49 AIX update

November 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.

Who gets killed when you send a signal?

Signals are sent to running processes by the operating system or other programs to indicate that an event, exterior to the process, has occurred to which the process should respond.

The simplest example is the 'hang up' signal, or SIGHUP. When a user logs on from a remote terminal, a variety of events can happen that cause the terminal to go dead as far as the central processor is concerned. The line can hang up for a number of reasons, such as phone outages, modem problems, and loss of power at the remote terminal. Added to this, the user could simply switch the terminal off without first logging out. All of these conditions cause programs to go 'out of control' – in other words, a program or process that was being run or controlled from a terminal is no longer under the control of that terminal. The program itself needs to know what to do in this case. The Unix operating system keeps track of which processes are under the control of which terminals; when a terminal hangs up, drops a connection, or just switches off, the operating system sends a SIGHUP signal to all processes that were launched by that terminal.

A process that receives a SIGHUP signal has three options:

1 The process carries out the default action, which is usually to stop executing immediately.

2 The process can be programmed to catch the signal (this is known as 'trapping the signal') and ignore it. The process just continues running.

3 The process can be programmed to catch (trap) the signal and do something sensible, such as update and close all open files and exit. This is a very sensible option for a program that is handling complex files, such as database files, that might be corrupted if they are just closed as a consequence of the process terminating.

Which of the three options is the right one depends on the type of program being run.

The **kill** command sends a signal to a specific job. You can use the

command **ps -ef** to locate a job number and then use **kill** to send a signal to that job. Before you start killing processes, you need to understand a little more about what signals do and how programs handle them.

In order to see some practical results let's try a few simple examples. Switch to the Korn shell, if you are not running it already, using the **ksh** command. The examples that follow use a short script and the **kill** command. The syntax for **kill** is:

```
kill -(SignalNumber) ProcessNumber
```

The hang up signal is '1', so the command:

```
kill -1 1234
```

sends the hang up signal to process '1234', which is exactly what would happen if the process's controlling terminal had hung up and the operating system had intervened to signal the process.

waiter is a short script that sleeps for five seconds, wakes up, prints a message that it's awake, and then goes back to sleep. Type this script in and save it as *waiter*, and then change its mode so that it can be executed using the command **chmod a+x waiter**.

WAITER

```
while true
do
    sleep 5
    echo "Awake now"
done
```

Run *waiter*, detaching it from the terminal. To do this type *waiter* followed by an ampersand ('&') on the command line. After you press enter, something like the following appears:

```
waiter &
[1]  4567
```

The second number will vary and is the process id or job number of the *waiter* process. Make a note of this number. This job is now running as a background process, but because it's echoing the words 'Awake now' every five seconds, 'Awake now' appears on your terminal at five-second intervals.

You can stop the job by issuing the command below, though make sure you use the actual process id that appeared when you started the job.

```
kill -1 4567
```

Don't worry if the words 'Awake now' butt into the middle of your typing – it won't affect the command you are entering. Finish typing the command and press <ENTER>. The **kill** command sends the hang up signal to *waiter* and *waiter* simply carries out the default action: it drops dead, and the 'Awake now' messages stop appearing at your terminal.

If 'Awake now' continues to appear on your terminal, check that you used the right job number and try the **kill** command again. If that doesn't work, repeat the **kill** command but change the **-1** to **-9**. (There will be more about the **-9** signal in just a moment.)

In the second listing (below), *waiter* has been modified to trap the hang up signal. A function (*echo01*) is implemented that prints a message to let the user know that a hang up signal was received. In the main body of the script (after the function), the *trap* command is used to set up a trap for the hang up signal. The *trap* command, *trap echo01 1*, states that, whenever interrupt signal '1' is received, the program interrupts whatever it's doing and executes function *echo01*. Notice that the program doesn't quit – after executing function *echo01*, it picks up where it left off and continues.

WAITER WITH A TRAP

```
function echo01 {
    echo "Received signal 1 (SIGHUP)"
}

trap echo01 1

while true
do
    sleep 5
    echo "Awake now"
done
```

Now if you start *waiter* with an ampersand and then issue a **kill -1**, your screen will look something like this (see overleaf):

```
$ waiter &
[1]   951
$ Awake now
Awake now
kill -1 951
$ Received signal 1 (SIGHUP)
Awake now
Awake now
```

The *trap* in the program now catches the signal '1', displays a message, and continues running. You can still stop the program by sending a different signal, such as **kill -2** or **kill -9**.

This technique tends to be used in large applications. The reason for this is that the program is frequently in the middle of important or complex actions that should not be dropped. A good example is closing open database files that need to have indexes updated and other housekeeping done to perform an orderly shutdown.

The third listing below is a closer approximation of how a large application might handle a hang up signal.

WAITER WITH A BIGGER TRAP

```
function echo01 {
    echo "Received signal 1 (SIGHUP)"
    echo "Now I would close files if I had any open."
    exit
}

trap echo01 1

while true
do
    sleep 5
    echo "Awake now"
done
```

Start this version of *waiter* with an ampersand and try to kill it with **kill -1**. Your screen should look something like the example below, and the program will stop executing.

```
$ waiter &
[1]   951
$ Awake now
Awake now
kill -1 951
```

```
$ Received signal 1 (SIGHUP)
Now I would close files if I had any open.
```

So what does **kill -9** do? Signal '9' is SIGKILL, and it cannot be trapped. If you send a signal '9' to a process, you are telling the operating system to cut it off at the knees – drop dead now! The advantage of signal '9' is that the program cannot trap it and ignore it. The disadvantage of signal 9 is that the program cannot intercept it and perform an orderly shut down, even if it needs to.

The fourth listing is *waiter*, modified with a trap for signal '9'.

WAITER TRYING TO TRAP SIGNAL 9

```
function echo09 {
    echo "Received signal 9 (SIGKILL)"
    echo "Now I would close files if I had any open."
    exit
}

trap echo09 9

while true
do
    sleep 5
    echo "Awake"
done
```

Start this version of *waiter* with an ampersand and try to kill it with **kill -9**. Your screen will look something like the example below, and the program will stop executing.

```
$ waiter &
[1]   1151
$ Awake now
Awake now
kill -9 1151
$
```

There is no friendly message on the screen about trying to close files. The process just dies where it stands on receipt of a signal '9', even though a trap was prepared for it. For this reason, using **kill -9** on a process that controls a database application or a program that updates files can be disastrous.

Most well behaved processes are written to allow an orderly shutdown

when a signal other than ‘9’ is received. Signal ‘1’, SIGHUP, is possibly the most common signal used for an orderly shutdown. Most applications intercept such signals and shut down correctly. So, if you need to kill an application, try sending it a ‘1’ or another signal below ‘9’ before you resort to a ‘9’.

Here are some of the other common signals along with the events that typically cause them to be generated.

1 SIGHUP (‘hang up’) is caused by the communication line to the terminal being dropped.

2 SIGINT (‘interrupt’) is generated from the user keyboard, usually by a <Control>+C, <BackSpace>, or <Del>. To find out which, issue the command **stty –a** and locate the entry for ‘intr’ (such as ‘intr = DEL’, ‘intr = ^C’, or ‘intr = ^H’) in the output.

3 SIGQUIT (‘quit’) is also generated from the user keyboard, usually by <Ctrl>+\ or <Ctrl>+Y. To find out which, issue the command **stty -a** and locate the entry in the output for ‘quit’ (for example, ‘quit = ^\’ or ‘quit = ^Y’). A SIGQUIT often causes a core file to be created. This is a copy of your current memory.

15 SIGTERM (‘software terminate’) is usually generated by another program. The **kill** command uses ‘15’ as the default interrupt signal. If you specify

```
kill job
```

without a signal number, **kill** sends signal ‘15’ to the job.

Using **kill** without a signal number is good practice. It is the least harsh signal and, if it does its job and the programmers did their job when they built the application, the process should execute an orderly shutdown.

It requires additional time and code to write a trap for a signal in a program. When a program includes a trap, this is a good indication that the programmer felt there was good reason to include one. If the program can simply die without doing any clean-up, then why go to the trouble of including a trap? That’s why it’s a good idea to try signals ‘15’, ‘1’, and ‘2’ before resorting to signal ‘9’.

Several signals are generated by error events in the operating system. SIGILL (signal ‘4’) is caused by an illegal instruction. SIGFPE (signal ‘8’) is caused by a floating-point error. SIGSEGV, (signal ‘11’) is caused either by a memory segmentation error or just a plain memory error.

Some signals don’t cause processes to die. SIGUSR1 (signal ‘16’) can be used to signal something to another process. To find out what signals are available on your system, locate the file named *signal.h*. This is usually in a directory path that ends with */sys/*, such as */usr/include/sys/signal.h*. You can do this by using the **find** command:

```
find / -name signal.h -print
```

When you find the file, open it in read-only mode so that you don’t make any changes to it. It will give a list of all available signals. You should also take a look at **man signal** for further information.

Mo Budlong (USA)

Message of the day

This month’s instalment concludes this article on generating an automatic ‘message of the day’.

