44 AIX

June 1999

In this issue

update

Published by

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

North American office

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

Contributions

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

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

Editor

Harold Lewis

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 (you'll need the user-id shown on your address label to access it).

Printed in England.

Displaying the AIX process hierarchy

INTRODUCTION

Tracing process hierarchy is important when it comes to killing processes for one reason or another. For a given parent process id, one needs to trace all its child processes in order to kill the parent process safely. It's not always the case that, if you kill a parent process, all its child processes are killed automatically. **ph** (process hierarchy) is a shell script that displays the process hierarchy for a given process. For a given process id, the shell script displays the hierarchy as well as all parent processes. Therefore, the script traces processes downwards (looking for children) and upwards (looking for parents).

NOTES

1 The script must be run in **ksh** as it requires array support.

2 The script can be run without modification as all temporary files are created in the */tmp* directory.

3 The script provides the option to print the displayed hierarchy.

4 The script can display a list of all the processes on the system while prompting for a process id to be entered.

5 In the displayed hierarchy, each level above or below level 0 is indented by 5 dots, as indicated in the sample output for process id '15776' (see page 16). The comments describing levels are included only for the purpose of this article – they do not appear in the actual output.

PH.SH

```
#!   /bin/ksh
################################################################################
#                                                                              #
# ph.sh - display process hierarchy.                                           #
#                                                                              #
# Displays the process hierarchy for a given process.                          #
#                                                                              #
```

```
# Notes   1   The script displays all the parents (parent,          #
#             grandparent, great grandparent, etc) in the           #
#             parent hierarchy.                                     #
#                                                                   #
#         2   The script displays all generations of children       #
#             (child, grandchild, great grandchild etc) in the      #
#             child hierarchy.                                      #
#                                                                   #
# Date          Author       Build                                  #
# ----------------------------------------------------------------- #
# 02/10/98      A Zaman      Initial                                #
#                                                                   #
#####################################################################

#####################################################################
#                                                                   #
# InitializeVariables                                               #
#                                                                   #
# This function initializes all variables.                          #
#                                                                   #
#####################################################################
InitializeVariables ( )
{
PID=                                # process id
PRPID=""                            # parent process id
CURPROC=$$                          # process running the script
PROC_FILE=/tmp/ph_1_$$.tmp          # all processes in the system
TEMP_FILE=/tmp/ph_2_$$.tmp
TEMP_FILE_1="${TEMP_FILE}_1"
LOV_FILE_1="/tmp/ph_lov_1.tmp"      # value files
LOV_FILE_2="/tmp/ph_lov_2.tmp"
DISPLAY_FILE=/tmp/ph_3_$$.tmp       # display file
DATETIME=`date "+%d/%m/%y at %H:%M:%S"`

# define arrays
LEVELO[0]=              # array for selected (or entered) process

CINDEX=0                # index for child array
PINDEX=0                # index for parent array

CHILDH[$CINDEX]=        # hierarchy for child processes
PARENTH[$PINDEX]=       # hierarchy for parent processes

SEC=0                   # success exit code
FEC=1                   # failure exit code

TRUE=0
FALSE=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

SLEEP_DURATION=3        # seconds for sleep command
ERROR="${RVON}${BON}ph.sh:ERROR:${BOFF}"
INFO="${RVON}ph.sh:INFO: "

# define messages
#
INTERRUPT="Program interrupted! Quitting...${RVOFF}"
WORKING="Working.................${RVOFF}"
PRINT_OK="Successfully submitted job for printing${RVOFF}"
PRINT_NOT_OK="Failed to submit job for printing${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
}

###############################################################################
#                                                                             #
# MoveCursor                                                                  #
#                                                                             #
# This function moves the cursor to location (Y, X)                           #
#                                                                             #
# Input   :   Y and X coordinates                                             #
#                                                                             #
# Notes   1   This function must run in ksh for print to work.                #
#             Print is used as echo doesn't 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`

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}
}

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

######################################################################
#                                                                    #
# ListAllProcesses                                                   #
#                                                                    #
# This function outputs process list to $PROC_FILE file.             #
#                                                                    #
# Notes   1   The process hierarchy comes from a snapshot of         #
#             processes held in the file $PROC_FILE.                 #
#                                                                    #
######################################################################
ListAllProcesses ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
> ${PROC_FILE}
```

```
# get all process details
ps -eaf > ${TEMP_FILE}

# format details
cat ${TEMP_FILE} | cut -c 1-10,11-16,17-20,47-70 > ${TEMP_FILE_1}
cat ${TEMP_FILE_1} | awk {'print "uid="$1" pid="$2" ppid="$3"
➤  com="$4'} > ${PROC_FILE}
}

#############################################################################
#                                                                           #
# DisplayListOfValues                                                       #
#                                                                           #
# This function diplays a list of all the processes for the user    #
# to select from.                                                           #
#                                                                           #
# Notes   1   Assigns the selected process id to $IPID.             #
#                                                                           #
#         2   Processes displayed are in the list of processes      #
#             in the file $TEMP_FILE.                               #
#                                                                           #
#############################################################################
DisplayListOfValues ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

# write header in LOV (list if values) file
echo " List of all processes on ${DATETIME} "                 >
➤  ${LOV_FILE_1}
echo " ============================================="          >>
➤  ${LOV_FILE_1}
echo "   Select a process Id by deleting corresponding line " >>
➤  ${LOV_FILE_1}
echo "   in vi and saving the file "                          >>
➤  ${LOV_FILE_1}

# append process details
cat ${TEMP_FILE}  >> ${LOV_FILE_1}
cat ${LOV_FILE_1} >> ${LOV_FILE_2}

# dispplay LOV file
vi ${LOV_FILE_2}

# assign the selected process id to IPID
IPID=`diff ${LOV_FILE_1} ${LOV_FILE_2} | tail -1 | awk {'print $3'}`
}

#############################################################################
#                                                                           #
# GetProcessId                                                              #
```

```
#                                                                              #
# This function gets a process id from the user.                               #
#                                                                              #
#  Notes   1   The captured process id is assigned to $IPID.                   #
#                                                                              #
################################################################################
GetProcessId ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
while true
do
    clear
    echo "Enter process id(l=list of values q=quit):\c"
    read IPID
    case $IPID in
         "") : ;;
          l) DisplayListOfValues ;
             if [ "${IPID}" = "" ]
             then
                 : ;
             else
                 break ;
             fi ;;
        q|Q) ProcessExit $SEC ;;
          *) break ;;
    esac
done

# get the command associated with this process id
COM=`cat ${PROC_FILE} | grep " pid=$IPID " | grep -v grep |
➤  grep -v $CURPROC | cut -d' ' -f4 | cut -d'=' -f2`

# write selected process details into array
LEVELO[0]="Pid=$IPID Command=$COM  <---- selected process id"
}

################################################################################
#                                                                              #
# GetParentProcessIds                                                          #
#                                                                              #
# This function gets the parent process id and the associated                  #
# command for a given process id.                                              #
#                                                                              #
# Notes   1   The function calls itself recursively.                           #
#                                                                              #
################################################################################
GetParentProcessIds ( )
{
while true
do
```

```
    trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

    PRPID=`cat ${PROC_FILE} | grep " pid=$PID " | grep -v grep | \
           grep -v $CURPROC | cut -d' ' -f3 | cut -d'=' -f2`

    COM=`cat ${PROC_FILE} | grep " pid=$PRPID " | grep -v grep | \
         grep -v $CURPROC | cut -d' ' -f4 | cut -d'=' -f2`
    if [ "${PRPID}" = "" ]
    then
        break
    else
        PID=$PRPID
        PARENTH[$PINDEX]="${LEVEL}PPid=$PRPID Command=$COM"
        LEVEL="$LEVEL....."
        PINDEX=`expr $PINDEX + 1`
        GetParentProcessIds
    fi
done
}

#############################################################################
#                                                                           #
# ProcessChildIds                                                           #
#                                                                           #
# This function gets all child process ids for a given process id.          #
#                                                                           #
# Notes   1   The function calls itself recursively to build the            #
#             child hierarchy.                                              #
#                                                                           #
#############################################################################
ProcessChildIds ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

INDEX=0                         # initialize to 0
OUTFILE="/tmp/phl$LEVEL.tmp"    # output file for current level

cat ${PROC_FILE} | grep " ppid=$PID " | grep -v "grep" | \
    grep -v "$CURPROC" > ${OUTFILE} 2>&1

cat ${OUTFILE} | while read LINE
do
    PID=`echo $LINE | cut -d' ' -f2 | cut -d'=' -f2`
    COM=`echo $LINE | cut -d' ' -f4 | cut -d'=' -f2`
    INDENT=
    while [ $INDEX -lt $LEVEL ]
    do
        INDENT="$INDENT....."
        INDEX=`expr $INDEX + 1`
    done
```

```
    CHILDH[CINDEX]="${INDENT}Pid=$PID Commmand=$COM"
    CINDEX=`expr $CINDEX + 1`
    LEVEL=`expr $LEVEL + 1`
    ProcessChildIds               # recursive call
done

# finished processing the current level, return to previous level
LEVEL=`expr $LEVEL - 1`
return $TRUE
}

##############################################################################
#                                                                            #
# BuildProcessHierarchy                                                      #
#                                                                            #
# This function builds the process hierarchy in arrays $PARENTH     #
# and $CHILDH and writes them into $DISPLAY_FILE file.               #
#                                                                            #
# Notes   1   The function calls following functions:                #
#             - GetParentProcesIds                                           #
#             - GetChildProcesIds                                            #
#                                                                            #
##############################################################################
BuildProcessHierarchy ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

DisplayMessage I "${WORKING}"

# build hierarchy for parents
PID=${IPID}       # set selected process id
PINDEX=0          # set array index to 0
LEVEL="....."     # set level marker for level 1
GetParentProcessIds

# build hierarchy for child process
PID=${IPID}       # set starting parent process id
LEVEL=1           # set level marker for level 1
CINDEX=0          # set index to child hierarchy array to 0
ProcessChildIds

# print parent hierarchy in reverse order (last parent on top)
echo "Process hierarchy for process Id ($IPID) on $DATETIME"
➤  > $DISPLAY_FILE
echo "================================================================="
➤  >> $DISPLAY_FILE
while true
do
    echo "${PARENTH[$PINDEX]}" >> $DISPLAY_FILE
    if [ $PINDEX -eq 0 ]
```

```
    then
        break
    else
        PINDEX=`expr $PINDEX - 1`
    fi
done
echo "     Parent Hierarchy  " >> $DISPLAY_FILE
echo "     ^  " >> $DISPLAY_FILE
echo "     |  " >> $DISPLAY_FILE
echo "     |  " >> $DISPLAY_FILE

# print level 0 (process id selected)
echo  "${LEVEL0[0]}" >> $DISPLAY_FILE
echo "     Child Hierarchy  " >> $DISPLAY_FILE
echo "     |  " >> $DISPLAY_FILE
echo "     |  " >> $DISPLAY_FILE
echo "     v  " >> $DISPLAY_FILE

# print child hierarchy (immediate child first)
CINDEX=0
while true
do
    if [ "${CHILDH[$CINDEX]}" = "" ]
    then
        break
    fi
    echo "${CHILDH[$CINDEX]}" >> $DISPLAY_FILE
    CINDEX=`expr $CINDEX + 1`
done
}

###############################################################################
#                                                                             #
# ProcessExit                                                                 #
#                                                                             #
# This function removes all temporary files and makes a graceful    #
# exit using the exit code passed as a parameter.                    #
#                                                                             #
###############################################################################
ProcessExit ( )
{
EXIT_CODE=$1
rm /tmp/ph*.tmp*
clear
exit $EXIT_CODE
}

###############################################################################
#                                                                             #
# ViewAndPrintProcessHierarchy                                                #
```

```
#                                                                              #
# This function is used to view and print the process hierarchy.    #
#                                                                              #
################################################################################
ViewAndPrintProcessHierarchy ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

# view hierarchy
view  ${DISPLAY_FILE}

# print hierarchy
while true
do
    clear
    echo "Do you wish to print the process hieracrhy(Y/N)?:\c"
    read REPLY
    case $REPLY in
        n|N) return $TRUE ;;
        y|Y) break ;;
          *) : ;;
    esac
done

# get the printer name
while true
do
    clear
    echo "Enter printer name for lp command(q to quit):\c"
    read PRINTER
    case $PRINTER in
         "") : ;;
        q|Q) break ;;
          *) lp -d$PRINTER $DISPLAY_FILE > /dev/null 2>&1 ;
             if [ $? -eq 0 ]
             then
                 DisplayMessage I "${PRINT_OK}" ;
                 break ;
             else
                 DisplayMessage E "${PRINT_NOT_OK}" ;
             fi ;;
    esac
done
}

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

