free_Rest+$Free_Space
            done
fi

# Calculate free space (1 PP = 4 MB).
let Free_Rootvg_Mb=$Free_Rootvg*4
let Free_Rest_Mb=$Free_Rest*4

# Redirect output to a file.
exec >/tmp/gather_space.$SITE
if [ $Free_Rootvg_Mb -lt $Min_Free_Space  -o $Free_Rest_Mb -lt
➤  $Min_Free_Space ]
then     echo “\n\t**********************************************
➤  **********”
    echo “\t*                                                    *”
    echo “\t*      Space problems detected on machine: $SITE\t*”
    echo “\t*             rootvg      : $Free_Rootvg Mb          \t\t*”
            if [ ! -z “$Rest_Vg” ]
                then    # Check $Free_Rest_Mb for additional capacity.
    echo “\t*                other vg’s: $Free_Rest Mb
    ➤  \t\t*”
            fi
    echo “\t*                                                    *”
    echo “\t******************************************************
    ➤  ***\n”
    # Send the file to the result machine.
    exec >/tmp/gather_afts1.netrc
    echo “put /tmp/gather_space.$SITE “
    echo “site chmod 664 /tmp/gather_space.$SITE”
    fi

if [ -s /home/oper/look_all ]
then
    # You are on the result machine.
    # Set the results in the proper format, clean-up unused files.
    mv /tmp/gather_dfI  /tmp/gather_dfI.$SITE
    mv /tmp/gather_mail /tmp/gather_mail.$SITE
    mv /tmp/gather_cadb /tmp/gather_cadb.$SITE
    rm /tmp/gather_afts1.netrc
else
    cat /home/data/einde.netrc
    exec >/dev/tty
```

```
    # Check whether batch FTP is already running
    # (lockfile /home/data/netrc.busy).
    # If yes, generate a message and wait one minute
    while [ -f /home/data/netrc.busy ]
        do read n1 </home/data/netrc.busy
            logmsg $0 “FTP of project $n1 busy - wait a minute.”
            sleep 60
        done

    # Lockfile isn’t there, so put “gather” in the file (this locks
    # other file tranfers.
    echo “gather” >/home/data/netrc.busy
    if [ -r $HOME/.netrc ]
        then
            # You already have a .netrc file: save it and
            # remember to set it back.
            mv $HOME/.netrc $HOME/netrc.old
            REPLACE_NETRC=0
        else
            REPLACE_NETRC=1
    fi
    mv /tmp/gather_afts1.netrc  $HOME/.netrc
    chmod 600 $HOME/.netrc

    # Start ftp to the result machine.
    ftp -v afts1.aft.schuitema 1>/tmp/gatherafts1 2>&1
    # Check the result of the ftp.
    testftp /tmp/gatherafts1 afts1 gather
    RETC=$?
    if [ “$RETC” -ne 0 ]
        then
            d_msg $0 gather SBY “FTP to afts1 failed” j
        else
            logmsg $0 “FTP to afts1 OK” gather
            rm /tmp/gatherafts1
    fi
    rm /home/data/netrc.busy
fi

# restore the original $HOME/.netrc
if [ “$REPLACE_NETRC” = 0 ]
    then
        mv $HOME/netrc.old $HOME/.netrc
    else
        rm $HOME/.netrc
fi

# Reset the “save-log” and check that the save is started.
echo “No saves started” >/home/logging/gather_saves.log
```

```
logmsg $0 “End script $0” gather
```

Script **look_all** looks at the output from all servers. This script has to be installed on the ‘result machine’.

LOOK_ALL