RMOTD.SH (REPLACE MESSAGE-OF-THE-DAY)

```
#! /bin/ksh
##############################################################################
#                                                                            #
# rmotd.sh - replace the 'message-of-the-day'.                               #
#                                                                            #
# This script replaces the current message-of-the-day with the               #
# following day's message. If a message for the following day is             #
# not found in the message file, the default message is used.                #
#                                                                            #
# Notes   1   The script should run as a cron job with all output            #
#             redirected to /dev/null.                                       #
```

```
#                                                                      #
#         2   The script can also be run from the command line.        #
#                                                                      #
#         3   The script contains following functions:                 #
#                                                                      #
#             - InitializeVariables                                    #
#             - InitializeRMOTDLogFile                                 #
#             - HandleInterrupt                                        #
#             - MoveCursor                                             #
#             - DisplayMessage                                         #
#             - PrepareMessageFile                                     #
#             - ReplaceMessageFile                                     #
#             - ProcessExit                                            #
#             - main                                                   #
#                                                                      #
#         4   If the RMOTD log file is not already initialized        #
#             the script initializes this file.                        #
#                                                                      #
# History                                                              #
# Date      Author    Description                                      #
# ---------------------------------------------------------------------- #
# 09/02/99  A Zaman   Initial Build                                    #
#                                                                      #
########################################################################

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

# define locations
MESSAGE_DIR="/home/ecatmgr/motd"
MESSAGE_FILE="${MESSAGE_DIR}/motd.dat"
TEMP_FILE="/tmp/cmotd_$$.tmp"
MOTD_FILE=/etc/motd         # original message-of-the-day file
RMOTD_LOG="${MESSAGE_DIR}/rmotd.log"
DATETIME=`date "+%d%m%Y at %H:%M:%S"`

# define date-related variables
DAY=           # day   of the message
MON=           # month of the message
YEAR=          # year  of the message

# define message related variables
MSG=           # message-of-the-day
```

```
# 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 Menu  title
MMOTD="${RVON}Manage Message-of-the-day${RVOFF}"

# define exit codes
SEC=0
FEC=1

TRUE=0
FALSE=1

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

# messages
INTERRUPT="Program interrupted! Quitting...${RVOFF}"
ROOT_USER="Script must be executed from root account${RVOFF}"
MOTD_ADDED="Message-of-the-day added${RVOFF}"
LOG_INITIALISED="Successfully initialized log file
\${RMOTD_LOG}${RVOFF}"
LOG_NOT_INITIALISED="Failed to initialize log file
\${RMOTD_LOG}${RVOFF}"
DMOTD_ADDED="Default message-of-the-day added${RVOFF}"
NO_MOTD="No message-of-the-day found${RVOFF}"
OSERROR="\${OSEM}${RVOFF}"

# define signals
SIGNEXIT=0 ; export SIGNEXIT   # normal exit
SIGHUP=1   ; export SIGHUP     # when session disconnected
SIGINT=2   ; export SIGINT     # ctrl-c
SIGTERM=15 ; export SIGTERM    # kill command
}

##############################################################################
#                                                                            #
# InitializeRMOTDLogFile                                                     #
#                                                                            #
# This function initializes the RMOTD log file.                              #
#                                                                            #
##############################################################################
InitializeRMOTDLogFile ()
```

```
{

# check the file exists
if [ ! -f ${RMOTD_LOG} ]
then
    # initialize the log file
    ( > ${RMOTD_LOG} > /dev/null 2>&1 )
    if [ $? -eq 0 ]
    then
        DisplayMessage I "${LOG_INITIALISED}"
        echo "        Log File for Message-of-the-day Replacement">
        ➤  ${RMOTD_LOG}
        echo "        ==========================================">>
        ➤  ${RMOTD_LOG}
        return $TRUE
    else
        echo "${DATETIME}:rmotd.sh:ERROR:Failed to initialize log
        ➤  file " > /dev/console
        return $FALSE
    fi
fi
}

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

##############################################################################
#                                                                            #
# MoveCursor                                                                 #
#                                                                            #
# This function moves the cursor to the required (Y, X) location.           #
#                                                                            #
# Input   :   X and Y coordinates                                            #
#                                                                            #
# Notes   1   The function must be run in ksh for print to work.             #
#             Also, the print command must be used to move the               #
#             cursor, as echo doesn't seem to work.                          #
#                                                                            #
##############################################################################
MoveCursor ( )
```

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

YCOR=$1
XCOR=$2

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

###############################################################################
#                                                                             #
# DisplayMessage                                                              #
#                                                                             #
# This function displays a message.                                          #
#                                                                             #
# Input   :   1 Message type (E = Error, I = Information)                     #
#             2 Error code, as 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}
}

###############################################################################
#                                                                             #
# PrepareMessageFile                                                          #
#                                                                             #
# This function prepares the new message-of-the-day file.                     #
#                                                                             #
# Returns :   $TRUE or $FALSE                                                 #
#                                                                             #
###############################################################################
PrepareMessageFile ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
```

```
# day of the message
DOTM=`date +%d%m%Y`

# check the message file
if [ ! -f ${MESSAGE_FILE} ]
then
    DisplayMessage E "${NO_MSG_FILE}"
    echo "${DATETIME}:rmotd.sh:ERROR:No message file found" >>
    ➤  ${RMOTD_LOG}
    return $FALSE
fi

# read the message-of-the-day from the file
MOTD=
MOTD="`grep  "${DOTM}:" ${MESSAGE_FILE}`"
if [ "${MOTD}" = "" ]
then
    # pick the default message-of-the-day
    MOTD="`grep  "DEFAULT:" ${MESSAGE_FILE}`"
    if [ "${MOTD}" = "" ]
    then
        # log message
        DisplayMessage E "${NO_MOTD}"
        echo "${DATETIME}:rmotd.sh:ERROR:No message-of-the-day
        ➤  found" >> ${RMOTD_LOG}
        return $FALSE
    else
        # strip the date from the default message
        MOTD="`echo $MOTD | cut -d':' -f2`"
        # set the DEFAULT flag to Y
        DEFAULT=Y
    fi
else
    # strip the date from the regular message
    MOTD="`echo $MOTD | cut -d':' -f2`"
fi

# prepare the message-of-the-day file
echo "
*********************************************************************
➤  ********


${MOTD}


*********************************************************************
➤  ********
   " > ${TEMP_FILE}
}
```

```
##############################################################################
#                                                                            #
# ReplaceMessageFile                                                         #
#                                                                            #
# This function replaces existing motd file with new one.                    #
#                                                                            #
##############################################################################
ReplaceMessageFile ( )
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP
mv ${MOTD_FILE} ${MOTD_FILE}.old
cp ${TEMP_FILE} ${MOTD_FILE}
if [ "${DEFAULT}" = "Y" ]
then
    DisplayMessage I "${DMOTD_ADDED}"
    echo "${DATETIME}:rmotd.sh:INFO:Default message-of-the-day
    ➤  added for ${DOTM}" >> ${RMOTD_LOG}
else
    DisplayMessage I "${MOTD_ADDED}"
    echo "${DATETIME}:rmotd.sh:INFO:New message-of-the-day added
    ➤  for ${DOTM}" >> ${RMOTD_LOG}
fi
}

##############################################################################
#                                                                            #
# RootUser                                                                   #
#                                                                            #
# This function checks that the user is root.                                #
#                                                                            #
# Returns :   TRUE if the 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
}

##############################################################################
#                                                                            #
# ProcessExit                                                                #
#                                                                            #
```

```
# This function removes temporary files and makes a graceful exit.  #
#                                                                    #
# Input   :   Exit Code                                              #
#                                                                    #
######################################################################
ProcessExit ()
{
EXIT_CODE="$1"
clear
rm -f ${TEMP_FILE}
exit ${EXIT_CODE}
}

######################################################################
#                                                                    #
# main                                                               #
#                                                                    #
# This function invokes all other functions.                         #
#                                                                    #
######################################################################
main ( )
{
InitializeVariables
if ! RootUser
then
    DisplayMessage E "${ROOT_USER}"
    ProcessExit $FEC
fi
if ! InitializeRMOTDLogFile
then
    ProcessExit $FEC
fi
if ! PrepareMessageFile
then
    ProcessExit $FEC
fi

ReplaceMessageFile
ProcessExit $SEC
}

# invoke main
main
```

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

Automating Oracle database set-up on AIX

The script below can be used to automate the setting up of an Oracle database on AIX.

The best way to illustrate the way the script works is by means of an example. In this example, we have a disk drive called */oracle1* and the *ORACLE_SID* of the database we're creating is 'dev'. The database requires the *ORACLE_BASE* and *ORACLE_HOME* variables to be set.

The script creates an OFA-compliant set-up, creating directories like *pfile/bdump/cdump/udump/create* under *$ORACLE_BASE/admin/$ORACLE_SID/*.

The script prompts the user for the names of the disks that are to hold the control files. At this point the user has to enter only the disk drive and not the full path (for example, */oracle1*). This entry is validated to ensure that the drive entered actually exists. The script then creates the control file by creating the directory */oracle1/ORACLE/dev* with control file *control01.ctl*.

The script asks for the same information for the other two control files as well. It then asks the user for other details, such as the *db_block_size* and whether the database is to be 'big' or 'small'.

It then adjusts the *init.ora* parameters to give an SGA of about 25 MB for a 'small' database and 50 MB for a 'big' one. This allows the *init.ora* file to be created. The parameters have been chosen to give optimum performance, but can be changed according to the requirements of the site after database creation.

All we have to do then is identify the drive that's to hold the *SYSTEM*, *RBS*, *TOOLS*, and *USERS* data files and the size of the files in megabytes. The script then creates files like:

```
/oracle1/ORACLE/dev/system01.dbf
```

The user input required by the script takes just a few minutes, after which database creation is fully automated.

The script also runs **catproc**, **catalog**, and other database housekeeping scripts. The script will then:

- Ask you to change the passwords for the users *SYS* and *SYSTEM* from their default.
- Create a user *OPS$ORACLE*, who is identified externally.
- Create other DBA accounts using names provided by the user.
- Update the *listener.ora* and *tnsnames.ora* files to add the database just created and also update the *oratab* file.
- Create a symbolic link in the *$ORACLE_HOME/dbs* directory for the *init.ora* file in $*ORACLE_BASE/admin/dev/pfile*.
- Give the user the option to change the logging mode of the database to *ARCHIVELOG* if we need it.

I have tested the script on Oracle 7.3 and Oracle 8 on AIX. The script makes the task of database creation, which is extremely tedious, easy and quick, with less room for human error. The only problem is that the user should have the required file and directory permissions to create the directories and sub-directories that are created by the script itself.

Any errors generated by the script are stored in the *createdb.log* file in the *$CREATE* directory.

To invoke the script, just run the file **createdb.sh**.

Note the use of the continuation character ('➤') in the code below to indicate that one line of code maps to more than one line of print.

CREATEDB.SH