```
#                                                                              #
################################################################################
main ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP

InitializeVariables
ListAllProcesses
GetProcessId
BuildProcessHierarchy
ViewAndPrintProcessHierarchy
ProcessExit  $SEC
}

# invoke main
main
```

SAMPLE OUTPUT FOR PROCESS ID 0

```
Process hierarchy for process Id (0) on 17/10/98 at 16:15:12
============================================
     Parent Hierarchy
     ^
     |
     |
Pid=0 Command=  <---- selected process id
     Child Hierarchy
     |
     |
     v
.....Pid=1 Commmand=/etc/init
..........Pid=3192 Commmand=/usr/sbin/syncd
..........Pid=3910 Commmand=/usr/lib/errdemon
..........Pid=5166 Commmand=/usr/sbin/srcmstr
..............Pid=3832 Commmand=/usr/lpp/smw/bin/smwd
..............Pid=5748 Commmand=/usr/sbin/biod
..............Pid=5968 Commmand=/usr/sbin/syslogd
..............Pid=6708 Commmand=sendmail:
..............Pid=6966 Commmand=/usr/sbin/portmap
..............Pid=7224 Commmand=/usr/sbin/inetd
.................Pid=22242 Commmand=telnetd
.......................Pid=21272 Commmand=-ksh
.................Pid=25058 Commmand=telnetd
.......................Pid=31296 Commmand=-ksh
..............Pid=7482 Commmand=/usr/sbin/snmpd
..............Pid=7740 Commmand=/usr/sbin/dpid2
..............Pid=7998 Commmand=/usr/sbin/muxatmd
..............Pid=9038 Commmand=/usr/sbin/rpc.statd
..............Pid=9296 Commmand=/usr/sbin/rpc.lockd
```

```
..............Pid=9808 Commmand=/usr/sbin/qdaemon
..............Pid=10070 Commmand=/usr/sbin/writesrv
..............Pid=10324 Commmand=/usr/dt/bin/dtlogin
..................Pid=11360 Commmand=/usr/lpp/X11/bin/X
..................Pid=12128 Commmand=dtlogin
......................Pid=11628 Commmand=dtgreet
.........Pid=6192 Commmand=./sapd
.........Pid=6450 Commmand=./npsd
.........Pid=8540 Commmand=/usr/sbin/cron
.........Pid=9556 Commmand=/usr/sbin/uprintfd
.........Pid=10580 Commmand=/usr/sbin/getty
.........Pid=11878 Commmand=/usr/sbin/getty
.........Pid=12902 Commmand=/usr/lpp/diagnostics/bi
.........Pid=13168 Commmand=./pciconsvr.ip
.........Pid=13438 Commmand=ora_smon_ecat6
.........Pid=13734 Commmand=ora_dbwr_ecat6
.........Pid=13938 Commmand=./pcimapsvr.ip
.........Pid=14996 Commmand=ora_reco_ecat2
.........Pid=15514 Commmand=/disk01/home/oracle/pro
.........Pid=16258 Commmand=ora_lgwr_ecat6
.........Pid=17290 Commmand=ora_reco_ecat6
.........Pid=17550 Commmand=ora_lgwr_ecat4
.........Pid=17808 Commmand=ora_pmon_ecat2
.........Pid=22748 Commmand=ora_reco_bor4
.........Pid=24554 Commmand=ora_lgwr_bor4
.........Pid=26198 Commmand=ora_pmon_ecat6
.........Pid=26464 Commmand=ora_smon_bor4
.........Pid=27924 Commmand=ora_lgwr_ecat2
.........Pid=28494 Commmand=ora_dbwr_bor4
.........Pid=30970 Commmand=ora_smon_ecat2
.........Pid=31228 Commmand=ora_reco_ecat4
.........Pid=31996 Commmand=ora_smon_ecat4
.........Pid=32044 Commmand=ora_pmon_bor4
.........Pid=32294 Commmand=ora_dbwr_ecat2
.........Pid=33826 Commmand=ora_dbwr_ecat4
.........Pid=34506 Commmand=ora_pmon_ecat4
```

SAMPLE OUTPUT FOR PROCESS ID 1

```
Process Hierarchy for Process Id (1) on 17/10/98 at 16:15:48
============================================
.....PPid=0 Command=
     Parent Hierarchy
     ^
     |
     |
Pid=1 Command=/etc/init  <---- selected process id
     Child Hierarchy
     |
```

```
     |
     v
.....Pid=3192 Commmand=/usr/sbin/syncd
.....Pid=3910 Commmand=/usr/lib/errdemon
.....Pid=5166 Commmand=/usr/sbin/srcmstr
..........Pid=3832 Commmand=/usr/lpp/smw/bin/smwd
..........Pid=5748 Commmand=/usr/sbin/biod
..........Pid=5968 Commmand=/usr/sbin/syslogd
..........Pid=6708 Commmand=sendmail:
..........Pid=6966 Commmand=/usr/sbin/portmap
..........Pid=7224 Commmand=/usr/sbin/inetd
..............Pid=22242 Commmand=telnetd
..................Pid=21272 Commmand=-ksh
..............Pid=25058 Commmand=telnetd
..................Pid=31296 Commmand=-ksh
..........Pid=7482 Commmand=/usr/sbin/snmpd
..........Pid=7740 Commmand=/usr/sbin/dpid2
..........Pid=7998 Commmand=/usr/sbin/muxatmd
..........Pid=9038 Commmand=/usr/sbin/rpc.statd
..........Pid=9296 Commmand=/usr/sbin/rpc.lockd
..........Pid=9808 Commmand=/usr/sbin/qdaemon
..........Pid=10070 Commmand=/usr/sbin/writesrv
..........Pid=10324 Commmand=/usr/dt/bin/dtlogin
..............Pid=11360 Commmand=/usr/lpp/X11/bin/X
..............Pid=12128 Commmand=dtlogin
..................Pid=11628 Commmand=dtgreet
.....Pid=6192 Commmand=./sapd
.....Pid=6450 Commmand=./npsd
.....Pid=8540 Commmand=/usr/sbin/cron
.....Pid=9556 Commmand=/usr/sbin/uprintfd
.....Pid=10580 Commmand=/usr/sbin/getty
.....Pid=11878 Commmand=/usr/sbin/getty
.....Pid=12902 Commmand=/usr/lpp/diagnostics/bi
.....Pid=13168 Commmand=./pciconsvr.ip
.....Pid=13438 Commmand=ora_smon_ecat6
.....Pid=13734 Commmand=ora_dbwr_ecat6
.....Pid=13938 Commmand=./pcimapsvr.ip
.....Pid=14996 Commmand=ora_reco_ecat2
.....Pid=15514 Commmand=/disk01/home/oracle/pro
.....Pid=16258 Commmand=ora_lgwr_ecat6
.....Pid=17290 Commmand=ora_reco_ecat6
.....Pid=17550 Commmand=ora_lgwr_ecat4
.....Pid=17808 Commmand=ora_pmon_ecat2
.....Pid=22748 Commmand=ora_reco_bor4
.....Pid=24554 Commmand=ora_lgwr_bor4
.....Pid=26198 Commmand=ora_pmon_ecat6
.....Pid=26464 Commmand=ora_smon_bor4
.....Pid=27924 Commmand=ora_lgwr_ecat2
.....Pid=28494 Commmand=ora_dbwr_bor4
.....Pid=30970 Commmand=ora_smon_ecat2
```