```
# Name script        : /home/oper/look_all
# Description        : Check all output from all AIX systems
#————————————————————————
check_answer ()
{
# The user made use of the keys <Ctrl>+C or <Ctrl>+D.
if [ -r /tmp/look_all.ans.$UNIQUE ]
then    # Exit program: the user wants to stop execution.
    # Remove work files.
    case $1 in
        mail  )
            rm /tmp/look_all.$USER.mail*
            rm /tmp/look_all.$USER.lokje
            # remove mail/mailbox that is created in the session
            rm $HOME/mbox                    2>/dev/null
            >/usr/spool/mail/$USER
            ;;
        db    )
            rm /tmp/look_all.cadb$UNIQUE
            ;;
        dfi   )
            rm /tmp/look_all.dfI*.$UNIQUE    2>/dev/null
            rm /tmp/look_all.$UNIQUE.dfi*    2>/dev/null
            ;;
        space )
            rm /tmp/look_all.space$UNIQUE    2>/dev/null
            ;;
        *     )
    esac

    rm /tmp/look_all.ans.$UNIQUE
    rm /tmp/look_all.c$UNIQUE*         2>/dev/null
    rm /tmp/look_all.pg$UNIQUE         2>/dev/null
    echo “\n\n”
    scripteinde look_all
    exit 1
fi
}
check_sites ()
{
# Check that all sites sent their output to the result machine.
# CHKFILE is a file with all the server names to be checked.
```

```
case $1 in
    cadb  ) CHKFILE=/home/data/look_all.dbservers;;
    *     ) CHKFILE=/home/data/look_all.allservers;;
esac

# Get all file names with extension $1 (the parameter for mail).
ls /tmp/gather_$1.* | awk -F. '{print $2}'
➤  >/tmp/look_all.c$UNIQUE.$USER.$1

# Search sites not in /tmp.
diff /tmp/look_all.c$UNIQUE.$USER.$1 $CHKFILE | grep ">" | cut -c3-
➤  >/tmp/look_all.c${UNIQUE}2.$USER.$1

# Search sites not in $CHKFILE.
diff /tmp/look_all.c$UNIQUE.$USER.$1 $CHKFILE | grep "<" | cut -c3-
➤  >/tmp/look_all.c${UNIQUE}3.$USER.$1

# Set $SHOW to "No" (all information is present).
SHOW=N

exec >/tmp/look_all.pg$UNIQUE
if [ -s /tmp/look_all.c${UNIQUE}2.$USER.$1 ]
    then
        SHOW=Y
        # There is a possible problem, such as ftp is not started,
        # and someone has to be informed - in this example, the
        # administrator.
        echo "\n\n\n\t*********************************************
        ➤  *********"
        echo "\t*         No information received ( $1 ) of:\t    *"
        echo "\t*                                                   *"
        cat /tmp/look_all.c${UNIQUE}2.$USER.$1 | while read LINE
        do
            echo "\t*               -> $LINE              \t    *"
        done
        echo "\t*                                                   *"
        echo "\t*              Inform the administrator             *"
        echo "\t*****************************************************
        ➤  ***\n"
fi
if [ -s /tmp/look_all.c${UNIQUE}3.$USER.$1 ]
    then
        SHOW=Y
        # Other sites have sent output to the result machine.
        echo "\n\t*************************************************
        ➤  ****"
        echo "\t*   The next sites aren't mentioned in the checkfile *"
        echo "\t*         $CHKFILE:\t     *"
        echo "\t*                                                  *"
        cat /tmp/look_all.c${UNIQUE}3.$USER.$1 | while read LINE
```

```
        do
            echo “\t*                -> $LINE     \t\t     *”
        done
        echo “\t*                                                    *”
        echo “\t*              Inform the administrator              *”
        echo “\t******************************************************
        ➤  **\n”
fi

exec >/dev/tty
if [[ $SHOW = Y ]]
    then
        # Show remarks above
        clear
        pg /tmp/look_all.pg$UNIQUE
        check_answer $1
        rm /tmp/look_all.pg$UNIQUE
        clear
fi
}

# Begin the main part of script.
scriptbegin $0
# Get unique trailer and site.
UNIQUE=‘date +”%H%M%S”‘
SITE=‘hostname‘
# Show filesystems with more than PRC used space.
PRC=90
# Set parameters to “No information required”
MAIL=N
DFI=N
DB=N
SAVE=N
SPACE=N
HELP=N
if [ $# -eq 0 ]
    then
        # No parameters: default = all
        MAIL=Y
        DFI=Y
        DB=Y
        SAVE=Y
        SPACE=Y
fi

while  [ $# -gt 0 ]
    do
        case $1 in
            mail ) MAIL=Y;;
            dfi  ) DFI=Y;;
```

