50

AIX

December 1999

In this issue

3 An overview of new RS/6000 hardware

9 Automated file copy to servers

13 A colourful aixterm

18 SSCCARS (part 2)

52 AIX news

update

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.

An overview of new RS/6000 hardware

The latest wave of announcements made by IBM's RS/6000 division on September 13 includes a number of new hardware devices, and I reckon there's something in there for just about all *AIX Update*'s readers. This article presents a technical summary of the new products, comparing them with previously available ones.

RS/6000 ENTERPRISE SERVER MODEL S80

The S80 is the latest member of the RS/6000's 'S' family of enterprise servers. It's also the largest RS/6000 SMP server so far, able to scale to 24 processors.

The S80 uses the latest 450 MHz RS64-III 'Pulsar' processors, which are first to incorporate IBM's new technology for copper circuitry on silicon wafers. The Pulsar is a superscalar RISC processor featuring a high-bandwidth, low-latency short pipe, large caches, and a zero cycle 'branch mispredict' penalty. Based on the PowerPC architecture, the Pulsar is the second generation of the 'Start' series microprocessors. The table below summarizes the differences between the new processor and its predecessor, the NorthStar, which is used in the S7A.

Feature	RS64 II	RS64 III
Die size	162 sq mm	140 sq mm
Manufacturing technology	0.35 micron	0.22 micron
Transistors	12.5 million	34 million
L1 cache:		
Instruction	64 KB	128 KB
Data	64 KB	128 KB
L2 directory	Off chip	On chip
L2 cache bandwidth	8.4 GB/s	14.4 GB/s
Power supply	2.5 Volt	1.8 Volt
Maximum power	27 Watts	22 Watts
Operating frequency	262 MHz	450 MHz

More information on the Pulsar is available in an IBM White Paper, *Fifth Generation 64-bit PowerPC-Compatible Commercial Processor Design*, which is available at:

```
http://www.rs6000.ibm.com/resource/technology/pulsar.html
```

Although the clock rate has been increased by 70%, overall system performance has increased by substantially more than this thanks to other improvements. The following table summarizes the differences between the S80 and the S7A.

Feature	S7A	S80
Processor type	4-way RS64 II	6-way RS64 III
Max number of processors	12	24
Processor and I/O paths	3 + 1	8 + 2
Processor and I/O rates	1.4 GB/s	2.4 GB/s
Memory paths	2	4
Memory transfer rates	2.8 GB/s	4.8 GB/s
Max system memory	32 GB	64 GB
System bus bandwidth	11.2 GB/s	43.2 GB/s
SPECweb96 (12-way)	20,200	40,161
Relative OLTP performance (12-way)	113.8	233.3

The maximum relative OLTP performance of a 24-way S80 is 400, which is 3.5 times that of a 12-way S7A. The S80's TPC-C benchmark result, submitted on October 6, is 126,671 tpmC.

The processors of the new system are installed in protected units, called 'books', each of which contains four processors. This enables gradual scalability from four to 24 processors.

In order to upgrade an S7A to an S80, the following procedure should be followed:

1 Disconnect the five cables between the first S00 rack and S7A CEC (this comprises two RIO cables, two SPCN cables, and one JTAG cable).

2 Replace the S7A CEC with a S80 CEC.

3 Reconnect the five cables.

4 Install AIX 4.3.3 (the only AIX revision supported by the S80).

5 Redefine the service processor features.

Thanks to the S series' modular design, no user data is affected by the upgrade.

As mentioned above, the S80 requires AIX 4.3.3, which is tuned to the system's enhanced scalability. The maximum number of threads

supported by AIX 4.3.3 is increased from 128 K to 512 K, while the maximum number of processes increases from 128 K to 170 K. In-kernel locking services have been enhanced, as has the Virtual Memory System to enable it to utilize multiple memory pools. Multiple run queues have been implemented to improve CPU affinity and improve application throughput.

POWER3 SMP HIGH NODE AND T70 SP EXPANSION UNIT

The new RS/6000 SP High Node is a 2 to 8-way SMP server based on 222 MHz POWER3 processors. It's available in two configurations: as an SP node or as a standalone workgroup server.