```
#########################################################################
# Program Name: createdb.sh
# Description:  Automate the creation of Oracle Databases
# Platforms:    AIX
# Tested On:    Oracle7.3.x, Oracle8.0.4.1
# Date Written: 30-Jun-1999
# Author:       Gavin Soorma, Emirates Airline, Dubai
#########################################################################
```

```
#############################################################################
# Ensure that the variables $ORACLE_BASE, $ORACLE_HOME, and
# $ORACLE_SID are properly defined.
#
# Run the script when connected as user 'oracle'.
#
# Ensure user has the permissions necessary to create directories.
#
# FUNCTIONALITY:
# 1)  Create OFA-compliant directory structure under $ORACLE_BASE.
# 2)  Create customized init.ora file with optimum parameter settings.
# 3)  Give user the option to create either 'Big' or 'Small' databases.
# 4)  Initialize SGA to 25 MB or 50 MB for 'small' or 'big' databases.
# 5)  User is prompted only for the location and size of initial files
#     for database creation. Full path names are not necessary - only
#     the base directory of the file needs to be entered.
# 6)  Run the catproc, catexp, catalog, catblock, and pupbld scripts.
# 7)  Allow the user to change the default SYS and SYSTEM passwords.
# 8)  Create DBA accounts in the database.
# 9)  Create symbolic link in $ORACLE_HOME/dbs for init.ora.
# 10) Update oratab, tnsnames.ora, and listener.ora files.
# 11) Allow the user to change the archivelog mode of the database.
#
#############################################################################

# Validate the disk drive and loop until a valid one is entered.

init_create ()
{ echo "db_name=$db" >>$ORACLE_BASE/admin/$db/pfile/init$db.ora
choice=""
until [ -n "$choice" ]
do
  echo "Please Enter The Drive For Control File #1"
  read reply1
         if [ -d $reply1 ]
         then choice=TRUE
         fi
done
choice1=""
until [ -n "$choice1" ]
do
        echo "Please Enter The Drive For Control File #2"
        read reply2
                if [ -d $reply2 ]
                then choice1=TRUE
                fi
done
choice2=""
until [ -n "$choice2" ]
do
```

```
        echo "Please Enter The Drive For Control File #3"
        read reply3
                if [ -d $reply3 ]
                then choice2=TRUE
                fi
done

# Create sub-directories if they do not exist.
# The format is: base_directory/ORACLE/$ORACLE_SID
# Example: /oracle1/ORACLE/dev

if [ ! -d $reply1/ORACLE/$db ]
        then mkdir -p $reply1/ORACLE/$db
fi
if [ ! -d $reply2/ORACLE/$db ]
        then mkdir -p $reply2/ORACLE/$db
fi
if [ ! -d $reply3/ORACLE/$db ]
        then mkdir -p $reply3/ORACLE/$db
fi

# Set up the init.ora file
# The choice entered determines the size of the SGA

echo "control_files=($reply1/ORACLE/$db/control01.ctl
➤  ,$reply2/ORACLE/$db/control02.ctl,$reply3/ORACLE/$db/control03.ctl)"
➤  >>$ORACLE_BASE/admin/$db/pfile/init$db.ora
blockchoice=""
until [ -n "$blockchoice" ]
do
echo "Please enter the db_block_size in bytes (2048, 4096, or 8192)"
read dbblock
if [ $dbblock -eq 2048 -o $dbblock -eq 4096 -o $dbblock -eq 8192 ]
        then blockchoice=TRUE
fi
done
echo "db_block_size=$dbblock" >>
➤  $ORACLE_BASE/admin/$db/pfile/init$db.ora
echo " Please Enter The Database Type (B for Big , S for Small)"
read type
case $type in
b|B) echo "db_block_buffers=6000 "
     ➤  >>$ORACLE_BASE/admin/$db/pfile/init$db.ora
     echo "shared_pool_size=30000000" >>
     ➤  $ORACLE_BASE/admin/$db/pfile/init$db.ora
        ;;
s|S) echo "db_block_buffers=2000" >>
     ➤  $ORACLE_BASE/admin/$db/pfile/init$db.ora
     echo "shared_pool_size=15000000" >>
     ➤  $ORACLE_BASE/admin/$db/pfile/init$db.ora
```

```
        ;;
esac
echo "db_file_multiblock_read_count=`expr 64 \* 1024 / $dbblock`" >>
➤  $ORACLE_BASE/admin/$db/pfile/init$db.ora
echo "Please enter the Oracle version for the COMPATIBLE parameter"
read version
echo "Compatible=$version" >> $ORACLE_BASE/admin/$db/pfile/init$db.ora
cat >> $ORACLE_BASE/admin/$db/pfile/init$db.ora <<EOF
rollback_segments=(r01,r02,r03,r04)
log_checkpoint_interval = 1000000000
log_checkpoints_to_alert = TRUE
#checkpoint_process = TRUE
sort_direct_writes = TRUE
sort_write_buffers = 4
sort_write_buffer_size = 65536
sort_area_size = 1048576
sort_area_retained_size=65536
processes = 200
transactions=300
distributed_transactions=250
dml_locks = 200
log_buffer = 1048576
open_cursors=250
sequence_cache_entries = 10
#sequence_cache_hash_buckets = 10
open_cursors = 300
#audit_trail = true             # for auditing
timed_statistics = false        # for timed statistics
max_dump_file_size = 10240      # limit trace file size to 5 MB each
global_names=false
optimizer_mode=rule
log_archive_format=arch%s.log
remote_os_authent = true        # enable remote ops$ login
#remote_login_passwordfile= exclusive
os_authent_prefix=ops$
#db_writers = 2
#job_queue_processes=2
#job_queue_interval=240
#job_queue_keep_connections=FALSE
nls_date_format=DD-MON-RR
background_dump_dest=$ORACLE_BASE/admin/$db/bdump
user_dump_dest=$ORACLE_BASE/admin/$db/udump
core_dump_dest=$ORACLE_BASE/admin/$db/cdump
EOF
}

tput clear
echo "DATABASE CREATION - please enter the information requested "
echo "*********************************************************"
echo
```

```
echo "Please note:"
echo
echo "* Ensure that the variables ORACLE_HOME, ORACLE_BASE, etc
      ➤  are set."
echo "* Please set COMPATIBLE to the current version of Oracle"
echo "* Please check the file \$CREATE/createdb.log for errors"
echo
echo "Checking enviroment variables ..."
echo "ORACLE_BASE is `echo $ORACLE_BASE`"
echo "ORACLE_HOME is `echo $ORACLE_HOME`"
echo "Proceed? (Y/N) "
read go
if [ "$go"  = n -o "$go" = N ]
        then exit
fi
echo "Please enter the SID of the database"
read db
echo "Creating directories and subdirectories for database $db ..."
if [ ! -d $ORACLE_BASE/admin ]
        then mkdir -p  $ORACLE_BASE/admin
fi
if [ ! -d $ORACLE_BASE/admin/$db ]
        then mkdir -p $ORACLE_BASE/admin/$db
        cd $ORACLE_BASE/admin/$db
        mkdir pfile udump bdump cdump create
fi
echo Directories and subdirectories created

#main -calls function init_create

echo " Checking for init$db.ora ..."
if [ ! -f $ORACLE_BASE/admin/$db/pfile/init$db.ora ]
then
        echo " init$db.ora not found ... creating ..."
        init_create
fi
if [  ! -d $ORACLE_BASE/scripts ]
then
         echo "Creating $SCRPT directory"
         mkdir $ORACLE_BASE/scripts
         else echo Scripts directory already exists
fi
if [ ! -f $ORACLE_BASE/scripts/set$db ]
then
        echo " set$db.ora not found ... creating ..."
cat >> $ORACLE_BASE/scripts/set$db <<EOF
ORACLE_SID=$db;export ORACLE_SID
PFILE=$ORACLE_BASE/admin/$db/pfile; export PFILE
BDUMP=$ORACLE_BASE/admin/$db/bdump; export BDUMP
UDUMP=$ORACLE_BASE/admin/$db/udump; export UDUMP
```

```
CDUMP=$ORACLE_BASE/admin/$db/cdump; export CDUMP
CREATE=$ORACLE_BASE/admin/$db/create; export CREATE
EOF
fi
. $ORACLE_BASE/scripts/set$db
echo "ORACLE_SID is now set to $ORACLE_SID"
choice3=""
until [ -n "$choice3" ]
do
echo Please enter the location for the SYSTEM datafile
read sys
if [ -d $sys ]
then choice3=TRUE
fi
done
if [ ! -d $sys/ORACLE/$db ]
then mkdir -p $sys/ORACLE/$db
fi
sysfile="'$sys/ORACLE/$db/system01.dbf'"
echo $sysfile
echo "Please enter the size of the datafile in megabytes"
read syssize
syssize1=`echo $syssize |tr -d '[a-z]'|tr -d '[A-Z]'`
A=`echo "datafile $sysfile" size "$syssize1" MB`
choice4=""
until [ -n "$choice4" ]
do
echo Please enter the location for the RBS datafile
read rbs
if [ -d $rbs ]
then choice4=TRUE
fi
done
if [ ! -d $rbs/ORACLE/$db ]
then mkdir -p $rbs/ORACLE/$db
fi
rbsfile="'$rbs/ORACLE/$db/rbs01.dbf'"
echo $rbsfile
echo "Please enter the size of the datafile in megabytes"
read rbssize
rbssize1=`echo $rbssize |tr -d '[a-z]'|tr -d '[A-Z]'`
B=`echo "datafile $rbsfile" size "$rbssize1" MB`
choice5=""
until [ -n "$choice5" ]
do
echo Please enter the location for the TEMP datafile
read tmp
if [ -d $tmp ]
then choice5=TRUE
fi
```