```
            db   ) DB=Y;;
            save ) SAVE=Y;;
            space) SPACE=Y;;
            ?    ) HELP=Y;;
            *)     clear
                   echo “\n\nWrong parameter: $1 “
                   echo “\nUsing help: type look_all ?”
                   echo “\n\nPress <Enter> to continue”
                   read ANS ;;
        esac
        shift 1
    done

if [ $HELP = Y  ]
    then
        pg /home/data/look_all.hlp
fi

# Reset the interrupt keys (eg <Ctrl>+D and <Ctrl>+C).
set -o ignoreeof
trap  “/home/oper/look_all.ctrlc $UNIQUE” 2

if [ $MAIL = Y  ]
    then
        if [ -r /tmp/look_all.$USER.lokje ]
            then
                # Lockfile exists - a Unix user can’t request mail
                # more than once.
                echo “\nSomebody is making use of your user-id”
                echo “An AIX-user can request mail only once.”
                echo “If you repeat the request, the previous process
                ➤  is interrupted.”
                echo “\n\nPress <Enter> to continue”
                read ANS
            else
                # cleanup files
                >/tmp/look_all.$USER.lokje
                >/tmp/look_all.$USER.mail2
                >/tmp/look_all.$USER.mail5
                # Check we’ve got all mail.
                check_sites mail

                # Save to another location.
                if [ -s /var/spool/mail/$USER ]
                    then
                        cp /var/spool/mail/$USER
                        ➤  /tmp/gather_namail.$USER.$SITE.$UNIQUE
                fi

                # Mail empty?
```

```
ls -s /tmp/gather_mail* | grep " 0 " | awk -F.
➤  '{print $2}' >/tmp/look_all.$USER.mail2
fgrep "No mail for root" /tmp/gather_mail* | awk -F:
➤  '{print $1}' | awk -F. '{print $2}'
➤  >>/tmp/look_all.$USER.mail2

# Sort the sites
sort /tmp/look_all.$USER.mail2 -o
➤  /tmp/look_all.$USER.mail4
exec >/tmp/look_all.pg$UNIQUE
echo "\n\t*****************************************
➤  ***********"
echo "\t*     There is no mail for root in the next
➤  sites:  *"
cat /tmp/look_all.$USER.mail4 | while read LINE
do
    echo "\t*                  $LINE
    ➤  \t    *"
done
echo "\t*****************************************
➤  *********"

exec >/dev/tty
clear
pg /tmp/look_all.pg$UNIQUE
check_answer mail
rm /tmp/look_all.pg$UNIQUE

 # Add sites from which there isn't any mail to ones
 # from which mail was received and sort them.
 cat /tmp/look_all.$USER.mail4
 ➤  >>/tmp/look_all.c${UNIQUE}2.$USER.mail
 sort /tmp/look_all.c${UNIQUE}2.$USER.mail -o
 ➤  /tmp/look_all.$USER.mail4
 # Compare this file with the checkfile to retrieve
 # the name of the site.
 diff /tmp/look_all.$USER.mail4 $CHKFILE | grep ">"
 ➤  >/tmp/look_all.$USER.mail3
 cat /tmp/look_all.$USER.mail3 | while read
 ➤  FIRST_CHR SITE
     do
         echo $SITE >>/tmp/look_all.$USER.mail5
     done
 # Show the mail.
 for SITE in 'cat /tmp/look_all.$USER.mail5'
     do    MAIL=/tmp/gather_mail.$SITE
         echo $SITE >>/tmp/look_all.$USER.mail2
         cp $MAIL /var/spool/mail/$USER
         clear
         echo "Mail of $SITE:\n\n"
```

```
                        mail
                        check_answer mail
                    done