```
.....Pid=31228 Commmand=ora_reco_ecat4
.....Pid=31996 Commmand=ora_smon_ecat4
.....Pid=32044 Commmand=ora_pmon_bor4
.....Pid=32294 Commmand=ora_dbwr_ecat2
.....Pid=33826 Commmand=ora_dbwr_ecat4
.....Pid=34506 Commmand=ora_pmon_ecat4
```

SAMPLE OUTPUT FOR PROCESS ID 15776

```
Process Hierarchy for Process Id (15776) on 17/10/98 at 16:17:21
================================================
.........................PPid=0 Command=
.....................PPid=1 Command=/etc/init
................PPid=5166 Command=/usr/sbin/srcmstr
..........PPid=7224 Command=/usr/sbin/inetd  #  ----------- Level 2
.....PPid=16786 Command=telnetd              #  ----------- Level 1
     Parent Hierarchy
     ^
     |
     |
Pid=15776 Command=-ksh  <---- selected process id  #  ----- Level 0
     Child Hierarchy
     |
     |
     v
.....Pid=19854 Commmand=-ksh             #  -------------  Level 1
..........Pid=14488 Commmand=sqlplus     # --------------  Level 2
................Pid=27424 Commmand=oracleecat2
```

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

New RS/6000 hardware and software features

This article describes various additions and enhancements to RS/6000 hardware and related software announced by IBM since the company's last major announcement in autumn 1998 (see *AIX Update*, Issue 39).

RS/6000 ENTERPRISE SERVER H70

The Enterprise Server model H70 is the first truly low-cost 64-bit server in IBM's RS/6000 range. It is packaged as a rack-mounted unit with an eight-EIA drawer cabinet, and is similar in design to the model H50, a 32-bit server that has been available for a while. The system can be configured with one to four RS64II processors mounted on two processor cards. Each processor is configured with a 64 KB L1 data cache, a 64 KB L1 instruction case, and a 4 MB L2 cache. The processors used are 340 MHz versions of the 252 MHz processor in the model S7A server. The system can be configured with up to 8 GB of SDRAM memory on two memory cards using 32 MB, 128 MB, and 256 MB DIMMs.

The H70 has 16 media bays, three of which are taken by a 32x-speed CD-ROM drive, a 1.44 MB 3.5-inch diskette, and a SCSI system disk. An additional bay is available for an optional disk that can either back up or mirror a system disk. Ultra SCSI or SSA internal disks can be installed in the twelve additional disk bays. These disks can be 4.5 GB or 9.1 GB units, providing a maximum internal disk capacity of up to 127.4 GB (including two disks installed in device bays). The system has one integrated Ethernet adapter that operates at both 10 and 100 Mbps, and also has two integrated Ultra SCSI adapters.

Eight PCI slots are available for 32-bit and 64-bit I/O graphics and communication cards. The system has four independent PCI buses providing two I/O slots each; one of the buses operates at 50 MHz and supports only 64-bit cards, while the other operates at 33 MHz and supports only 32-bit cards. Two other buses operate at 33 MHz and can accommodate one 32-bit and one 64-bit card. The computer has three serial ports and one parallel port, as well as a keyboard and a mouse port. Power supplies available are 220 volts AC and 48 volt DC, with an optional redundant hot-swappable second power supply. The service processor, which comes as standard, provides the following functions:

- Environmental monitoring and alerting for AC/DC voltage, fan speed, and temperature.
- Early power-off warning and error log analysis and alerts.

- Auto dial-out call placement for system operators and IBM personnel.
- Automatic reboot after a power loss or hardware checkstop failure, in the event of a machine check interrupt, and after an operating system failure.

A field upgrade from a model H50 is available, and this permits the re-use of memory DIMMs, hard disks, hard disk backplanes, internal media devices, PCI adapters, and RS/6000 racks.

The table below shows the performance of the H70 compared with that of the H50 using relative OLTP performance estimates for commercial processing.

	H50	H70	% improvement
1-way	10.0	16.6	66%
2-way	17.9	31.7	77%
3-way	25.2	44.2	75%
4-way	32.8	56.7	73%

RS/6000 HA-H70 CLUSTER SERVER

The HA-H70 cluster server is a pre-packaged highly available system that comprises two model H70 servers and a 7133 model D40 serial disk system in a model S00 rack. The computers are configured with redundant power supplies, dual boot disks, dual advanced serial RAID PCI SSA disk adapters, and dual LAN adapters. The disk subsystem contains eight 4.5 GB, 9.1 GB, or 18.2 GB disks and has dual data paths and a redundant power supply. This system runs AIX and version 4.3 of IBM's HACMP software, which is responsible for guaranteeing the high availability of applications by monitoring and detecting system hardware and software failures, including those resulting from the processor, network, adapter, power supply, and applications, and also those resulting from user-defined events. In the case of failure, HACMP initiates a sequence of pre-defined recovery actions to fail-over to a designated back-up component or server. HACMP also facilitates the graceful recovery and re-integration of a 'failed' server back into HA-H70 cluster while the cluster system is operational.

The system is expandable and is designed for easy growth. In addition to server scalability, it is also possible to install additional disks in the 7133-subsystem supplied, providing a maximum capacity of 291.2 GB per subsystem. An additional five 7133 drawers can be connected to the system, yielding a total capacity of no less than 1.74 TB.

IBM has published the following estimates of system availability: 99.999% (less than six minutes unplanned annual downtime) for systems running AIX 4.3 and HACMP 4.3, and 99.998% (less than eleven minutes unplanned annual downtime) for systems running AIX 4.3, HACMP 4.3, and DB2 UDB EE version 5.

IBM provides HACMP scripts for popular databases and applications such as DB2, Oracle, Informix, BAAN, and SAP, via the Web.

IBM ADVANCED SERIALRAID ADAPTER

The Advanced SerialRAID Adapter supports high-performance Serial Storage Architecture (SSA) devices with bandwidths of up to 160 MB/sec. This adapter allows users to connect up to 96 SSA disk drives in two loops to the system, or to up to 1.7 TB of disk storage per adapter. The adapter contains 64 MB of read cache and can be configured with an optional 32 MB of Fast/Write cache (supported only on single-host configurations). The adapter supports the following clustering configurations: eight-way JBOD (Just Bunch Of Disks) and two-way RAID 5. Up to 32 RAID 5 arrays can be configured per adapter, comprising three to 16 drives each. Optional hot-spare disks can back up multiple RAID arrays located on the same loop. RAID 0 (mirroring) is supported by adapter hardware and is available in single-host configuration only.

The adapter is compatible with previous adapters and disks operating at speeds of 80 MB/sec. However, to reach its full capacity, it must be connected to 7133 model D40 or T40 disk drives, which were announced in March.

Early tests performed with this adapter report substantial improvements in I/O throughput using both new and existing SSA disks. This improvement is attributed to improved design and a larger read cache.

PCI 3-CHANNEL ULTRA SCSI RAID ADAPTER

The Ultra SCSI RAID adapter is supported on RS/6000 models 150, 260, F50, H50, and H70. This adapter enables ULTRA SCSI support, at speeds up to 40 MB/sec, using appropriate internal or external disks. Two external and one internal Ultra SCSI ports are provided. As a result of its standard 32 MB Fast/Write cache, higher performance is anticipated even when the adapter operates with non-Ultra enabled disks, such as SCSI-2 F/W disks.

GXT2000P GRAPHICS ACCELERATOR

The GXT2000P replaces the GXT500P and GXT550P series adapters, and also the entry-level 3D graphics accelerator installed in 43P model 140, 150, and 260 workstations. The adapter is packaged as a half-length PCI card and is optimized for PHIGS, OpenGL, 2D, and video work. The following is the list of its major characteristics:

- 32 MB unified frame buffer configurable as:
  - Double buffered 24-bit colour
  - 24-bit ‘Z buffer’
  - 8-bit overlay buffer
  - Up to 16 MB local texture memory
  - Up to 1920x1200 resolution
- Four hardware colour maps
- 4-bit stencil buffer
- 4-bit window ID buffer
- Trilinear texture mapping
- Face culling support.

The performance of the adapter, using industry-standard 3D benchmarks, is up to three times that of GXT500P. Additionally, it features excellent 2D performance (43.1 Xmark93 on the 43P 150 and 42.61 Xmark93 on the 43P 260). With a lower price than the model GXT255P, this adapter effectively replaces the GXT255P, GXT500P,

GXT550P, and GXT800P adapters on RS/6000 workstations.

IBM 7206 MODEL 110 12 GB 4MM DDS-3 EXTERNAL TAPE DRIVE

The 7206 model 110 is a stand-alone SCSI 4 mm streaming tape drive. Using Digital Audio Tape (DAT) technology, it provides a media capacity of 12 GB (24 GB compressed). The tape has a sustained data transfer rate of 1.1 MB/sec (2.2 MB/sec with compression). As well as being compatible with DDS-3 formatted cartridges, the 7206 model 110 is also compatible with DDSIII and DS-2 tape cartridges. This model advances previously available 4 mm technology in terms of capacity, speed of data transfer, and a number of other important features, such as self-cleaning and energy conservation.

AIX 4.3.2 ENHANCEMENTS

The following enhancements were announced for AIX 4.3 and are available on new orders of AIX 4.3.2 or as an update to AIX 4.3 on CD-ROM.

INTERNET SECURITY ARCHITECTURE KEY MANAGEMENT PROTOCOL (ISAKMP)

ISAKMP, also know as Internet Key Exchange (IKE), allows users to negotiate security parameters with other hosts and to refresh security-associated information (algorithms, keys, etc) when setting up secure IP channels. Pre-shared key authentication is supported, as well as user-defined time and size limits for information refresh. A Web-based System Management GUI, currently available only in English, handles the configuration of IKE. This feature is currently available only for IPv4.

IPV6 GATEWAY CAPABILITY

The ability of AIX to act as a gateway between IPv4 and IPv6 has been extended by bundling Merit's GateD V6.0 package. This product provides gateway routing functions for RIP, EGP, BGP, HELLO, OSPF, and SNMP and is limited to unicast routing. Currently supported protocols are RIP Versions 1 and 2, OSPF Version 2, ISIS, Hello, and

SLSP. Inter-domain routing is provided by **gated** using BGP Versions 2, 3, or 4, and EGP. SNMP support is based on ISODE Version 8, through the SMUX protocol.

IPv6 is not available for either SP systems or models S70 or S7A systems attached to an SP.

AIX FAST CONNECT FOR WINDOWS AND OS/2

AIX Fast Connect for Windows and AIX Fast Connect for OS/2 enable basic file and printer sharing between AIX and Windows, allowing computers operating under AIX to participate in Microsoft's Network Neighborhood. AIX Fast Connect enables PCs running Windows to utilize the following resources on AIX-based systems:

- AIX Journaled File System (JFS)
- CD File System (CDFS)
- Network File System (NFS)
- Print services.

The following features are supported by AIX Fast Connect for Windows:

- Support for clients running Windows for Workgroups (WfW), Windows 95, Windows 98, and Windows NT.
- Viewing of shared AIX files and print services (CIFS client resource browsing).
- Support for long file names.
- Execution of programs written for the NetBIOS API (RFC 1001/1002).
- Use of TCP/IP's domain name system to resolve NetBIOS machine names.
- Support for Microsoft WINS Server.
- Support for 'opportunistic locking'.
- Unicode user, file, and printer name support.

- AIX authentication/authorization with passwords encrypted using 56-bit encryption.
- Interaction with NT 4 Server for:
  - User authentication and authorization.
  - Use of NT Server's domain master browser to find and publish shared resources across TCP/IP subnets.
  - Use of NT Server's Windows Internet Name Service (WINS) to resolve NetBIOS machine names.
- IBM service and support.
- HTML-based documentation (in English only).
- Command line interface for NetBIOS NET commands, SMIT, and Web-based system management.

AIX Fast Connect is based on AIX 4.3.2 and provides at least twice the performance of the previously available PC collaboration product, AIX Connections. AIX Fast Connect utilizes the latest TCP/IP **sendfile** API, which uses an in-kernel network file cache for improved performance. Note that, in order to provide Microsoft Master Browser support, it is necessary to have at least one Windows NT Server in the network.

AIX Fast Connect for OS/2 includes all functions available in AIX Fast Connect for Windows, additionally supporting OS/2 clients using LAN Server, DOS LAN Server 2.0, or DOS LAN Requester 3.0.

AIX SECURITY CERTIFICATIONS

IBM has announced that AIX has been granted two important levels of security certification: C2 and B1. To support this level of security it is necessary to specify special releases of the operating system that are available from IBM as PRPQs.

The operational subset of AIX 4.3.1 has been certified at the US Government C2 level, becoming the first 64-bit Unix operating system to meet this classification. The price of the corresponding PRPQ does not depend on the class of the machine.

B1 functionality was achieved by a joint team of IBM and Groupe Bull engineers using Bull's B1/EST-X 2.0.1 product under the Common Criteria 2.0. EAL4 evaluation has been performed by IABG in Germany. Version 2.0.2A of Groupe Bull's B1/EST-X, which supports AIX 4.3.2, is offered as IBM's PRPQ with a price that depends on the class of the machine.

AIX 4.3 AND 4.2 BONUS PACK ENHANCEMENTS

The contents of the AIX Bonus Pack CD, which ships with every copy of AIX 4.2.1 and 4.3, has been refreshed, and the following products have been updated to their latest release:

- Java Multimedia Framework V1.1.0.1 on AIX (JMF)
- VisualAge for Java Entry V2.0.0.3 (which is Euro ready)
- Netscape Navigator V4.0.8
- Ultimedia Services V2.2.1.2.

In addition, the following products have been added to the CD:

- A 30-day trial version of Oracle8 Server database.
- The Kernel Group's ZeroFault Dynamic Debugger V2.3 Evaluation Software, which uses advanced virtual machine technology to detect and report memory utilization errors and leaks in any AIX executable, without requiring recompiling, relinking, or access to the source code.
- Geodesic's Great Circle V3.1 (30-day 'try-and-buy'), which is a collection of C and C++ debugging libraries that link to object code and use garbage collection technology to find and eliminate memory leaks and errors in an application's code automatically.

MCAD OPERATING ENVIRONMENTS FOR AIX

'MCAD Operating Environments for AIX' is a set of CDs that contain all the software needed for the installation of the following pre-packaged MCAD applications on AIX: CATIA, I-DEAS Master Series, and Pro/ENGINEER. Support for I-DEAS Master Series has

been enhanced to allow the downloading of updates from the Web. The CATIA Operating Environment, V2.2.1 has been enhanced to support CATIA versions 5.1, 4.2.1, 4.2.0, and 4.1.9 as well as CATWeb Version 2.2, the GXT2000P graphics accelerator, the model 051 Spaceball, and installation assistance for AIX Fast Connect.

CHECK POINT TECHNOLOGIES FIREWALL-1 VERSION 4 FOR AIX

IBM has announced the availability of Check Point Technologies' FireWall-1 Version 4 for AIX 4.2.1 and AIX 4.3.1. While FireWall-1 is one of the leading products in the area of network security, previous versions of the product did not support the latest AIX versions and, consequently, the latest RS/6000 hardware. The new version adds support for the following features:

- Virtual Private Networks (VPNs), including full IPSec encryption with IKE key exchange and support for Entrust Public Key Infrastructure (PKIs).
- Worldwide 40-bit encryption.
- Network Address Translation for all H.323 applications and support for SSL gateway connections.
- User authentication using Web-based Client Authentication and Automatic Client Authentication Sign-off.
- Performance enhancements through support for multiple processors (SMPs).

REFERENCES

1 IBM Announcement Letters at *http://www.ibmlink.ibm.com*.

System Engineer (Israel)

Implementing DCE for AIX (part 2)

This month's instalment concludes this article on implementing DCE on AIX.

If you haven't already set up an AIX logical volume to be converted into an LFS aggregate, you can do so now. Choose the menu option *Export an Aggregate from the Local Machine*. You have to provide the system with a name and an ID for the aggregate. The ID must be unique among all JFS filesystems and LFS aggregates exported from this machine. It may be simpler to leave the field blank, in which case the next available ID is used.

Once the aggregate is created, you may then create filesets to enable your clients to access and create DFS data. The user and group that owns the root directory of a newly created fileset is always the DCE root principal and the DCE system group. The access rights initially assigned are *rwx*------ (700), with no explicit DCE ACL settings. The default quota limit is five megabytes, which you may change.

AIX CLIENTS

You don't need to order separate client software for AIX, as DCE client software is part of the AIX operating system. In order to install it on a client running AIX, use the **installp** command or **smit** using the command:

```
smit install_latest
```

You need to select the components *dce.client* and *dce.compat*, which contain the DCE Client Tools and the DFS Client Services, then start the installation.

After this administrator portion, logon as *root* and start **smit** using the command **smit mkdceclient**. Then choose the option *Local Only Configuration*. You have to type the name of the cell and the hostname of the master Security Server. The program enables you to activate an RPC Endpoint Mapper, a Security Client, a CDS Client, a DTS Client, and/or a DFS client. DTS and DFS are optional clients.

STANDARD ADMINISTRATION

Administration can be performed from any machine in the cell. You have a relatively free choice of administration platform and administrative software. Administrative rights are granted via the relevant ACLs.

Several programs can be used to administer the DCE environment. 'Standard Administration Tools' is a set of tools provided with just about every implementation of DCE. The most important component is the DCE Control Program or **dcecp**. This has a command line interface and is almost identical on all platforms on which DCE is available. The Control Program lets you configure DCE, browse information, change existing configurations, and perform many other administrative tasks.

Also worth mentioning are **rgy_edit** for administering Security Server objects and **cdscp** for administering the Directory Server.

COMMAND LINE

The DCE Control Program (**dcecp**) allows you to enter administrative commands directly. Many administrators prefer this method of managing their distributed environment. Another advantage is that **dcecp** is available on almost all implementations of DCE, which enables you to use the same commands on all platforms. Of course it doesn't make any difference whether you control the DCE system via the command line or by using graphical tools.

The following examples illustrate the use of the DCE Control Program via the command line (note the use of the continuation character, '➤', to indicate that one line of code maps to more than one line of print):

```
account create Account -mypwd Password -password Password
➤  -group Group -organization Org
```

This command creates a new user account. Using the **-mypwd** option, you have to type your own password in order to confirm your identity. You can then enter the new user's password. You must associate a group with the account, and you have to supply at least the organization name.