```
done
if [ ! -d $tmp/ORACLE/$db ]
then mkdir -p $tmp/ORACLE/$db
fi
tmpfile="'$tmp/ORACLE/$db/temp01.dbf'"
echo $tmpfile
echo "Please enter the size of the datafile in megabytes"
read tmpsize
tmpsize1=`echo $tmpsize |tr -d '[a-z]'|tr -d '[A-Z]'`
C=`echo "Datafile $tmpfile" size "$tmpsize1" MB`
choice6=""
until [ -n "$choice6" ]
do
echo Please enter the location for the TOOLS datafile
read tll
if [ -d $tll ]
then choice6=TRUE
fi
done
if [ ! -d $tll/ORACLE/$db ]
then mkdir -p $tll/ORACLE/$db
fi
tllfile="'$tll/ORACLE/$db/tools01.dbf'"
echo $tllfile
echo "Please enter the size of the datafile in megabytes"
read tllsize
tllsize1=`echo $tllsize |tr -d '[a-z]'|tr -d '[A-Z]'`
D=`echo "datafile $tllfile" size "$tllsize1" MB`
choice7=""
until [ -n "$choice7" ]
do
echo Please enter the location for the USERS datafile
read usr
if [ -d $usr ]
then choice7=TRUE
fi
done
if [ ! -d $usr/ORACLE/$db ]
then mkdir -p $usr/ORACLE/$db
fi
usrfile="'$usr/ORACLE/$db/users01.dbf'"
echo $usrfile
echo "Please enter the size of the datafile in megabytes"
read usrsize
usrsize1=`echo $usrsize |tr -d '[a-z]'|tr -d '[A-Z]'`
E=`echo "datafile $usrfile" size "$usrsize1" MB`
choice8=""
until [ -n "$choice8" ]
do
echo Please enter the path of the 'redo' log files
```

```
read redo
if [ -d $redo ]
then choice8=TRUE
fi
done
if [ ! -d $redo/ORACLE/$db ]
then mkdir -p $redo/ORACLE/$db
fi
r1="'$redo/ORACLE/$db/redo01.log'"
r2="'$redo/ORACLE/$db/redo02.log'"
r3="'$redo/ORACLE/$db/redo03.log'"
echo "Please enter the size of the redo log files with
➤  the suffix 'k' or 'm'"
read redosize
echo $r1 size $redosize,
echo $r2 size $redosize,
echo $r3 size $redosize
echo "Starting database creation ..."
echo "Please wait - this takes approximately 15 minutes, so
➤  grab a coffee!"
$ORACLE_HOME/bin/svrmgrl <<EOF
spool $ORACLE_BASE/admin/$db/create/createdb.log
connect internal
startup nomount pfile=$ORACLE_BASE/admin/$db/pfile/init$db.ora
create database "$db"
    maxinstances 1
    maxlogfiles  16
    maxdatafiles 200
    character set "US7ASCII"
    $A
    logfile
 $r1 size $redosize,
 $r2 size $redosize,
 $r3 size $redosize ;
create rollback segment r0 tablespace system
storage (initial 16k next 16k minextents 2 maxextents 20);
alter rollback segment r0 online;
create tablespace rbs
$B
default storage (
    initial      1m
    next         1m
    pctincrease    0
    minextents     10
);
create tablespace temp temporary
$C
default storage (
    initial      1024k
    next         1024k
```

```
        pctincrease  0
maxextents unlimited
);
create tablespace tools
 $D ;

create tablespace users
$E ;
create rollback segment r01 tablespace rbs;
create rollback segment r02 tablespace rbs;
create rollback segment r03 tablespace rbs;
create rollback segment r04 tablespace rbs;
 alter rollback segment r01 storage (optimal 10m);
 alter rollback segment r02 storage (optimal 10m);
 alter rollback segment r03 storage (optimal 10m);
 alter rollback segment r04 storage (optimal 10m);
alter rollback segment r01 online;
alter rollback segment r02 online;
alter rollback segment r03 online;
alter rollback segment r04 online;
alter rollback segment r0 offline;
drop rollback segment r0;
alter user sys temporary tablespace temp;
alter user system default tablespace tools temporary tablespace temp;
create user ops\$oracle
identified externally
default tablespace tools
temporary tablespace temp
quota unlimited on  tools;
grant dba to ops\$oracle;
connect sys/change_on_install ;
@$ORACLE_HOME/rdbms/admin/catalog.sql;
@$ORACLE_HOME/rdbms/admin/catproc.sql;
@$ORACLE_HOME/rdbms/admin/catblock.sql;
@$ORACLE_HOME/rdbms/admin/catexp.sql
connect system/manager ;
@$ORACLE_HOME/sqlplus/admin/pupbld.sql;
 spool off
EOF
echo "Please enter the names of the DBAs"
read dbalist
echo "Creating DBA accounts ..."
for i in $dbalist
do
sqlplus -s system/manager  <<EOF
create user ops\$$i identified externally
default tablespace users
temporary tablespace temp
quota unlimited on users;
grant dba to ops\$$i;
```

```
EOF
done
echo Please enter the password for SYS
read pwd1
syspwd=$pwd1
sqlplus -s system/manager <<EOF
alter user sys identified by $syspwd ;
EOF
 echo "Password Set For SYS"
echo Please enter the password for SYSTEM
read pwd2
systempwd=$pwd2
sqlplus -s sys/$syspwd <<EOF
alter user system identified by $systempwd ;
EOF
 echo "Password Set For SYSTEM"
echo "Updating the network configuration files"
echo "Amending ORATAB ...."
if
        test -f /var/opt/oracle/oratab
        then
        ORATAB=/var/opt/oracle/oratab
        else
        ORATAB=/etc/oratab
fi
echo "Please enter a brief description for the new database"
read desc
dbcap=`echo $db | tr '[a-z]' '[A-Z]'`
cat >> $ORATAB <<EOF
$db:$ORACLE_HOME:Y:$desc:$dbcap
EOF
echo "Amending tnsnames.ora"
if
        test -f /var/opt/oracle/tnsnames.ora
        then
        TNS=/var/opt/oracle/tnsnames.ora
        else
        TNS=/etc/tnsnames.ora
fi
echo "Please enter the TNS alias ..."
read alias
echo "Updating tnsnames.ora ..."
cat >> $TNS  <<EOF
$alias =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = TCP)
          (PROTOCOL = TCP)
          (HOST = `hostname`)
```

```
          (PORT = 1640)
        )
    )
    (CONNECT_DATA =
      (SID = $db)
     )
  )
EOF
echo "tnsnames.ora updated"
echo " Please copy the listener.ora file from /tmp either to /etc"
cat >> /tmp/listener.ora <<EOF
`hostname` =
  (ADDRESS_LIST =
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY=`hostname`)
        )
        (ADDRESS =
          (PROTOCOL = TCP)
          (HOST = `hostname`)
          (PORT = 1640)
        )
  )

STARTUP_WAIT_TIME_`hostname` = 0
CONNECT_TIMEOUT_`hostname` = 10

SID_LIST_`hostname` =
  (SID_LIST =
EOF
if
        test -f /var/opt/oracle/oratab
        then
        ORATAB=/var/opt/oracle/oratab
        else
        ORATAB=/etc/oratab
fi
cat $ORATAB | while read LINE
do
LISTENER_SID=`echo $LINE | awk -F: '{ print $1 }'`
LISTENER_HOME=`echo $LINE | awk -F: '{ print $2 }'`
cat >> /tmp/listener.ora <<EOC
 (SID_DESC =
      (SID_NAME = $LISTENER_SID)
      (ORACLE_HOME=$LISTENER_HOME)
    )
EOC
```