                # Show the mail present when look_all started-up
                for MAIL in `ls /tmp/gather_namail.$USER.$SITE.*
                ➤  2>/dev/null`
                    do
                        cp $MAIL /var/spool/mail/$USER
                        clear
                        echo "Press <ENTER> to see other mail of
                        ➤  $USER of $SITE:\n\n"
                        read ANS
                        check_answer mail
                        mail
                    done
                >/var/spool/mail/$USER

                # remove temp files and created mailbox
                rm /tmp/look_all.$USER.mail*
                rm /mbox 2>/dev/null

                # remove lock on mail and check_site files
                rm /tmp/look_all.$USER.lokje
                rm /tmp/look_all.c$UNIQUE*.$USER.mail
        fi
fi

if [ $DFI = Y ]
    then
        # Get the threshold amount of unused space (as a percentage).
        >/tmp/look_all.$UNIQUE.dfiFiltered
        clear
        echo "\n\t*************************************************
        ➤  ***" >/tmp/look_all.$UNIQUE.dfiPRC
        echo   "\t*   The free space in the filesystems is ok for:
        ➤  *" >>/tmp/look_all.$UNIQUE.dfiPRC
        echo "\n\n"
        echo "Show filesystems with used space greater than:    %
        ➤  (default $PRC%) \b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\c"
        read PROCENT
        check_answer dfi
        # If the user pressed the <Enter> key, use the default.
        if [ -n "$PROCENT" ]
            then
                PRC=$PROCENT
        fi

        if [ -f /tmp/look_all.dfi$PRC -a ! -f
        ➤  /tmp/look_all.dfiChecked$PRC ]
```

```
then
    pg /tmp/look_all.dfi$PRC
    cp /tmp/look_all.dfiFil$PRC
    ➤  /tmp/look_all.$UNIEK.dfiFiltered
else
    # Compare files received with the checkfile.
    check_sites dfI
    rm /tmp/look_all.c$UNIQUE*
    clear
    echo “Checking servers, please wait\n”
    for DFIK in ‘ls /tmp/gather_dfI*‘
        do
            SITE=‘echo $DFIK | cut -d”.” -f2‘
            >/tmp/look_all.dfI1.$UNIQUE
            echo “\tchecking $SITE”

            # Get filesystems to skip (static filesystems
            # etc).
            if [ -f /home/data/look_all.skipfs.$SITE ]
                then
                    VAR1=‘cat /home/data/look_all.
                    ➤  skipfs.$SITE‘
                else
                    VAR1=””
            fi

            # Get general filesystems that also have to
            # be skipped (eg databases, CDRFS, etc).
            cat /tmp/gather_dfI.$SITE | \
            grep -v -E “${VAR1}ora_|orasys|/db|/dev/cd0|
            ➤  %used|/infocd|Uitvoer|Filesystem “
            ➤  >>/tmp/look_all.dfI1.$UNIQUE
            cat /tmp/gather_dfI.$SITE | grep ora_arch \
            >>/tmp/look_all.dfI1.$UNIQUE

            # Make a list of filesystems skipped
            diff /tmp/gather_dfI.$SITE /tmp/look_all.
            ➤  dfI1.$UNIQUE | \
            grep “<“ | grep -v -E “Uitvoer|Filesystem|
            ➤  ora_arch” \
            >/tmp/look_all.dfI3.$UNIQUE

            # Produce the list the user requested
            exec >/tmp/look_all.dfI2.$UNIQUE
            echo “Filesystems of $SITE, that are using
            ➤  more than $PRC%:\n”
            echo “Filesystem               Total Kb
            ➤  Free Kb  %used”
            while read fs total used free used mounted
                do
```