The machine supports 2, 4, 6, or 8-processor configurations using the 64-bit 222 MHz POWER3 processor with 4 MB of Level 2 cache per processor. The system can be configured with up to 16 GB of RAM and it accommodates up to two internal Ultra SCSI disks with a capacity of either 9.1 GB or 18.2 GB. Five PCI slots are available (one is 32-bit and the other four are 64-bit) as well as an internal Ultra SCSI interface, a 10/100 Mbps Ethernet interface, and a serial port. Connection to the SP switch is performed using an MX2 adapter (feature number 4023) that connects directly to the computer's mezzanine (MX) bus and doesn't take up any of the system's PCI slots.

The system may be connected to up to six of the newly announced SP I/O expansion units. Each unit has eight PCI slots and four hot-pluggable storage bays, which support SCSI as well as SSA disks. The unit occupies one thin node slot in an SP frame.

Two enclosures are available for this system: a tall Model 550 frame for the SP node and a newly announced medium height frame (it's 53.5", or 1.36 m, tall) for the T70 workgroup server. The medium height frame can accommodate one T70 server and up to four I/O expansion units.

Note that, when the system is packaged as a T70 SMP workgroup server, its operation requires the installation of an RS/6000 7043 Model 140, which is used as a control workstation for the server.

The following table compares the new SP node's performance with

that of previously available nodes:

	332 MHz SMP Thin/ Wide	POWER3 SMP Thin/ Wide	POWER3 SMP High	Improvement (asterisk, '*', shows figures used)
SPECint95 (1-way)	14.4	13.2	14.2	
SPECfp95 (1-way)	12.6	30.1	28.2	
SPECint_base_rate95				
1-way		111		
2-way	249	222	254	
4-way	490*		743	
8-way			981*	200%
Specfp_base_rate95				
1-way		366		
2-way	206	468*	505	
4-way	243		1422	
8-way			1863*	398%
RelativeOLTP				
1-way		10.5		
2-way	17.9	21.0	23.0	
4-way	32.8*		43.3	
8-way			81.3*	248%

RS/6000 SERVER MODEL B50

The RS/6000 Model B50 server is designed to meet the needs of ISPs and ASPs for packing servers densely in industry-standard 19" racks. In such environments, it’s common to dedicate servers to specific functions, such as Web server, firewall, proxy server, mail server, etc. The features that make this machine suitable for these tasks are listed below.

- The unit is just 2 AU (3.5") tall and conforms with standard 19" form factor.
- Up to 20 B50s can be installed in an industry-standard 19" rack.
- Up to 16 B50s can be installed in RS/6000 S00 racks.
- The system provides plug-in serviceability and tool-less access to all major system components.
- Support for PowerPC version of Linux.

The B50 is based on an RS/6000 model 43P 150 and has similar features: one 373 MHz PowerPC 604e processor, up to 1 GB of RAM,

two PCI slots, two disk drive bays, an integrated 10/100 Mbps Ethernet adapter, an integrated Ultra SCSI controller, two serial ports, and one parallel port. Both AIX 4.3.2 and AIX 4.3.3 are supported.

The version of Linux is available from *Terra Soft Solutions*, and details can be obtained from:

```
http://www.yellowdoglinux.com
```

IBM EXPANDABLE STORAGE PLUS ('EXP PLUS')

Exp Plus is a storage enclosure that can accommodate up to ten SCSI disk drives. The system is available in tower and rack-mounted configuration.

The enclosure supports 7,200 rpm drives with capacities of 9.1, 18.2, and 36.4 GB and 10,200 rpm drives with capacities of 9.1 and 18.2 GB. Also supported is the Ultra 2 (LVD) interface, which provides transfer rates of up to 80 Mbps. Optional features include redundant power supply and a second host interface. Advanced monitoring features are provided by support for SCSI enclosure services.

TOKEN RING PCI ADAPTER

This adapter replaces the previously available version, which is being withdrawn. The adapter supports data rates of 4 and 16 Mbps in half-duplex or full-duplex mode. Connection type is autodetected at both speeds. Both UTP (RJ45) and STP (DB9) cable connections are supported.