```
done
cat >> /tmp/listener.ora <<EOF
)
EOF
echo "Creating symbolic link"
 cd $ORACLE_HOME/dbs
ln -s $ORACLE_BASE/admin/$db/pfile/init$db.ora init$db.ora
if test $? -eq 0 ;
then echo Symbolic link created
else echo Failed to create link
fi
echo "Do you wish to run the database in ARCHIVELOG mode?"
read arcreply
if [ "$arcreply" = y -o "$arcreply" = Y ]
then
archchoice=""
until [ -n "$archchoice" ]
do
 echo "Please enter the disk to hold the archive log files"
        read arcdest
if [ -d $arcdest ]
then archchoice=TRUE
if [ ! -d $arcdest/ORACLE/$db/arch ]
then mkdir -p $arcdest/ORACLE/$db/arch
fi
echo "The LOG_ARCHIVE_DEST is $arcdest/ORACLE/$db/arch"
echo "log_archive_dest=$arcdest/ORACLE/$db/arch" >>$ORACLE_BASE/admin/
$db/pfile/init$db.ora
echo "log_archive_start=TRUE" >>$ORACLE_BASE/admin/$db/pfile/init$db.ora
fi
done
$ORACLE_HOME/bin/svrmgrl << EOF
connect internal
shutdown immediate
startup mount
alter database archivelog;
alter database open;
EOF
else exit
fi
echo "******** SCRIPT OVER *********"
```

Gavin Soorma
Database Support Controller
Emirates Airline (UAE)

Who was on?

Login records are written to the file */etc/utmp*, and also to the file */var/adm/wtmp* (if it exists) along with *inittab* records. While the *utmp* file is just a record of who is logged on at present, the *wtmp* file is cumulative and continues to grow. This can result in */var* filling up unless the file is periodically pruned. For instance, ours grows at the rate of about 1 KB per user per day, though this does depend on how often users log off and log on again.

We manage the file by copying it to *wtmp.old* and then zeroing the file itself once a week just prior to shutdown and reboot. We thus have between seven and 14 days' back information at any time.

There are a number of ways of interrogating *wtmp*. For example, **who -u /var/adm/wtmp** gives a list of all those who logged on during the life of the file together with the device on which they logged on, when they logged on, and the *pid* and host. It also gives an idle time such as you'd get by using the command **who -u**, though the value reported doesn't seem to relate to anything. The command **ac** gets connect times from *wtmp* (either the total connect time or connect time by user), and **acctcon1**, which is normally called by **runacct** for accounting purposes, does a similar job.

What none of these commands can tell you is who was actually logged on at any particular time, even though this information is in the file.

To find this, it is necessary to analyse the records in *wtmp*. While this is a binary file, the command **/usr/lib/acct/fwtmp** is provided to turn it into ASCII format. This is normally used to mend a corrupted *wtmp* file, as it can also be used to rebuild the binary file from an ASCII file (using the **command /usr/lib/acct/fwtmp -ic <ascii.file >/var/adm/wtmp**). However, it can also be used as the starting point of an analysis.

There are ten record types in *wtmp*, which are described in the file */usr/include/utmp.h*. Two of them, types '3' and '4', are used only when the system clock is changed, as they record the old time and new time

respectively. Records of this type will probably be present in your file if you've been doing year 2000 testing.

Type '0', also known as 'EMPTY', appears at shutdown. Types '1' and '2' come from *inittab* and record run level and boot time respectively. *Inittab* also produces type '5' records as it launches each process and type '8' records when a process completes – for example, those specified as 'wait' or 'once' in */etc/inittab*. Type '9' records are for accounting and appear when accounting stops just before shutdown.

The remaining types, '6' and '7', together with type '8' mentioned above, are those used when users log on and off. Type '5' is a process waiting for a login (in other words, a 'connect'). Type '6' is a user process, and thus marks a successful login. At disconnect, a type '8' record is produced. These are the records that the script analyses. A complete connect-login-disconnect sequence should show three time entries, though one or more may be absent for various reasons.

A blank disconnect time means that the user is still logged on. A blank login time means a failed login attempt that was broken off. A connect time may be absent under certain circumstances, for example for **ftp** sessions. The script actually forces the login time into the connect field if the connect time is absent, as it improves sorting. However, if necessary this could be left out.

Because we archive and re-initialize the *wtmp* file at 10.30pm every Sunday, followed by a reboot, the new *wtmp* file contains some *inittab* type '8' records. To cope with this, the script contains a test for Sundays to ensure it looks at the old file rather than the new one. This needs changing if the file is archived at a different time.

The script requests a date and accepts as input null (which defaults to today), a day number (defaults to current month), or month and day. The output is a file in */tmp*, so it is there for printing, if required. Piping the output straight to **pg** might be marginally faster.

The script has been tested at AIX 4.2.1, though there is no reason why it shouldn't work at any level provided the *wtmp* record layout isn't changed.

Note the use of the continuation character, '➤', in the script below to indicate that one line of code maps to more than one line of print.

WHOWASON.SC

```
#!/usr/bin/ksh
# 28-07-98    whowason       show all logins for given date

cls
echo "\t\t\c"
date
tdy="`date '+%h %d'`"
echo "\n\nwhich day? [$tdy] \c"
read dat
[ -z "$dat" ] && dat=$tdy

# format date

set $dat
mm=$1
typeset -R2 dd=$2
if [ -z "$2" ]
then
  dd=$mm
  mm=`date '+%h'`
else
  init=`echo $mm|cut -c1|tr '[a-z]' '[A-Z]'`
  rest=`echo $mm|cut -c2-`
  mm=$init$rest
fi
((dd=$dd+0))                                # remove leading zeros
dat="$mm $dd"

# get records

lll=`/usr/lib/acct/fwtmp </var/adm/wtmp|grep -E " 6 | 7 | 8 "|
➤  grep "$dat"`
#echo "$lll">/tmp/lll
dwk=`echo "$lll"|tail -1|cut -c85-87`
if [ -z "$lll" -o "$dwk" = Sun ]            # pre reboot Sunday
then
  lll=`/usr/lib/acct/fwtmp </var/adm/wtmp.old|grep -E " 6 | 7 | 8 "|
➤  grep "$dat"`
fi
if [ -z "$lll" ]
then
  echo "\nno records for that date\n"
```

```
  exit
fi

# get display option

echo
echo "$lll"|tail -1|cut -c85-95,105-108
echo
echo "sort by 1 connect time, 2 logon time, 3 logoff time, 4 user,
➤  5 pts? [1] \c"
read tt
echo ok, please wait
[ -z "$tt" ] && tt=1
case $tt in
  1 ) tt="0.20 -0.28" ;;
  2 ) tt="0.30 -0.38" ;;
  3 ) tt="0.40 -0.48 +0.20 -0.28" ;;
  4 ) tt="0 -1 +0.20 -0.28" ;;
  5 ) tt="0.14n" ;;
  * ) exit ;;
esac

# set output fields

file=ww$$
typeset -L8 usr
typeset -L8 dev
typeset -L8 ti1
typeset -L8 ti2
typeset -L8 ti3
pid=0

prout ()
{
  echo "$usr  $dev  $ti1  $ti2  $ti3  $host"
  usr=""
  ti1=""
  ti2=""
  ti3=""
  host=""
}

# extract

(
echo "\nuser\t  device    connect   logon\tdisconn\t  from"

(
```

```
echo "$lll"|sort +0.40 -0.45 +0.38 -0.39|while read line
do
  [ -z "$line" ] && continue
  set $line
  if [ $# -eq 14 ]
  then
    usn=$1
    shift
  fi
  if [ $4 -ne $pid ]
  then
    prout
    pid=$4
  fi
  case $3 in
  6 )
    usr=$usn
    usn=""
    dev=$2
    host=$8
    ti1=${12}
    ;;
  7 )
    usr=$usn
    usn=""
    dev=$2
    host=$8
    ti2=${12}
    [ x$ti1 = x ] && ti1=$ti2            # force connect-time for sort
    ;;
  8 )
    ti3=${11}
    ;;
  esac
done
prout
)|sort +$tt|pg -22

echo "\nuser\t  device    connect   logon\tdisconn\t  from"
)>/tmp/$file

pg -e /tmp/$file
```

Hugh Walding
Senior Systems Officer
Isle of Wight Council (UK)

SSCCARS

INTRODUCTION

SSCCARS (Server Source Code Control And Release System) is a custom source code control and release system. The utility is particularly useful on a server that keeps several program sources (eg Oracle PL/SQL program sources) that are constantly being updated and need to be released to many sites. The utility has an interface to an Oracle database that is used for reporting on SSCCARS-related information (eg details of checked-out source, build releases, etc) through a set of Gupta forms. This article explains the back-end of this utility, which has been developed almost entirely in shell script (the exception being one C module).

SSCCARS FUNCTIONALITY

- Check in new source.
- Check in existing source.
- Check out read-only copy of latest source.
- Check out read-only copy of specific source.
- View list of all checked-out source code.
- View the source-file-to-source-name mapping file.
- Initialize the SSCCARS log file.
- Initialize source-file-to-source-name mapping file.
- Cancel a booked-out copy.
- Build a patch release.
- Build a full release.
- View log file for build release.
- Install a release to distributed sites.

- View the log file for an installed release.

SSCCARS MODULES

The utility comprises the following programs:

- **ssccarss.sh** main module.
- **ssccarss_lib.sh** library module.
- **chkout.sh** source check-out module.
- **chkin.sh** source check-in module.
- **execsu.c** superuser execution module.
- **br.sh** build release module.
- **iron.sh** (install release over network) install release module.

HOW TO MAKE IT WORK

1 Decide what type of source is to be managed.

2 Edit the value of variable *$FILE_EXTS* to reflect this (the source that needs to be managed must have a file extension defined in *$FILE_EXTS*).

3 Create the directory structure below according to your choice in step 1.

```
Directory name          Variable

main                    SSCCARS_DIR
Source                  SOURCE_DIR=${SSCCARS_DIR}/source
Trigger source          TRIG_DIR=${SOURCE_DIR}/trig
Procedure source        PROC_DIR=${SOURCE_DIR}/proc
View source             VIEW_DIR=${SOURCE_DIR}/view
Shell script source     SH_DIR=${SOURCE_DIR}/sh
Awk script source       AWK_DIR=${SOURCE_DIR}/awk
Pro*C source            PC_DIR=${SOURCE_DIR}/pc
C source                C_DIR=${SOURCE_DIR}/c
SQL source              SQL_DIR=${SOURCE_DIR}/sql
SSCCARS log             LOG_DIR=${SSCCARS_DIR}/log
SSCCARS lock            LOCK_DIR=${SSCCARS_DIR}/lock
Temporary directory     TEMP_DIR="${SSCCARS_DIR}/temp
Release directory       ROOT_RELEASE_DIR=${SSCCARS_DIR}/release
Release log             RELEASE_LOG_DIR=${ROOT_RELEASE_DIR}/log
```

4 Set the variable *$SSCCARS _DIR* in module **ssccars_lib.sh**.

5 Set the directory variable *$SSCCARS_BIN* in the *main()* block of the **ssccars.sh** module.

6 Put all the modules in directory *$SSCCARS_BIN*.

7 Comment out calls to Oracle database interface functions, if not required.

8 If Oracle database interface functions are required, re-work these functions.

9 Set the value of *$AUTHORISED_USER* in **ssccars_lib.sh** module as required.

NOTES

1 When the main module **ssccars.sh** is run, it performs a 'sanity check' to establish that the utility is in an executable state. Therefore, if anything is missing, you'll soon discover it.

DATABASE INTERFACE

The utility interfaces with an Oracle database through the following library functions:

- *SmartDatabaseStatus*
- *UpdateSmart*.

These two functions must be modified to make this database interface work with other databases. These two functions are located in the **ssccs_lib.sh** module.

SSCCARS MESSAGES

By default, every message that is displayed on the screen must be acknowledged by the user. However, from time to time, SSCCARS displays messages that users must not acknowledge. These messages have an extra piece of text appended to them that says: 'Do not acknowledge this message'.

MODULE DESCRIPTION AND SOURCE LISTING

ssccars.sh is the main module that calls all other modules. The following menu options in this module require authorized privilege:

- Initialize the SSCCARS log file.
- Initialize source-name-to-source-file mapping file.
- Build a release.
- Install a release.
- Cancel booking for modifiable source.

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

SSCCARS.SH