```
                    PRO=‘echo $used | awk -F% ‘{ print
                    ➤  $1}’‘
                    if [ “$PRO” -gt “$PRC” ]
                        then
                            echo “$mounted $total $free
                            ➤  $used”| awk ‘{printf
                            ➤  (“%-25s %8s %8s
                            ➤  %4s”,$1,$2,$3,$4 “\n”)}’
                    fi
                done </tmp/look_all.dfI1.$UNIQUE

            if [ ‘cat /tmp/look_all.dfI2.$UNIQUE | wc -l ‘
            ➤  -eq 3 ]
                then
                    # Only header lines - clean-up file.
                    echo “\t*                  $SITE
                    ➤           \t
                    ➤  *”>>/tmp/look_all.$UNIQUE.dfiPRC
                    >>/tmp/look_all.$UNIQUE.dfiPRC
                else
                    cat /tmp/look_all.dfI2.$UNIQUE
                    ➤  >>/tmp/look_all.$UNIQUE.dfiGT
            fi
            exec >/tmp/look_all.dfI4.$UNIQUE
            echo “Skipped filesystems in $SITE:”
            while read char fs total used free used mounted
                do
                    echo “$mounted”
                done </tmp/look_all.dfI3.$UNIQUE
            cat /tmp/look_all.dfI4.$UNIQUE
            ➤  >>/tmp/look_all.$UNIQUE.dfiFiltered
            echo “\n\n\n”
            ➤  >>/tmp/look_all.$UNIQUE.dfiFiltered
            exec >/dev/tty
            rm /tmp/look_all.dfI*.$UNIQUE
        done
echo   “\t*
➤         *” >>/tmp/look_all.$UNIQUE.dfiPRC
echo “\t*******************************************
➤  *******”>>/tmp/look_all.$UNIQUE.dfiPRC
exec >/dev/tty
clear
pg /tmp/look_all.$UNIQUE.dfiPRC
check_answer dfi
clear
pg /tmp/look_all.$UNIQUE.dfiGT
check_answer dfi
clear
cp /tmp/look_all.$ UNIQUE.dfiPRC /tmp/look_all.dfi$PRC
echo “\n\n\n” >>/tmp/look_all.dfi$PRC
```

```
                cat /tmp/look_all.$ UNIQUE.dfiGT
                ➤  >>/tmp/look_all.dfi$PRC
                cp /tmp/look_all.$ UNIQUE.dfiFiltered
                ➤  /tmp/look_all.dfiFil$PRC
        fi
        echo “\n\nWould you like to see the skipped filesystems?
        ➤  (Y/N)\c”
        # Set variable to uppercase.
        typeset -u ans
        read ans
        if [ “$ans” = “Y” ]
            then
                pg /tmp/look_all.$UNIQUE.dfiFiltered
                check_answer dfi
        fi
        rm /tmp/look_all.$UNIQUE.dfi*
fi
if [ $DB = Y ]
    then
        clear
        echo “Press <ENTER> to see output of checking databases:”
        read ANS
        check_answer db

        check_sites cadb
        rm /tmp/look_all.c$UNIQUE*

        for CADB in ‘ls /tmp/gather_cadb*‘
            do
                SITE=‘echo $CADB | cut -d”.” -f2‘
                clear
                cat $CADB >>/tmp/look_all.cadb$UNIQUE
                echo “\n\n” >>/tmp/look_all.cadb$UNIQUE
            done
        pg /tmp/look_all.cadb$UNIQUE
        check_answer db
        rm /tmp/look_all.cadb$UNIQUE
fi

if [ $SAVE = Y ]
    then
        clear
        echo “Press <ENTER> to see output of saves:”
        read ANS
        check_answer save

        check_sites saves
        rm /tmp/look_all.c$UNIQUE*

        for SAVES in ‘ls /tmp/gather_saves*‘
```

```
            do
                SITE=‘echo $SAVES | cut -d”.” -f2‘
                clear
                echo “Save of: $SITE:\n”
                pg $SAVES
                check_answer save
            done
fi
if [ $SPACE = Y ]
    then
        clear
        echo “Press <ENTER> to see if there are any space
        ➤  problems:”
        read ANS
        check_answer space