```
registry catalog [Registry_Replica_Name] [-master]
```

This returns a list of names of Security Servers running in the cell.

```
principal modify Principal_Name_List -add
➤  Extended_Registry_Attribute_List
```

This command lets you add extended registry attributes for your environment.

```
acl replace
```

This command enables you to replace the entire access control list of a given object. (Bear in mind that you also have to provide a whole bunch of options not specified in the above example.)

The DCE Control Program contains a portable command language called Tool Command Language (Tcl and pronounced ‘tickle’). Tcl is a platform-independent command language that runs on every system on which DCE is installed. The Tcl command interpreter is provided as part of the DCE software. It provides administration objects with names like *principal*, *endpoint*, and *clearinghouse*. Tcl’s design is a further approach toward an object-oriented administration interface.

You can use the task objects supplied with DCE or write your own objects and scripts using **dcecp** and other Tcl commands. The language supports variables, looping functions, conditional statements, and other programming constructs.

Scripts can easily be moved to other platforms without being rewritten. An enterprise with multiple cells can use **dcecp** scripts to propagate a common cell configuration throughout the entire company.

You’re provided with numerous C-like commands, including the looping commands *while*, *for*, and *foreach*, a *case* statement for testing values against patterns, and a *proc* construct for defining new custom commands.

You work from the *dcecp>* prompt; if, for example, you want to check the time of a DTS clock, you type the following command, with the indicated response:

```
dcecp> clock show
1998-10-24-15:31:08.099+00:00I-----
dcecp>
```

From the command shell you can use **dcecp** directly, in which case you just type **dcecp –c** followed by the command or commands you want to execute, which are separated by a semicolon. The commands can be included in a script or batch file.

The on-line reference contains complete descriptions of all the available commands and their arguments and options. One advantage of using the command line is that you can write scripts to automate repeated tasks and command sequences. The command reference tells you exactly which commands you can use within scripts and which of them can be used only interactively. Using InfoExplorer on AIX is a more convenient way of reading the command reference.

However, **dcecp** cannot access all DCE functionality – additional standard programs are needed. The administration tool for RPC mappings, profiles, and groups is **rpccp**. Use **cdscp** to administer the Directory Server, **rgy_edit** for Security Server objects, and **sec_admin** to administer the Security Server itself. With **acl_edit** you can set, list, modify, and delete ACLs, **dtscp** controls the Time Server, and you will need **bak**, **bos**, **fts**, and **cm** to administer DFS.

While the standard command line interface allows you to enter commands efficiently and to write scripts, **smit** enables you to perform many DCE administration tasks much more easily.

DCE/DFS WEB TOOLS FOR AIX

DCE for AIX comes with two Web-based graphical administration tools for DCE and DFS. To use them, you first have to install Netscape FastTrack or Enterprise Web server. Based on this Web server platform, you can install DFS Web Secure and DCE Web Administration. The second of the two needs Web Secure installed, in addition to Netscape's server.

DFS Web Secure allows users to access documents in DFS and provides DCE credentials to CGI programs accessed through a Web browser. This package is discussed in more detail later.

To use the DCE Administration utility, you need a Web browser that's frame-enabled. DCE Administration allows you to manage DCE and DFS objects using a Web browser. This allows you to administer your

entire DCE or DFS environment. The tasks of creating users, modifying group membership, working with permissions on DCE objects, and setting up filesets are all greatly simplified with this graphical administration utility.

You configure the Web utilities using **smit**: From the main **smit** panel, select *Communication Application and Services*, then *DCE*, *Configure/Unconfigure Web Administration*, and *Configure Web Administration*. At the Netscape *Directory* panel, type in the home directory of the Netscape server installation. At the *Configuration* panel, enter the ID of the Netscape server as well as your own user ID, then tick *'All'* in *Components to configure*.

IBM DCE MANAGER FOR AIX

The DCE Manager for AIX, which is available as an add-on product for DCE for AIX 2.1, enables an administrator to work from within a graphical interface. The program includes a graphical view of one or even more DCE cells and their resources. DCE Manager is fully integrated with IBM's NetView for AIX – it works as a logical extension to the management capabilities of NetView. Consequently, the program supports the same user interface used in NetView, with the same graphical representation of machines and services. These representations are called *maps* and *submaps*. IBM DCE Manager can also be launched from NetView for AIX.

All DCE core servers can be 'discovered' and monitored automatically. The core servers comprise the Security, Directory, Global Directory, Time, RPC, and DFS Servers in cells being managed. DCE Manager acts as a DCE client. It's able to monitor the DCE daemons on systems within the cell.

The DCE Manager for AIX is able to collect dynamic performance statistics from servers. This information helps you to keep the system running and plan future performance. Furthermore, DCE Manager can manage multiple cells, adding a new map and associated submaps to NetView's root submap. The software can automatically discover cells and create the necessary submaps.

Marginal and critical usage thresholds can be set for each aggregate

holding filesets on DFS servers. When these thresholds are exceeded, a change of colour used to represent the server on the submap immediately communicates the change of status.

Administration programs have tended to become more and more graphically-oriented from one version to the next, as is the case with Web administration. This means that you then have to decide whether to install the graphical programs, which are easy to use, or the command line program, which you can automate by writing powerful scripts.

DCE AND THE WEB

The Internet has become an essential element in modern information technology. It gives you fast access to relevant data and applications, which opens new ways for enterprises to support their businesses. Basic Web technologies are cheap and simple. However, if you need security (and who doesn't!), it's not so simple. Managing controlled access and protecting resources is an important but complex issue. To this end, the DCE environment sets a high level of security for distributed applications.

Each networking element requires a transparent security policy and architecture to control access to data and services. DCE's and DFS's authentication and privacy services work within an organization, but are also scalable for controlling access from outside the organization through linked cells. DCE Security Services can also be extended to the Web and, therefore, provide a high level of security for Internet and Web services too.

Netscape's FastTrack and Enterprise servers provide only minimal authentication and authorization services for document access. Users are prompted to type their name and password when requesting a protected page. If both are correct, access is authorized. Netscape's authorization is restricted to read and write privileges based on users or groups. The user database is maintained within the Web server, which constitutes an isolated security structure.

IBM offers DFS Web Secure, which is aimed at overcoming the aforementioned problems by linking Web document access to the

DFS and DCE security infrastructure. The entire authentication is executed by DCE and not the Web server. This function is implemented as a plug-in for the server.

Authorization functionality is thus extended through added privileges and authorization types available under DCE. Similarly, DCE differentiates between directory browsing and access to the Unix file system. DFS Web Secure implements a single, central security administration infrastructure and provides a scalable document storage environment using DFS.

DFS Web Secure acts as a gateway between DFS's file-level security and that of Netscape's Web servers. Using NSAPI, the existing Netscape authentication model is extended to embrace DCE/DFS access security when accessing documents stored within DFS. The program is responsible for the authentication, access control, and logging of document requests in its designated namespace. In order to authenticate the user, DCE security services are contacted and the file access is verified by DFS based on the user's credentials.

Security administration is carried out using the DFS Web Manager, which maintains databases for logging failed DCE authentications, document access counts, and page access counts and traffic volume by authenticated users. You can view and reset the databases either manually or automatically at pre-defined intervals.

The installation of IBM Web Secure comprises three main steps. First you set up the Netscape Web server, then DFS access, and lastly the DFS Web Secure program.

Netscape's Fasttrack Server and Enterprise Server can be used interchangeably. You should set *MinThreads* and *MaxThreads* in the Web server to 1 as this conforms more closely with DFS Web Secure's model. The DFS client has to be installed and configured on the same machine as the Web server and DFS Web Secure. It is not, however, necessary to have any DFS server functions installed.

The installation of Web Secure requires 5 MB of extra disk space in */usr*. If necessary, the **installp** program automatically increases the size of the filesystem. You need a frames-capable Web browser, such as Netscape Navigator 3.0 or later. Create a DCE principal *anonymous*

with the same password. You can use this account to allow users who are not defined in your environment to access areas within your DFS filespace, as DFS Web Secure always forces a user to login to DCE.

DFS Web Secure can be installed with the following command:

```
# installp -acv -d <image directory> dce.dfsweb
```

Another way to install this component is to use **smit**'s install facility.

After installation, it is recommended that you verify whether the program is successfully set up. Type the following command:

```
# lslpp -L | grep dce.dfsweb
```

This should yield the following message:

```
dce.dfsweb.rte       1.1.1.0   C   DFS Web Secure
```

The following command enables you to configure DFS Web Secure in a single step:

```
# mkdfsweb -n <Netscape_Server_Installation_Directory>
➤  -s <NS_Server_Name> -i <NS_AIX_User_ID> -a anonymous
➤  -p anonymous
```

The changes are written to the Netscape server configuration files *magnus.conf* and *obj.conf*. When using Netscape Administration Server interface via a browser at the same time, you have to reload the actualized configuration files.

Configuration facilities are provided with the DCE Administrator, which offers facilities for DCE user and group administration, organization administration, and ACL management for DFS file objects via a Web-based graphical utility. DCE Administrator offers you the functionality you need to use DCE and its security services along with your Web server.

IBM DFS Web Secure for AIX is bundled with the IBM DFS Starter Kit for AIX, and you can also download it from *www.networking.ibm.com/dws/dwsprod.html*.

More information on DFS Web Secure is available from the *DFS Web Secure Product Guide*. You can access this guide from a Web browser after installing and configuring DFS Web Secure. Open the guide by requesting *http://servername/dceweb*.

If you want to uninstall DFS Web Secure for AIX, don't forget to unconfigure it first so as to recover the original Web server configuration. Only then you can uninstall the component without endangering the integrity of the Web server's set-up.

PROGRAMMING SUPPORT

DCE systems can be configured for the development of DCE applications. This requires the basic DCE Client configuration to be undertaken, and also that of the interface specification files and the IDL compiler.

DCE for AIX supports the C language. DCE has initially been defined with C as a primary target language. That means that other languages cannot be used as easily as C. The compatibility between C and C++ makes it simple to develop DCE applications using the latter. You should, however, avoid using C++ in DCE-specific parts of the project. On the other hand, OSF DCE Version 1.2 has been extended to support C++ object features, such as inheritance and object references.

Various tools allow developers to write programs for DCE. For instance, every program, interface definition, and so on needs a unique identifier that is provided by the UUID generator. UUIDs are needed when defining new RPC interfaces or creating objects managed by application servers. DCE applications could be written in any of a number of programming languages, and the two parts of a client/server system need not necessarily be written in the same language. This flexibility is enabled by the existence of a common language – the Interface Definition Language or IDL, which is supported by the IDL compiler.

The software package includes a thread library, DCE libraries, and 'include' files. Apart from this, there are thread-aware versions of the AIX standard debuggers **dbx** and **xde** that allow the debugging of multithreaded software.

Another product from IBM, *Getting Started with DCE for Application Developers*, provides DCE code samples, tools, extensive help, and information.

If application development is your major objective, you should also have a look at third-party products. There are many of them around for the Unix platform as this was the first platform to provide a commercial DCE environment. These products include high-level libraries, generators, and procedural and object-oriented wrapper libraries. Such third-party products are aimed at hiding the complexity of DCE's low-level standard APIs.

HETEROGENEOUS NETWORKS

Since DCE is available for all major platforms, you can use it for integrating operating systems you run in your company. AIX can be included as client platform, server, or both. Namespaces and the X.500 hierarchy are used equally on all platforms. Computers are called *principals*, and these principals figure among all DCE objects. It doesn't matter whether your servers and clients use AIX, Windows NT, OS/2, or other platforms.

The user management features integrate with those of specific operating systems and can be propagated throughout an entire environment, giving each operating system what it needs. Of course, there is no need for double administration as the DCE definitions are sufficient and provide the underlying operating system with the information it requires. It makes sense to combine all platforms that are needed by an enterprise. The most common DCE implementation I've come across in real environments is to use Unix servers and Windows NT clients, along with other clients such as Windows 95 and OS/2.

Even more complex environments can be realized since DCE is also available on the Apple Macintosh, OpenVMS on VAX and Alpha AXP, MPE/iX on the HP/3000, Hitachi mainframes, Bull DPS Systems, Cray Systems, and Siemens BS2000/OSF mainframes. DCE services are also available for IBM's AS/400, but there seems very little demand for it at the moment.

Other Unix platforms supported include HP-UX, DEC Unix, Hitachi, SCO Unix, Data General, Gradient, Siemens, and Sun Solaris (this list is not exhaustive).

The Integrated Login for AIX demonstrates how DCE facilities can be

integrated with those of an underlying operating system, and this example just scratches the surface of what's possible. DCE's transparent functionality hides many particularities of the underlying operating system and can thus be applied to forge different interfaces together. Using homogeneous or even the same applications in a large networked environment is a must for today's business and enterprises. At the moment, DCE seems to me to be the only environment that fulfils such a demand. In addition, DCE also gives you greater flexibility as it allows you to migrate from one platform to another and still use DCE and its accompanying applications.

Klaus Ebner
Curriculum Manager
IBM Austria E+T (Austria)

Date arithmetic

If my oldest daughter Luzia was born on May 7, 1988 and my youngest daughter Florine was born on March 20, 1990, how many days separate their birthdays? How do you work out many days a loan collects interest? And how many days did your errant friend spend in jail? Even a supposedly simple question, such as: "what day comes before *x?*", where *x* is a date in future, can be less than obvious if *x* is March 1, 2000.

JULIAN DATE FORMAT

Date arithmetic is simpler in the so-called Julian date format. In this format a date is expressed as a four-digit year followed by a three-digit ordinal number of the day in the year. January 1, 1999, for example, is '1999001', and February 1, 1997 is '1997032'. To write March 1, 2000 in Julian format, though, you first need to know whether 2000 is a leap year. Note that this date format has nothing to do with the Julian calendar and should probably be called a year-and-ordinal-day format. However, the term 'Julian format' has stuck to this format.

CONVERTING YYYYMMDD TO YYYYDDD

This problem would be fairly straightforward were it not for leap years, which usually, but not always, occur every four years. Before we develop the script **daysInYear** we must first establish what constitutes a leap year.

The rules for a leap year are as follows (each one overrides the preceding one):

- A year is a leap year if it's divisible by four
- Years divisible by 100 are not leap years
- Years divisible by 400 are leap years.

Testing for these rules is simplified by turning them upside down and first testing whether the year's divisible by 400, then by 100, then by four, and branching as soon as we have a match. Here's the shell script to calculate the number of days in a year.

DAYSINYEAR

```
#!/bin/sh

# daysInYear
# returns the number of days in the year
# usage: daysInYear {YYYY from stdin}
# or     daysInYear YYYY

# if there's no command line argument, use stdin
if [ "$1" = "" ]
then
    read year
else
    year=$1
fi

# a year is a leap year if it's divisible by 4, but not
# if it's divisible by 100, unless divisible by 400

# if the year is divisible by 400, it's a leap year
if [ `expr $year % 400` = 0 ]
then
    echo 366
    exit 0
fi
```

```
# if the year is divisible by 100, it's not a leap year
if [ `expr $year % 100` = 0 ]
then
    echo 365
    exit 0
fi

# if the year is divisible by 4, it's a leap year
if [ `expr $year % 4` = 0 ]
then
    echo 366
    exit 0
fi

# otherwise the year is not a leap year
echo 365

exit 0
```

In our shell script we use the function *expr*, which performs integer arithmetic on strings and shell variables. The operator *%* returns the remainder of a division. We use it to test whether the year is divisible by 400, 100, or four. Only if it's divisible does the operation return the value '0'. Now we can determine the number of days in a year, we move to the next step, which is to determine the number of days in different months in a given year. The shell script **daysInMonth** allows its input to be piped in, entered as a single parameter in the form *YYYYMM*, or as two parameters *YYYY* and *MM*.

DAYSINMONTH

```
#!/bin/sh
# daysInMonth
# returns the number of days in a month in a specified year
# usage: daysInMonth {YYYYMM from stdin}
# or     daysInMonth YYYYMM
# or     daysInMonth YYYY MM

if [ "$1" = "" ]
then
    read yearMonth
elif [ "$2" = "" ]
then
    yearMonth=$1
else
    yearMonth=`expr \( \( $1 \* 100 \) + $2 \)`
fi
```

```
# extract the year and the month
year=`expr $yearMonth / 100`
month=`expr $yearMonth % 100`

# Months 1, 3, 5, 7, 8, 10, and 12 have 31 days
# Months 4, 6, 9, and 11 have 30 days
case $month in
    1 | 3 | 5 | 7 | 8 | 10 | 12 )
        echo 31
        exit 0
        ;;
    4 | 6 | 9 | 11 )
        echo 30
        exit 0
        ;;
    2 )
        ;;
    * )
        echo "$month: illegal month"
        exit 1
        ;;
esac

# month 2 is a special case as it depends on the year -
# use the script daysInYear to determine whether
# year is a leap year
days=`daysInYear $year`
case $days in
    365 )
        echo 28
        ;;
    366 )
        echo 29
        ;;
esac

exit 0
```

Note that the sixteenth line contains the command:

```
expr \( \( $1 \* 100 \) + $2 \)
```

The characters ‘(’, ‘*’, and ‘)’ are preceded by a backslash, ‘\’, to avoid having them misinterpreted by the shell. Otherwise we would just have written:

```
expr ( ($1 * 100) + $2)
```

Now that we can handle leap years properly we still need to establish how to convert a date from the Gregorian *YYYMMDD* format to the

erroneously-named Julian *YYYYDDD* format. Let's first look at the **ymd2yd** script before we look at its explanation.

YMD2YD

```
#!/bin/sh

# ymd2yd
# converts a date from YYYYMMDD format to YYYYDDD format
# usage: ymd2yd {YYYYMMDD from stdin}
# or     ymd2yd YYYYMMDD
# or     ymd2yd YYYY MM DD

if [ "$1" = "" ]
then
    read yearMonthDay
elif [ "$2" = "" ]
then
    yearMonthDay=$1
else
    yearMonthDay=`expr \( \( $1 \* 10000 \) + \( $2 \* 100 \) + $3 \)`
fi

# extract the year and month
year=`expr $yearMonthDay / 10000`
month=`expr \( $yearMonthDay % 10000 \) / 100`
day=`expr $yearMonthDay % 100`

# add the days in all preceding months to the day itself
ddd=0
mm=1
while [ `expr $mm \< $month` = 1 ]
do
    d=`daysInMonth $year $mm`
    ddd=`expr $ddd + $d`
    mm=`expr $mm + 1`
done
ddd=`expr $ddd + $day`

# combine the year and number of days to form the Julian date
echo `expr \( $year \* 1000 \) + $ddd`

exit 0
```

The main logic in this script comprises iterating backwards through the preceding months using **daysInMonth** to establish how many days they have. We then add the day itself to come up with the ordinal day of the year. For example, using November 23, 1967, we would add

the days in the months from one to 10 (though not 11) and then add 23. Because the number of days in February depends on the year, we have to know the year. In this case, it's 1967, which is not a leap year. If a date in the format *YYYYMMDD* is converted to *YYYYDDD* for date arithmetic, we also need the ability to convert back. The script **yd2ymd** converts a Julian date back to a Gregorian date.

YD2YMD