```
#! /bin/ksh
###############################################################################
# Name        : ssccars.sh
#
# Decsription : A utility to manage the server source codes
#               and its release and implementation.
#
# Notes       : 1 The script  calls the following modules:
#                     o chkout.sh
#                     o chkin.sh
#                     o br.sh
#                     o iron.sh
#                     o execsu
#                     o ssccars_lib.sh
#
#               2 The script contains the following functions:
#                     o main
#                     o DefineModuleVariables
#                     o DisplayMenuOptions
#                     o HandleInterrupt
#                     o ProcessOverallExit
#                     o DisplayLocalMessage
#                     o Movecursor
#                     o ProcessMenuOption
#                     o CancelCheckedOutBooking
#                     o InitialiseSSCCARSSLogFile
#                     o InitializeSourceNameMappingFile
#                     o InitializePatchBuiltDateFile
#                     o ValidateLastPatchBuiltDate
```

```
#                       o ViewCheckedOutSourceNames
#                       o ViewCheckinCheckoutLog
#                       o ViewSourceNameMappingFile
#                       o BuildRelease
#                       o InstallReleaseOverNetwork
#                       o ViewBuiltReleaseLogFile
#                       o ViewInstalledReleaseLogFile
################################################################################
################################################################################
# Name        : DefineModuleVariables
#
# Decsription : Define all the module variables
################################################################################
DefineModuleVariables ()
{
DATETIME=`date "+%d/%m/%y at %H:%M:%S"`
# Define escape sequences
ESC="\0033["   ; export ESC
RVON=_[7m      ; export RVON            # revrese video on
RVOFF=_[27m    ; export RVOFF           # reverse video off
BOLDON=_[1m    ; export BOLDON          # bold on
BOLDOFF=_[22m  ; export BOLDOFF         # bold off
BON=_[5m       ; export BON             # blinking on
BOFF=_[25m     ; export BOFF            # blinking off
ERROR="${RVON}${BON}ssccars.sh:ERROR:${BOFF}"
INFO="${RVON}ssccars.sh:INFO: "
TEMP_FILE="${TEMP_DIR}/ssccars_$$.tmp" ; export TEMP_FILE
LIBRARY_MISSING="Library ${SSCCS_BIN}/ssccars_lib.sh is
➤  missing${RVOFF}"
LIBRARY_NOT_EXECUTABLE="Library ${SSCCS_BIN}/ssccars_lib.sh is
➤  not executable by user${RVOFF}"
OS_ERROR="\$ERROR_MSG${RVOFF}"
}
################################################################################
# Name        : HandleInterrupt
#
# Overview    : Call ProcessOverallExit
################################################################################
HandleInterrupt ()
{
DisplayMessage I "${INTERRUPT}"
ProcessOverallExit  $FEC
}
################################################################################
# Name        : ProcessOverallExit
#
# Overview    : Remove temporary files and make a graceful exit
#
# Input       : Exit code ($SEC or  $FEC)
################################################################################
```

```
ProcessOverallExit ()
{
rm -f  ${TEMP_DIR}/*ssccars*
rm -f  ${TEMP_DIR}/*chkout*
rm -f  ${TEMP_DIR}/*chkin*
rm -f  ${TEMP_DIR}/*br*
rm -f  ${TEMP_DIR}/*iron*
exit $EXIT_CODE
}
################################################################################
# Name     : DisplayLocalMessage
#
# Overview : Display message about checking the library module.
#
# Input    : 1 Message type (E = Error, I = Information)
#            2 Error code as defined in DefineModuleVariables ().
################################################################################
DisplayLocalMessage ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
MESSAGE_TYPE="$1"
MESSAGE_TEXT=`eval echo $2`
clear
MoveCursor 24 1
# display the message
if [ "${MESSAGE_TYPE}" = "E" ]
then
   echo "`eval echo ${ERROR}`${MESSAGE_TEXT}\c"
else
   echo "`eval echo ${INFO}`${MESSAGE_TEXT}\c"
fi
read  DUMMY
return ${TRUE}
}
################################################################################
# Name        : MoveCursor
#
# Input       : Y and X coordinates
#
# Overview    : Move the cursor to (Y, X)
#
# Notes       : 1 Must run in ksh for print to work. Print is
#                   used to move the cursor as echo doesn't work
################################################################################
MoveCursor ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
YCOR=$1
XCOR=$2
echo "${ESC}${YCOR};${XCOR}H"
```

```
}
################################################################################
# Name        : DisplayMenuOptions
#
# Decsription : Display options
################################################################################
DisplayMenuOptions ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
clear
echo ""
echo "        ##########################################################"
echo "        #                                                        #"
echo "        # ${SSCCARS}                                             #"
echo "        #                                                        #"
echo "        #  5. Check out source                                   #"
echo "        # 10. Check in source                                    #"
echo "        # 15. View list of checked-out sources                   #"
echo "        # 20. View check-in/check-out log                        #"
echo "        # 25. View source file to source name mapping            #"
echo "        # 30. Cancel booking for modifying source                #"
echo "        # 35. Initialize log file                                #"
echo "        # 40. InitialiseSource name mapping file                 #"
echo "        # 45. Build patch release                                #"
echo "        # 50. Build full release                                 #"
echo "        # 55. View built release log file                        #"
echo "        # 60. Initialize last patch built date file              #"
echo "        # 65. View last patch build date file                    #"
echo "        # 70. Install release                                    #"
echo "        # 75. View installed release log file                    #"
echo "        # 99. Exit                                               #"
echo "        #                                                        #"
echo "        # Please acknowledge the message displayed               #"
echo "        ##########################################################"
echo "                Enter Option ---->\c"
read OPTION
}
################################################################################
# Name        : ViewCheckedOutSourceNames
#
# Decsription : List all source files that are checked-out for
#               modification, sorting by by date
################################################################################
ViewCheckedOutSourceNames ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# create a temporary file with read-only permission
> ${TEMP_FILE}
echo "The following files are checked out for modification" >> \
                                $TEMP_FILE
```

```
echo " ==========================================================\n" >> \
                                                          $TEMP_FILE
echo "  MONTH     DAY      TIME     SOURCE" >> $TEMP_FILE
echo "  =====     ===      ====     ======" >> $TEMP_FILE
ls -lt $LOCK_DIR | \
   awk {'print "  "  $6 "       " $7 "      " $8 "   " $9'} >>
➤  $TEMP_FILE
chmod 444 ${TEMP_FILE}
view $TEMP_FILE
rm ${TEMP_FILE} <<!
y
!
}
#############################################################################
# Name        : ViewCheckInCheckOutLog
#
# Decsription : List all source files that are checked in or out
#               for modification, sorting by date/time.
#############################################################################
ViewCheckInCheckOutLog ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
if [ ! -f "$LOG_DIR/$LOG_FILE" ]
then
     DisplayMessage E "${NO_LOG_FILE}"
     return $FALSE
fi
cp $LOG_DIR/$LOG_FILE ${TEMP_FILE}
view ${TEMP_FILE}
}
#############################################################################
# Name        : ViewSourceNameMappingFile
#
# Decsription : Display the file of name mappings
#############################################################################
ViewSourceNameMappingFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP  $SIGTSTP
# check for file existence
if [ ! -f $LOG_DIR/$DATA_FILE ]
then
     DisplayMessage E "${NO_DATA_FILE}"
     return $FALSE
fi
# copy the file
cp $LOG_DIR/$DATA_FILE ${TEMP_FILE}
view ${TEMP_FILE}
}
#############################################################################
# Name        : CancelCheckedOutBooking
```

```
#
# Description : Cancel the registration of a checked out source
#               by releasing the lock on file
#
# Notes       : 1 The function checks the directory $LOCK_DIR to
#                 establish whether the source file name is already
#                 there. If it is, it removes the file from the
#                 directory and updates the log.
#
#               2 Only available to authorized users
#
#               3 The function calls the following functions:
#                   o CheckAuthorisedUserId
#                   o GetSourceName
#                   o ProcessFileExtension
#                   o GetLatestVersion UPDATE
#                   o CheckLock ${TARGET_SOURCE}
#                   o SmartDatabaseStatus
#                   o SeekConfirmation
#                   o FreeLock "${TARGET_SOURCE}"
#                   o UpdateSmart "CANCEL"
#                   o DisplayMessage
#
#               4 The function calls the following C module:
#                   o execsu
##############################################################################
CancelCheckedOutBooking ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# check for authorized userid
if ! CheckAuthorisedUserId
then
     return $FALSE
fi
# get source name
if ! GetSourceName "CO"
then
      return $FALSE
fi
# process file extension
if ! ProcessFileExtension
then
     return $FALSE
fi
# get latest version number for the source
if !  GetLatestVersion UPDATE
then
     return $FALSE
fi
# establish file details
```