        if [ ! -f /tmp/gather_space.* ]
            then
                echo “\tThere are no space problems.”
            else
                for SPACE in ‘ls /tmp/gather_space.*‘
                    do
                        clear
                        SITE=‘echo $SPACE | awk -F. ‘{print $2}’‘
                        if [ -f /home/data/look_all.space.$SITE ]
                            then
                                # An action is already started
                                # (eg by a disk)
                                cat $SPACE
                                ➤  >/tmp/look_all.space$UNIQUE
                                echo “\n”
                                ➤  >>/tmp/look_all.space$UNIQUE
                                # Explain to the user that this
                                # isn’t a problem.
                                cat /home/data/look_all.space.$SITE
                                ➤  >>/tmp/look_all.space$UNIQUE
                                echo “\n”
                                ➤  >>/tmp/look_all.space$UNIQUE
                                pg /tmp/look_all.space$UNIQUE
                                check_answer space
                                rm /tmp/look_all.space$UNIQUE
                            else
                                pg $SPACE
                                check_answer space
                        fi
                    done
        fi
        clear
fi
```

```
echo "\n"

scriptend $0
```

This article concludes in next month's issue of *AIX Update*.

Teun Post
Unix Specialist
Schuitema NV (The Netherlands)

Creating a fancy command line prompt

Have you ever wanted a fancy command-line prompt? Using the Korn shell, you'll find that creating one is fairly simple.

The environment variable *PS1* specifies the string to be used as the primary system prompt. By default, it's set to '$ ' for a non-root user. Would you like something more interesting and informative? Information such as the system's hostname, your username, the present working directory, and other textual information can all be displayed in your command-line prompt.

The system's hostname is already set in either the environment variable *HOST* or *HOSTNAME*, and it can also be retrieved by the command **uname -n**. Your username is in *$LOGNAME* and can similarly be retrieved using the command **logname**. The Korn shell automatically sets the variable *PWD* to point to the previous working directory set using the **cd** command. You can add a little spice to the output by using **tput** to display some of the information in reverse mode. Consider adding the following lines to your *.profile*:

```
export bold=‘tput smso‘
export norm=‘tput rmso‘
export HOSTNAME=‘uname -n‘
export LOGNAME=‘logname‘

export PS1=’
${bold} (${HOSTNAME}:${LOGNAME}) ${norm}
$PWD
> ‘
```

Rather than displaying all *$PWD* when you're in one of your lower-level directories, you can cut off the unnecessary home directory information using the Korn shell's parameter substitution. For instance, *$PWD* displays */home/customer/xephon/AIXupdate*, but, if you want to see only *customer/xephon/AIXupdate*, you can use the command *${PWD#$HOME/}* – it does the trick by deleting your home directory from *$PWD*'s absolute path.

Regretfully, AIX's Korn shell doesn't re-evaluate commands set in its system prompts. Setting *PS1* to ``date`` sets *PS1* to the date valid when the command is evaluated and then leaves it at that. Someday it might be possible for *PS1* to continue evaluating its command so it constantly shows the valid date. That, however, remains a future dream...

System Programmer (Switzerland)

Experiences with the AIX Software ServicePac

Last October I received a letter from IBM Global Services about their AIX Software ServicePac, a 'special IBM offering' that provides an easy way to keep AIX systems at the most current level. This interested me in as much as using a consistent method of preventative maintenance could reduce the likelihood of system problems or, even worse, crashes. Announcements about preventative maintenance always get my attention!

CONTENTS AND FUNCTIONS

The letter mentioned that I would receive the newest APARs (Authorized Problem Analysis Reports) and PTFs (Program Temporary Fixes) for the AIX operating system (Version 4.1.5 and higher) and all supported IBM AIX software products (except CATIA) on CD-ROM every month. Also available on the CD-ROMs are IBM technical databases, which include support documents and APAR descriptions that can be used for error analysis. Furthermore, other information on AIX and RS/6000 is included on the CD-ROMs. The functions of the

AIX Software ServicePac are accessible via a Java-based user-friendly graphical interface.

The available functions are:

- Display the most current APARs and PTFs classed by product group.
- Upgrade the system by product group.
- Search for updates by APARs, PTFs, or Fileset name.
- Install PTFs.
- Search the problem solving databases.

ORDERING

The AIX Software ServicePac can be purchased by annual subscription, either as a single item or in units of ten. Information on ordering and prices, as well as more general information, can be found on the ServicePac's homepage at *http://www.de.ibm.com/aixpac*. I've got to admit that I am used to Sun Microsystem's SunSolve offering, which is very similar to the ServicePac and also contains the latest operating system version updates, but is less expensive. Fortunately, one of my customers, who has a fairly large number of RS/6000s, also has a subscription to the ServicePac and asked me to evaluate it.

IMPRESSIONS

Before using the ServicePac, I installed it using the command **smitty install_latest**. This very simple procedure installs the AIX Software ServicePac as well as the Java Runtime Environment (if *Java.rte 1.1.2* or higher is not already installed). It then needs to be executed using the command **runASP**. Much to my chagrin, I had a **tty** device problem after installing a Y2K PTF. Rather then trying to locate the PTF needed, I just upgraded my system completely. Among other components, I clearly loaded the right PTF as the problem disappeared.

The AIX Software ServicePac can also be used to ensure Y2K compliance. More information about PTFs can be found at:

```
http://www.software.ibm.com/year2000/papers/aixy2k.html
```

CONCLUSION

I find IBM's AIX ServicePac somewhat expensive, especially when compared either with the downloadable fixes that are offered free of charge at the company's Boulder Web site (*service.boulder.ibm.com/service/rs6000*) or with Sun Microsystem's SunSolve. Nevertheless, its combination of problem-solving databases and fixes available in APAR and/or PTF form provide an additional degree of protection for systems in the form of preventative maintenance.

Werner Klauser (Switzerland)

New home page for *AIX Update*

We've restructured our Web site to give subscribers access to all relevant information directly from the AIX Update home page. In particular, subscribers can now download code from that page, rather than follow the general download path for all journals. The page also includes AIX product announcements and links to useful Web sites.

The address is www.xephon.com/aixupdate.html – use that address, in full, to go straight there, bypassing the Xephon home page.

AIX news

IBM has announced a new AIX Bonus Pack for AIX Version 4.3, which includes GraphOn GO-Joe Version 2.2.0.0 and AIX Fast Connect Version 2.1. GO-Joe allows Java-enabled devices or browsers to access X applications running on AIX hosts from any location, over the Internet, or via a dial-up connection. AIX Fast Connect supports OS/2 clients using LAN Server, DOS LAN Server, and DOS LAN Requestor 3.0, as well as Windows for Workgroups, Windows 9*x*, and NT. The Bonus Pack is available now.

The company also announced Data Path Optimizer, which uses redundant paths between a system server and the disk storage in a Versatile Storage Server to enable optimum availability and to distribute performance across the paths. The server-based product provides the intelligence to automatically reroute I/O operations when there's a failure.

Out now, prices start at US$14,000 for the AIX version.

For further information contact your local IBM representative.

* * *

Chili!Soft's ASP Developer Edition 3.0, which brings Microsoft's ASP Web server technology to platforms including AIX and Sun Solaris, is now available free by download. Chili!Soft's product allows ASP to run on a variety of Web servers from Netscape, Apache, and Lotus.

For further information contact:
Chili!Soft, Richards Road, Suite 103, Bellevue, WA 98005, USA
Tel: +1 425 957 1122
Fax: +1 425 562 9565
Web: http://www.chilisoft.com

* * *

Transition Networks' Conversion Center now supports NetView on AIX. Conversion Center is a multiprotocol and multimedia converter that may now be managed within the NetView framework. The 16-slot unit can convert protocols, including UTP to fibre conversions, for Ethernet, Fast Ethernet, ATM, FDDI, and Token Ring, as well as single-mode to multi-mode fibre conversions for these and Gigabit Ethernet and OC-12 environments. A redundant power option provides additional network and data protection.

The new support for management platforms is included without charge as part of the Management Module of the Conversion Center, which costs US$395.

For further information contact:
Transition Networks, 6475 City West Pkwy, Eden Prairie, MN 55344, USA
Tel: +1 612 941 7600
Fax: +1 612 941 2322
Web: http://www.transition.com

Wadsworth Electronics, Central Avenue, West Molesey, Surrey KT8 2QB, UK
Tel +44 181 268 7006
Fax: +44 181 979 3213