```
#!/bin/sh

# yd2ymd
# converts a date from YYYYDDD format to YYYYMMDD format
# usage: yd2ymd {YYYYDDD from stdin}
# or     yd2ymd YYYYDDD

if [ "$1" = "" ]
then
    read yearDays
else
    yearDays=$1
fi

# extract the year and the days
year=`expr $yearDays / 1000`
ddd=`expr $yearDays % 1000`

# subtract the days of each month beginning with month 1 from the days
# in the date until the value goes below 1. Then we have the month and
# need to add back the remaining days
mm=1
while [ `expr $ddd \> 0` = 1 ]
do
    d=`daysInMonth $year $mm`
    ddd=`expr $ddd - $d`
    mm=`expr $mm + 1`
done

# we went one month too far
mm=`expr $mm - 1`
d=`daysInMonth $year $mm`
ddd=`expr $ddd + $d`

# combine the year, month, and days to form the Gregorian date
echo `expr \( $year \* 10000 \) + \( $mm \* 100 \) + $ddd`

exit 0
```

DATE ARITHMETIC

Now we're finally equipped to perform date arithmetic. The script **ydAdd** allows a positive or negative number of days to be added to a date in *YYYYDDD* format. (Adding a negative number of days is the same effect as subtracting the number of days.) Here's the script:

YDADD

```
#!/bin/sh

# ydAdd
# adds days to a YYYYDDD format date
# usage: ydAdd {YYYYDDD from stdin} days
# or     ydAdd YYYYDDD days

if [ "$2" = "" ]
then
    read yearsDay
    addDays=$1
else
    yearsDay=$1
    addDays=$2
fi

# extract the year and the days
year=`expr $yearDays / 1000`
day=`expr $yearDays % 1000`

# add the number of days
day=`expr $day + $addDays`

# while the calculated day exceeds the days in the year, subtract
# the number of days in the year and add one year to the year
d=`daysInYear $year`
while [ `expr $day \> $d` = 1 ]
do
    d=`daysInYear $year`
    day=`expr $day - $d`
    year=`expr $year + 1`
done

# the reverse process:
# while the calculated day is less than 1, subtract one from the year
# and add the number of days in that year
while [ `expr $day \< 1` = 1 ]
do
    year=`expr $year - 1`
```

```
    d=`daysInYear $year`
    day=`expr $day + $d`
done

# combine the year and days to form the Julian date
echo `expr \( $year \* 1000 \) + $day`

exit 0
```

This time the main logic in our script is to repeatedly subtract the number of days in a year and go to the next year till we've used up all our days. For the reverse process we go back a year and add its number of days until we have a positive number of days.

Let's look at the example of adding 1000 days to January 1, 1998.

Action	*Result*
Starting date	January 1, 1998
Convert to Julian using **ymd2yd**	1998001
Separate year	1998
Separate days	1
Add 1000 to 1	1001
Days in 1998	365
Is 1000 > 365?	True
Then compute 1000 – 365	635
Year + 1	1999
Days in 1999	365
Is 635 > 365?	True
Then compute 635 – 365	270
Year + 1	2000
Days in 2000	366
Is 270 > 366?	False
Is 270 < 1?	False

Combine year and days	2000270
Convert to Gregorian using **yd2ymd**	20000926

Now let’s take a look at the reverse process by subtracting 1000 days from January 1, 1998.

Action	*Result*
Starting date	January 1, 1998
Convert to Julian using **ymd2yd**	1998001
Separate year	1998
Separate days	1
Add –1000 to 1	–999
Days in 1998	365
Year – 1	1997
Days in 1997	365
Compute –999 + 365	-634
Is –364 < 1?	True
Year – 1	1996
Days in 1996	366
Compute –634 + 366?	-268
Is –268 < 1?	True
Years – 1	1995
Days in 1995	365
–268 + 365	97
Is 97 < 1?	False
Combine year and days	1995097
Convert to Gregorian using **yd2ymd**	19950407

Now we can add or subtract days from a date that is in Julian format. Our script **addDate** is similar to **ydAdd** but expects the date in the more common Gregorian format.

ADDDATE

```
#!/bin/sh

# addDate
# adds days to a YYYYMMDD formatted date
# usage: addDate {YYYYDMMDD from stdin} days
# or     addDate YYYYMMDD days

if [ "$2" = "" ]
then
    read yearMonthDay
    addDays=$1
else
    yearMonthDay=$1
    addDays=$2
fi

# echo the Gregorian date through the pipe to convert it to Julian
# format, then add the number of days before converting it back to
# 8-digit Gregorian format
echo $yearMonthDay | ymd2yd | ydAdd $addDays | yd2ymd

exit 0
```

MORE DATE ARITHMETIC

But we still cannot answer our original question. If my oldest daughter Luzia was born on May 7, 1988 and my youngest daughter Florine on March 20, 1990, how many days separate their birthdays? For that we need to operate on two Gregorian dates. Again we first solve the problem using Julian dates.

YDDIFF

```
#!/bin/sh

# ydDiff
# calculates the difference in days between two dates
# usage: ydDiff YYYYDDD YYYDDD
```

```
yearDays1=$1
yearDays2=$2

# extract the year and the days
year1=`expr $yearDays1 / 1000`
day1=`expr $yearDays1 % 1000`
year2=`expr $yearDays2 / 1000`
day2=`expr $yearDays2 % 1000`

# do the two dates contain the same year?
if [ $year1 = $year2 ]
then
    # is the second date older than the first date?
    if [ `expr $day2 \> $day1` = 1 ]
    then
        echo `expr $day2 - $day1`
    else
        echo `expr $day1 - $day2`
    fi
    exit 0
fi

days=0
# is the second date older than the first date?
if [ `expr $year2 \> $year1` = 1 ]
then
    # look at the next year
    year=`expr $year1 + 1`
    # while the next year is still not the second year
    while [ `expr $year \< $year2` = 1 ]
    do
        # add the number of days in the next year
        d=`daysInYear $year`
        days=`expr $days + $d`
        # make the new next year the year after the old next year
        year=`expr $year + 1`
    done

    # add the days to the end of the first year
    d=`daysInYear $year1`
    days=`expr $days + \( $d - $day1 \)`
    # add all the days in the second year
    days=`expr $days + $day2`
else
    # look at the next year
    year=`expr $year2 + 1`
    # while the next year is still not the second year
    while [ `expr $year \< $year1` = 1 ]
    do
```

```
        # add the number of days in the next year
        d=`daysInYear $year`
        days=`expr $days + $d`
        # make the new next year the year after the old next year
        year=`expr $year + 1`
    done

    # add the days to the end of the first year
    d=`daysInYear $year2`
    days=`expr $days + \( $d - $day2 \)`
    # add all the days in the second year
    days=`expr $days + $day1`
fi

echo $days

exit 0
```

We extract the years between the two dates, then add the number of days to the end of the first year and the number of days in the last year. But first we have to determine which date of the two comes first.

Our last script calls the script **ydDiff**, but also allows Gregorian dates as its input.

DIFFDATE

```
#!/bin/sh

# diffDate
# calculates the difference in days between two dates in YYYYMMDD
# format
# usage: diffDATE YYYYMMDD YYYYMMDD

yearMonthDay1=$1
yearMonthDay2=$2

# convert the Gregorian dates to convert them to Julian format
yyyyddd1=`ymd2yd $yearMonthDay1`
yyyyddd2=`ymd2yd $yearMonthDay2`

ydDiff $yyyyddd1 $yyyyddd2

exit 0
```

That wasn’t so difficult, was it?

DATE FORMATS

Though we've now finished the actual date arithmetic, we should also have scripts to format our dates. The Gregorian date format *YYYYMMDD* may be nice and practical but it's certainly not the only format used in today's world. Our script needs to use formats such as those used by the **date** command.

Format string	Represents	Applied to '19990209'
%yyyy	Four-digit year	1999
%yy	Two-digit year	99
%mm	Two-digit month with leading zeros	02
%m	Two-digit month with no leading zeros	2
%dd	Two-digit day with leading zeros	06
%d	Two-digit day with no leading zeros	6
%mon	Three-character month abbreviation	Feb
%month	Full month name	February

The script **dateFormat** accepts an eight-digit date and a format string. Using the above table of format strings, output examples of the script are shown below.