```
REQ_DIR="${SOURCE_DIR}/${SOURCE_EXT}"
REQ_FILE="${REQ_DIR}/${REQ_SOURCE}"
TARGET_FILE="${TARGET_DIR}/${TARGET_SOURCE}"
SOURCE_VERSION=`echo $TARGET_SOURCE |  sed  's/.*\_//' | cut -d'.' -f1`
# check the lock on the source
if  ! CheckLock ${TARGET_SOURCE}
then
    # file is not locked
    DisplayMessage E "${SOURCE_NOT_CHECKED_OUT}"
    return $FALSE
fi
# check database status
if  ! SmartDatabaseStatus
then
    DisplayMessage  E  "${DB_NOT_OK}"
    return $FALSE
fi
# get confirmation
MESSAGE="Are you sure you want to cancel the booking? (Y/N):\c"
if ! SeekConfirmation
then
   return $FALSE
fi
if !  FreeLock "${TARGET_SOURCE}"
then
    return $FALSE
fi
# update log
${SSCCARS_BIN}/execsu "UL" ${LOG_FILE} ${LOG_DIR} ${LOG_DAY}
➤  ${LOG_TIME} ${USERID} ${TARGET_SOURCE} "LOCKED SOURCE RELEASED"
➤  > ${TEMP_FILE2} 2>&1
if [ $? -ne 0 ]
then
     # undo everything
     ERROR_MSG=`cat ${TEMP_FILE2} | head -1`
     DisplayMessage E "${EXECSU_ERROR}"
     LockSource ${REQ_SOURCE}
     return $FALSE
fi
# update smart database
if  UpdateSmart "CANCEL"
then
     DisplayMessage I "${SMART_UPDATED}"
else
     DisplayMessage E "${SMART_NOT_UPDATED}"
fi
return $TRUE
}
##############################################################################
# Name        : InitializeSSCCARSLogFile
```

```
#
# Decsription : The function initializes the SSCCARS log file.
#
# Notes       : 1 The function calls the following functions:
#                   o CheckRootUserId
#                   o SeekConfirmation
#                   o Display Message
#############################################################################
InitializeSSCCARSLogFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# check user id
if ! CheckRootUserId
then
   return $FALSE
fi
# check for file existence
if [ -f $LOG_DIR/$LOG_FILE ]
then
   #  file exists
   MESSAGE="Do you wish to re-initialize existing log file? (Y/N):\c"
   if ! SeekConfirmation
   then
        return $FALSE
   fi
fi
# initialize the file
echo "
                  CHECKIN/CHECKOUT LOG
                  ====================
DAY      TIME        DEVELOPER      MODULE                 COMMENT
-----------------------------------------------------------------------
➤  ----------
"  >  $LOG_DIR/$LOG_FILE
DisplayMessage  I "${LOG_INITIALIZED}"
return $TRUE
}
#############################################################################
# Name        : InitializeSourceNameMappingFile
#
# Decsription : The function initializes the SSCCARS source name
#               mapping file.
#
# Notes       : 1 The function calls the following functions:
#                   o CheckRootUserId
#                   o SeekConfirmation
#                   o DisplayMessage
#############################################################################
InitializeSourceNameMappingFile ()
{
```

```
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# check user id
if ! CheckRootUserId
then
   return $FALSE
fi
# check for file existence
if  [ -f  $LOG_DIR/$DATA_FILE ]
then
   # file exists
   MESSAGE="Do you wish to re-initialize existing map file(Y/N):\c"
   if ! SeekConfirmation
   then
       return $FALSE
   fi
fi
#  initialize the file
echo "
                  SOURCE FILE TO SOURCE NAME MAPPING
                  ==================================
File name             Source Name                      Descriptions
-------------------------------------------------------------------
➤  -----------
"  >  $LOG_DIR/$DATA_FILE
DisplayMessage I "${MAP_FILE_INITIALIZED}"
return $TRUE
}
#############################################################################
# Name        : BuildRelease
#
# Decsription : The function builds a release by calling the script
#               br.sh using a parameter of type release.
#
# Input       : Release Type (F = full release; P = patch release)
#
# Notes       : 1 The function calls the following functions:
#                   o CheckAuthorisedUserId
#
#               2 The function calls the following module:
#                   o  br.sh ${P_RELEASE_TYPE}
#############################################################################
BuildRelease ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_RELEASE_TYPE="$1"
# check authorized  userid
if ! CheckAuthorizedUserId
then
     return $FALSE
fi
```

```
# call br.sh with the input parameter
. ${SSCCARS_BIN}/br.sh "${P_RELEASE_TYPE}"
return $TRUE
}
##############################################################################
# Name        : InstallReleaseOverNetwork
#
# Decsription : The function installs a specific release by calling
#               the script iron.sh.
#
# Notes       : 1 The function calls the following function:
#                   o CheckAuthorisedUserId
#
#               2 The function calls the following module:
#                   o iron.sh
##############################################################################
InstallReleaseOverNetwork ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# check authorized  userid
if ! CheckAuthorizedUserId
then
     return $FALSE
fi
# call iron.sh
. ${SSCCARS_BIN}/iron.sh
return $TRUE
}
##############################################################################
# Name        : ProcessMenuOption
#
# Decsription : The function processes the option selected.
#
# Notes       : 1 The function may call DisplayMenuOptions ()
##############################################################################
ProcessMenuOption ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
while  true
do
   clear
   DisplayMenuOptions
   case "${OPTION}" in
       5)  clear;
           GLOBAL_EXIT="";
           . ${SSCCARS_BIN}/chkout.sh;
           PROG="ssccars_lib.sh" ;
           if [ "${GLOBAL_EXIT}" = "Y" ]
           then
                 ProcessOverallExit $SEC ;
```

```
            fi ;;
       10)  clear;
            GLOBAL_EXIT="";
            . ${SSCCARS_BIN}/chkin.sh;
            PROG="ssccars_lib.sh" ;
            if [ "${GLOBAL_EXIT}" = "Y" ]
            then
                  ProcessOverallExit $SEC ;
            fi ;;
       15) clear; ViewCheckedOutSourceNames ;;
       20) clear; ViewCheckInCheckOutLog ;;
       25) clear; ViewSourceNameMappingFile ;;
       30) if CancelCheckedOutBooking
           then
              DisplayMessage I "${CANCELLATION_DONE}" ;
           else
              DisplayMessage E "${CANCELLATION_FAILED}" ;
           fi ;;
       35) InitializeSSCCARSLogFile;;
       40) InitializeSourceNameMappingFile ;;
       45) clear;
           BuildRelease "P" ;
           PROG="ssccars_lib.sh" ;;
       50) clear;
           BuildRelease "F" ;
           PROG="ssccars_lib.sh" ;;
       55) clear;
           ViewBuiltReleaseLogFile ;;
       60) clear;
           InitializePatchBuiltDateFile ;;
       65) clear;
           ViewLastPatchBuiltFile ;;
       70) clear;
           InstallReleaseOverNetwork;
           PROG="ssccars_lib.sh";;
       75) clear;
           ViewInstalledReleaseLogFile;;
       99) ProcessOverallExit  $SEC ;;
        *) if [ "${FUNCTION_INTERRUPTED}" = "Y" ]
           then
               FUNCTION_INTERRUPTED=N ;
           else
               DisplayMessage E  "${INVALID_ENTRY}" ;
           fi ;;
   esac
done
}
##############################################################################
# Name        : ViewBuiltReleaseLogFile
#
```

```
# Decsription : Allows the user to view log files for releases
#               that have been built.
#
# Notes       : 1 The log file name has the following form:
#
#                 fr_<$TODAY>_<sequence>.log      (full release)
#
#                 pr_<$TODAY>_<sequence>.log      (patch release)
#
#               2 The function calls the following functions:
#                   o GetReleaseLogFileNameFromLov "B"
#                   o DisplayMessage
################################################################################
ViewBuiltReleaseLogFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
while true
do
  clear
  LOG_FILE_NAME=""
  echo "Enter log file name (l = list of values, q = quit):\c"
  read LOG_FILE_NAME
  case  $LOG_FILE_NAME in
      q|Q) return $TRUE ;;
      l|L) GetReleaseLogFileNameFromLov "B";
           if  [ -z "${LOG_FILE_NAME}" ]
           then
                continue ;
           else
               FULL_LOG_FILE_NAME="${RELEASE_LOG_DIR}/
               ➤  ${LOG_FILE_NAME}";
               break ;
           fi ;;
        "") DisplayMessage E "${INVALID_ENTRY}" ;;
         *) FULL_LOG_FILE_NAME="${RELEASE_LOG_DIR}/${LOG_FILE_NAME}" ;
           if [ ! -f  ${FULL_LOG_FILE_NAME} ]
           then
                DisplayMessage E "${INVALID_LOG_FILE}" ;
           else
                break ;
           fi ;
  esac
done
# copy the log file to a temporary file before viewing it
cp  ${FULL_LOG_FILE_NAME}  ${TEMP_FILE1}
view  ${TEMP_FILE1}
return $TRUE
}
################################################################################
# Name        : ViewInstalledReleaseLogFile
```

```
#
# Decsription : Allow the user to view any installed release log
#               file that is generated by iron.sh ('install
#               release over network').
#
# Notes       : 1 The installed release log file name has the
#                 following form:
#
#                 iron_<$TODAY>_<sequence>.log
#
#               2 The function calls the following functions:
#                    o GetReleaseLogFileNameFromLov "I"
#                    o DisplayMessage
#################################################################################
ViewInstalledReleaseLogFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
while  true
do
  clear
  LOG_FILE_NAME=""
  echo "Enter log file name(l= list of values q to quit):\c"
  read LOG_FILE_NAME
  case  $LOG_FILE_NAME in
    q|Q) return $TRUE  ;;
    l|L) GetReleaseLogFileNameFromLov "I" ;
         if  [  ! -z "${LOG_FILE_NAME}" ]
         then
              FULL_LOG_FILE_NAME="${RELEASE_LOG_DIR}/
              ➤  ${LOG_FILE_NAME}";
              break ;
         fi ;;
     "") DisplayMessage E "${INVALID_ENTRY}" ;;
      *) FULL_LOG_FILE_NAME="${RELEASE_LOG_DIR}/${LOG_FILE_NAME}";
         if [ ! -f  ${FULL_LOG_FILE_NAME} ]
         then
               PROG="ssccars.sh"
               DisplayMessage E "${INVALID_LOG_FILE}" ;
         else
               break ;
         fi ;;
  esac
done
# copy the log file to a temporary file before viewing it
cp  ${FULL_LOG_FILE_NAME}  ${TEMP_FILE1}
view  ${TEMP_FILE1}
return $TRUE
}
#################################################################################
# Name        : InitializePatchBuiltDateFile
```

```
#
# Decsription : Initialize the last patch build date file.
#
# Notes       : 1 The last patch build date file is called
#                   .last_patch_built and it's in $ROOT_RELEASE_DIR
#
#               2 The function calls the following functions:
#                   o CheckAuthorisedUserId
#                   o ValidateLastPatchBuiltDate
#                   o DisplayMessage
################################################################################
InitializePatchBuiltDateFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# seek root user privilege
if ! CheckAuthorisedUserId
then
     return $FALSE
fi
# check for file existence
if [ -f "${LAST_PATCH_BUILT_DATE_FILE}" ]
then
    while true
    do
        clear
        echo "Last Patch Built Date file exists"
        echo "Do you wish to re-initialize? (Y/N or q=quit):\c"
        read REPLY
        case  $REPLY in
          q|Q) return $TRUE  ;;
          n|N) return $FALSE ;;
          Y|y) break ;;
            *) DisplayMessage E "${INVALID_ENTRY}" ;;
        esac
    done
fi
# initialize the file
while true
do
    clear
    echo "Enter a date in format <DDMMYYYYHHMISS>
    ➤  (${LAST_PATCH_BUILT_DATE}):\c"
    read  LAST_PATCH_BUILT_DATE
    case  $LAST_PATCH_BUILT_DATE in
       "") DisplayMessage  E  "${INVALID_ENTRY}" ;;
        *) # validate the date
           if  ValidateLastPatchBuiltDate
           then
               echo "${LAST_PATCH_BUILT_DATE}" >
               ➤  ${LAST_PATCH_BUILT_DATE_FILE} ;
```

```
                DisplayMessage I
                ➤  "${LAST_PATCH_BUILT_DATE_FILE_INITIALIZED}" ;
                return $TRUE ;
            fi ;;
    esac
done
}
#############################################################################
# Name        : ViewLastPatchBuiltFile
#
# Decsription : Allows the user to view the last patch build date
#               file
#
# Notes       : 1 The file is called .last_patch_built and
#                 resides in the $ROOT_RELEASE_DIR directory.
#
#               2 The function calls folllowing function:
#                   o DisplayMessage
#############################################################################
ViewLastPatchBuiltFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# build file name
LAST_PATCH_BUILT_DATE_FILE="${ROOT_RELEASE_DIR}/.last_patch_built"
# check for file existence
if [ ! -f "${LAST_PATCH_BUILT_DATE_FILE}" ]
then
    DisplayMessage E "${NO_LAST_PATCH_BUILT_DATE_FILE}"
    return $FALSE
fi
# copy the file into a temporary file and view it
cp $LAST_PATCH_BUILT_DATE_FILE  ${TEMP_FILE1}
view ${TEMP_FILE1}
return $TRUE
}
#############################################################################
# Name        : main
#
# Decsription : The function defines the variable $SSCCARS_BIN,
#               where all other scripts reside. It checks
#               the library module is in a valid state before
#               trying to load it.
#
# Notes       : 1 The function calls the following functions:
#                   o DefileModuleVariables
#                   o DisplayLocalMessage
#                   o PerformSanityCheck
#                   o ProcessMenuoption
#                   o ProcessOverallexit
#
#               2 The function calls the following module:
```

```
#                    o ssccars_lib.sh
################################################################################
main ()
{
SSCCARS_BIN="/usr/bin" ; export  SSCCARS_BIN
# perform sanity check for library
if [ ! -f ${SSCCARS_BIN}/ssccars_lib.sh ]
then
     DisplayLocalMessage E "${LIBRARY_MISSING}"
     ProcessOverallExit $FEC
fi
if [ ! -x ${SSCCARS_BIN}/ssccars_lib.sh ]
then
     DisplayLocalMessage E "${LIBRARY_NOT_EXECUTABLE}"
     ProcessOverallExit $FEC
fi
# load the library
. ${SSCCARS_BIN}/ssccars_lib.sh
# define local variables
DefineModuleVariables
# perform overall sanity check
if ! PerformSanityCheck
then
    ProcessOverallExit $FEC
fi
ProcessMenuOption
}
# execute main
# invoke main
main
```

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

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

Create and move to a directory

How often do you create a directory and then move into it using your next shell command? If your working methods are like mine, then probably more often than not.

While the Korn shell has the command **mkdir** to make a directory and

cd to move to a directory, it doesn't have an **md** command to make a directory and move into it in one go. No problem, we'll create a simple **md** shell command to do just that:

```
mkdir -p $1 && cd $1
```

mkdir's **-p** parameter creates multilevel subdirectories. The '&&' tells **md** to go to the **cd** command only if the **mkdir** command executed without error.

If you run this command, you'll see that it creates the required directory but leaves you exactly where you were before you executed the **md** command. But didn't I promise that **md** would move you into the directory just created? An explanation: the Korn shell creates a child process to execute your command. In the child process, the current directory *is* changed to the newly created directory, but when the child process terminates and control returns to the parent, the working directory reverts to the parent's working directory. Hence the user sees no change in the current directory.

A solution to this problem is to execute the command as:

```
. md <newDirectory>
```

The dot before a shell command replaces the parent process with the newly created child process. An improvement to this slightly messy solution is to use an alias. Move your **md** command to */usr/local/bin/* and make the alias **md** execute the command as the new process:

```
alias md='. /usr/local/bin/md'
```

AIX Specialist (Switzerland)

Fixing the password database in AIX 4.3.2

In a large system with thousands of users, such as you'd find in a university environment, where users are added and deleted regularly, the password database needs to be kept in good shape.

Whenever entries in the */etc/passwd* and */etc/security/passwd* files are not in the same order, or the */etc/passwd* file or another user database

file has an invalid entry, an error occurs in updating the password database.

When the password database is corrupted, functions related to user attributes (for instance, those used to change a password or group, or add or delete users) cannot be performed. To illustrate this, a sample session (including the system's response), during which a user password was changed, is shown below.

```
$passwd raheem
Changing password for "raheem"
raheem's Old password:
raheem's New password:
Re-enter raheem's new password:
An error occurred updating the password database.
```

The command **pwdck -y ALL** can be run to fix password information on all users in user database files. The **pwdck** command locks both the */etc/passwd* and */etc/security/passwd* files when it updates them. If either of these files is locked by another process, the **pwdck** command waits a few minutes for the files to be unlocked, and terminates if this does not happen within the specified time frame.

If for some reason the **pwdck** command fails repeatedly, then the most likely reason is that the */etc/passwd* file is locked by a runaway process. The command **fuser -u /etc/passwd** lists all processes that are using the */etc/passwd* file. After killing the runaway process, the **pwdck** command should be re-run until it successfully fixes the password database.

A sample output of the **pwdck -y ALL** command being run successfully is shown below.

```
#pwdck -y ALL
The user "g934691" has an invalid password field in /etc/passwd.
The authname "g934691" has no entry in /etc/security/passwd.
Adding "g934691" stanza to /etc/security/passwd.
/etc/security/passwd is not in the same order as /etc/passwd.
Sorting /etc/security/passwd in the same order as /etc/passwd.
The DBM files for /etc/passwd are out of date.
Use the mkpasswd command to update these files.
```

Mohammed Abdul Raheem
Systems Analyst
King Fahd University (Saudi Arabia)

AIX news

IBM has announced new RS/6000 models and some new AIX features. Among the new products are the RS/6000 S80, a 24-way system based on copper technology PowerPC processors. It's claimed to be more than three times faster than an S70A. IBM also announced the RS/6000 HA-S80, a clustered version of the S80.

Targeting the Internet Service Provider markets is the new B50, along with the Intel-based Netfinity 4000R, a cheap 3.5" rack-mounted unit that supports both AIX and Linux. Also announced is a new POWER3 SMP node for the T70 SP system, which also accepts older nodes. The new node has four times the number of processors and memory capacity of its predecessors, 18 times the disk capacity, and 26 times as many I/O adapters.

AIX 4.3.3 gets a new Workload Manager, derived from the S/390. The company said most Linux applications will run on AIX 4.3.3 in the first half of next year through a no-charge open-source download.

For further information, contact your local IBM representative.

* * *

Tripwire Security Systems has released Tripwire Version 2.2, the latest version of its integrity assessment software for Unix, which now supports AIX. The software comprises a number of modules, including ones for damage assessment and recovery, software verification, auditing, and policy compliance. It protects installations from changes to operating system files and can protect users against any 'back door' attacks coded into software.

Out now, prices weren't announced.

For further information contact:
Tripwire Security Systems, 1631 NW Thurman St, Portland, OR 97209, USA
Tel: +1 503 223 0280
Fax: +1 503 223 0182
Web: http://www.tripwiresecurity.com

* * *

Platinum Technology, now a part of CA, has announced the ProVision Adapter for HP OpenView, which provides OpenView IT/ Operations users with single-point integration to ProVision software for managing applications, databases, desktops, networks, and servers. The product is intended to provide improved control of IS resources and service levels.

In addition to AIX (and, obviously, HP-UX), the ProVision adapter for OpenView runs on NT and Solaris. It's out now, with prices starting at US$25,000.

For further information contact:
Platinum Technology Inc, 1815 S Meyers Road, Oakbrook Terrace, IL 60181, USA
Tel: +1 630 620 5000
Fax: +1 630 691 0718
Web: http://www.platinum.com

Platinum Technology (UK) Ltd, Platinum House, North Second Street, Central Milton Keynes MK9 1BZ, UK
Tel: +44 1908 2484000
Fax: +44 1908 248444