```
formatDate 19980206 %dd.%%mm.%yy
06.02.99

formatDate 19980206 "%month %d, %yyyy"
February 6, 1999

formatDate 19980206 %yyyy-%mm-%dd
1998-02-06
```

Notice that our second example's date format contains spaces. To avoid problems we have to convert these spaces to the underscore character, '_', using **tr**.

FORMATDATE

```
#!/bin/sh

# formatDate
# convert a date from YYYYMMDD format to the required format
```

```
# usage: dateFormat format {YYYYMMDD from stdin}
# or     dateFormat YYYYMMDD format

if [ "$2" = "" ]
then
    read date
    format=$1
else
    date=$1
    format=$2
fi

# replace spaces in $format with '_'
format=`echo $format | tr " " _`

# divide the eight-digit YYYYMMDD date into its components to yield a
# four-digit year, two-digit year, two-digit month, and two-digit day
yyyy=`echo $date | cut -c1-4`
yy=`echo $date | cut -c3-4`
mm=`echo $date | cut -c5-6`
dd=`echo $date | cut -c7-8`

# create numeric versions of month and date to suppress leading zeros
m=`expr $mm \* 1`
d=`expr $dd \* 1`

# echo the month using a sed script to assign a month abbreviation
# to the variable $mon
mon=`echo $mm | sed -e "
s/01/Jan/
s/02/Feb/
s/03/Mar/
s/04/Apr/
s/05/May/
s/06/Jun/
s/07/Jul/
s/08/Aug/
s/09/Sep/
s/10/Oct/
s/11/Nov/
s/12/Dec/"`

# echo the month using a sed script to assign a full month
# to the variable $month
month=`echo $mm | sed -e "
s/01/January/
s/02/February/
s/03/March/
s/04/April/
s/05/May/
```

```
s/06/June/
s/07/July/
s/08/August/
s/09/September/
s/10/October/
s/11/November/
s/12/December/"`

# echo the format string using a sed script to search for the
# format strings and replace them with the appropriate values
# this step displays the date in the desired format
echo $format | sed -e "
s/%yyyy/$yyyy/g
s/%month/$month/g
s/%mon/$mon/g
s/%yy/$yy/g
s/%mm/$mm/g
s/%dd/$dd/g
s/%m/$m/g
s/%d/$d/g
s/_/ /g"

exit 0
```

Another question asked at the start of this article was: “which day comes before March 1, 2000?” Now we can answer that question too:

```
addDate 20000301 -1 | formatDate "%month %d, %yyyy"
```

The answer, February 29, 2000, means that the year 2000 is a leap year. That means you will have 366 days to celebrate the year. The next big question is whether the year 2000 or the year 2001 is the beginning of the next millennium…

Werner Klauser
Klauser Informatik (Switzerland)

Viewing disk usage

When creating new logical volumes (LV) you need to decide which volume group (VG) the LV will be part of, which physical volume (PV) or volumes it will reside on, and where it will actually be placed

on the disk. To help you in this you can use the command **lsvg –p vg_name**, which shows you the PVs that make up the VG, the amount of space on that PV, the free space, and the distribution of free space. However, this information is presented in terms of physical partitions, the size of which is not even on the output of the above command. The script below resolves these issues by presenting this information in megabytes and formatting the output. Note the use of the continuation character, '➤', to indicate that one line of code maps to several lines of print.

THE SCRIPT

```
#!/bin/ksh
#
# This script shows details of space available on physical volumes
# (PV) in volume groups (VG). This information is presented in
# megabytes, as opposed to Physical Partitions. This makes it a
# useful replacement for 'lsvg -p vg_name'. For each PV it shows
# the total space on the disk, the free space, and the distribution
# of that free space.
#
# One or more VGs can be specified on the command line. If none
# are specified, the script prints details for all VGs. If the
# details for more than one VG are printed the script also prints
# a summary of the total available and free space in those VGs.

integer tavail=0
integer tfree=0
integer vgavail
integer vgfree
integer ppsize
integer tpp
integer fpp
integer position
integer count=0

if [[ $# -eq 0 ]]
then
  VG_LIST=$(lsvg)
Else
  VG_LIST=$*
Fi

for vg in $VG_LIST
do
  vgavail=0
```

```
  vgfree=0
  ppsize=$(lsvg $vg | grep "^VG STATE" | awk '{print $6}')
  echo "Data for volume group '$vg' (PP Size = ${ppsize}Mb):"
  echo "             Total    Free    Outer  Middle  Centre
  ➤  Middle   Inner"
  lsvg -p $vg | tail +3 | while read hd state tpp fpp dist
  do
    tpp=tpp*$ppsize
    fpp=fpp*$ppsize
    vgavail=vgavail+$tpp
    vgfree=vgfree+$fpp
    printf "  %-7s: %5dMb %5dMb " $hd $tpp $fpp
    dist=$(echo $dist | tr '.' ' ')
    for position in $dist
    do
      printf " %5dMb" $(($position*$ppsize))
    done
    echo
  done
  printf "\n  VG available = %dMb  VG free = %dMb\n\n" $vgavail
  ➤  $vgfree
  tavail=tavail+vgavail
  tfree=tfree+vgfree
  count=count+1
done

if [[ $count -gt 1 ]]
then
  printf "Total available = %dMb  Total free = %dMb\n" $tavail
  ➤  $tfree
fi
```

Robin Venables
AIX Systems Administrator (UK)

SCSI address resolution

You've taken delivery of a new SCSI-attachable device, along with all the necessary cables. In the back of your mind you can remember being told something about SCSI addresses and possible problems. The problem is, you can no longer remember the details. What is the proper way to connect a SCSI adapter to an RS/6000?

THE PROPER WAY TO CONNECT A SCSI ADAPTER

Each device must be configured and put in the 'available' state before it can be used. To achieve this, you've first got to examine the list of SCSI controllers in your system. The following command identifies all SCSI devices by device name, location, and description:

```
lscfg -l scsi*
```

The slot where the SCSI adapter device is located is an important configuration variable – the fourth number in the output identifies the adapter slot number. If the fourth number is '0' and the sixth is an 'S', then the device is configured on the planar and doesn't have a slot number.

Next, list all devices that are connected to the SCSI I/O controllers:

```
lsdev -C -s *scsi* -H
```

In order for a SCSI device to be set in the available state, an address that has already been selected cannot be used. Possible numbers range from '0' to '6'. The configuration manager first configures any and all internal devices to the system. Therefore, if you are going to attach an external device to a SCSI I/O controller adapter, use a higher address such as ''6', '5', and 4' (in that order of preference) to avoid duplication.

PHYSICALLY CONNECTING THE SCSI DEVICE

Before you connect a new device, it is strongly recommended that you back up your data to allow yourself to recover in case something goes wrong. Before connecting the device, shut the system down completely. Then connect all your new devices. First turn all attached devices on, leaving the RS/6000 system off. Then turn the RS/6000 system on. At that time, the AIX configuration manager automatically updates the custom configuration database with information about externally attached devices.

If you select an address that has already been used, the external device will not be set in the available state. If the external device is already listed in the custom configuration database, don't just unplug it as it'll still be defined to the system but 'not available for use'.

If you simply turn the address on the back of the device to another number and then try to configure it, it will appear as though you have two devices in the system. To remedy the situation, remove the device totally from the configuration database using **smit**, then add the device to the system with the correct address number.

If you show a little caution when connecting new SCSI devices, and don't just connect them to powered-up systems, things should go smoothly and you shouldn't need you to restore that back-up you made!

System Programmer (Switzerland)

Checking CPU status

The **cpu_state** command enables the (root) user to list, disable, and enable CPUs. However, this command works only on systems based on the Micro Channel Architecture (MCA) bus that have a service processor. It is not possible to disable or enable CPUs on machines based on the Peripheral Connect Interface (PCI) bus, though it is possible to list the CPUs' current state. The script below serves two purposes: it checks whether the machine is PCI-based, in which case it defaults to the reduced functionality of simply offering the user a display of CPU status, otherwise the script calls **cpu_state** and returns the user the output of this function.

THE SCRIPT

```
#!/bin/ksh
#
# This script first checks whether the system is PCI bus-based;
# if it is, details of all CPUs are displayed. If it is not, the
# script calls /usr/sbin/cpu_state, passing all arguments to the
# command that were passed to it. The scripts exits with the number
# of CPUs on a PCI machine and the return code from the call to
# cpu_state on an MCA machine.

if [[ $(lscfg | grep -ci "pci bus") -ge 1 ]]
```



```
then
  integer count=0
  echo "\tName    Cpu  Status     Location"
  lsdev -C -c processor | while read name trash1 location trash2
  do
    case $(lsattr -E -l $name -a state -F value) in
      "enable")
        state="Enabled"
        ;;
      "disable")
        state="Disabled"
        ;;
      "faulty")
        state="Faulty"
        ;;
      *)
        state="Unknown"
        ;;
    esac
    printf "\t%-7s %d    %-8s   %s\n" $name $count $state $location
    count=count+1
  done
  exit $count
else
  /usr/sbin/cpu_state $*
  exit $?
fi
```

Robin Venables
AIX Systems Administrator (UK)

In partnership with IBM, Candle has announced high-availability software for RS/6000 clusters running AIX 4.3.2. The Candle Command Centre (CCC) for High-Availability Systems provides software to help protect availability and reliability by identifying and eliminating potential downtime threats.

The software is aimed at clusters running ERP-based applications and is part of IBM's Virtual Bundle, which includes RS/6000 hardware and various software components. Versions of the Candle software also support multiple databases and applications including SAP R/3, DB2 Universal Database, and PeopleSoft.

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

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

* * *

HAHT Software has announced HAHTsite Version 4, a Web application development platform for AIX, other versions of Unix, and NT. It supports all major TP monitors and provides enterprise-level security from the page object level to triple-DES encryption through the firewall. New features include support for 100% Java projects or a mix of HAHTtalk Basic and Java, inclusion of Visibroker, security enhancements, an HTML editor, and application partitioning.

The integrated development environment costs £1250 per seat in the UK, while the Integrated Publisher costs £450 a seat. An Application Server Starter Pack for AIX costs £4,700.

For further information contact:
HAHT Software, 4200 Six Forks Road, Raleigh, NC 27609, USA
Tel: +1 919 786 5100
Fax: +1 919 786 5250
Web: http://www.haht.com

HAHT Software UK, The Mews, Kings Ride Court, Kings Ride, Ascot, Berks Sl5 7JR, UK
Tel: +44 1344 624949
Fax: +44 1344 872229

* * *

Tivoli Systems has announced Global Sign-On for Tivoli Enterprise, which provides a single point of access to all resources in an enterprise, including systems and servers, databases, and the Internet. Logon information and passwords are kept in a secure central database using DES encryption. In addition to AIX, the product also supports a range of other systems, such as S/390, AS/400, and Windows NT.

Out now, prices start at US$2,000 per server and US$75 per client.