PCI DUAL-CHANNEL ULTRA2 SCSI ADAPTER

This 64-bit adapter supports up to 80 Mbps data rate transfer on each channel. Drives conforming to the Low Voltage Differential (LVD) standard can be located at cable distances of up to 20 metres using the appropriate drivers and receivers. Each of the two SCSI buses can be internal or external.

PCI 3-CHANNEL ULTRA2 SCSI RAID ADAPTER

This adapter offers three 80 Mbps Ultra2 SCSI buses. One of the buses

is internal and the other two are external, and each bus supports up to 15 drives in RAID 0, 1, or 5 configuration. As many as eight logical arrays can be configured per adapter using physical disks connected to various channels. 16 KB, 32 KB, and 64 KB stripe sizes are supported, with the stripe size being global to all drives on the adapter (it cannot be changed without loss of data). Spare 'hot' drives can be defined, enabling automatic replacement of failed physical drives and automatic logical array rebuild. The card contains 32 MB of fast-write cache, implemented as non-volatile RAM, which substantially increases write performance.

RS/6000 MODEL CONVERSION FROM MODEL F50 TO MODEL H70

A 64-bit upgrade option has been announced for the Model F50, converting it to a Model H70. Some internal components of the F50, such as memory DIMMs, disk drives, and PCI cards, will be reused during the conversion. It should be noted that the F50 has nine PCI slots, whereas the H70 has only eight. In addition, the H70 does not support ISA slots, which means that ISA adapters installed in the F50 cannot be moved to the upgraded system. The new H70 chassis will have the same serial number as the replaced F50 system, allowing reuse of existing software licences. Another benefit of reusing the serial number is that the amortization of the system may be continued.

For more details on this upgrade, please check the White Paper *Migrating to the 64-bit RS/6000 H70*, which is located at:

`http://www.rs6000.ibm.com/resource/technology/h70migrate.html.`

IBM P76 17" AND P260 21" MONITORS

The new P-Series monitors provide improved dot pitch (0.24 mm), utilize new improved Trinitron flat CRTs, and conform with the latest safety standards. VESA 85 MHz non-interlaced display modes are supported at resolutions of 640x480, 800x600, 1024x768, and 1024x1280.

Alex Polak
System Engineer
APS (Israel)

Automated file copy to servers

If there are many AIX servers at a site, the management of the servers is always a problem; for instance, it's often necessary to copy a file or files to all of the servers. If network connections are not good enough, the files may not be sent to some branches, which then requires us to re-send the files to the servers later. In some cases, one AIX server could be in the process of being shut down as you're sending it files. Another possibility is that we have to restart the process if the distribution server is shut down before the sending process is successfully completed.

The easiest way to automate this job is to start a process that ensures all files are sent to all branches eventually. This is precisely the task of the scripts in this article: to enable the user to send files easily to a number of servers.

There are three scripts. The first (**rcopydef**) reads the list of servers' host addresses from a file called *Branches_List* and starts to send the files. The script keeps trying to send files to servers that did not successfully receive them until all files are sent to all servers. The second script (**rcopydef.cont**) is run after the distribution server is shut down or the process of the first script is killed. This script is run by the script **rcopydef.cont.restart**, which prepares for the restart of all processes that did not complete successfully. This script must be defined in */etc/inittab*.

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 text.

RCOPYDEF

```
#!/bin/ksh

if [ "$1" = "" -o "$2" = "" ]

then
  print ""
  print " Usage : "
  print ""
  print " nohup rcopydef source target & "
```

```
  print ""
  print " Example :  nohup rcopydef /home/pmkdbp2/mntdir/test.tar.Z
  ➤  /home/pmkdbp2/mntdir &"
  print ""
  exit
fi

UPATH='/home/pmkdbp2/mntdir/'

#get a unique rcopydef number
rcp_num=`cat "$UPATH"rcopydef.cnt`
rcp_num=`expr $rcp_num + 1`
print "$rcp_num" > "$UPATH"rcopydef.cnt

ULOGPATH="$UPATH"rcopydef."$rcp_num".log

if [ -a "$UPATH"notrcped."$rcp_num" ]
then
  rm "$UPATH"notrcped."$rcp_num"
fi

if [ -a "$UPATH"success."$rcp_num" ]
then
  rm "$UPATH"success."$rcp_num"
fi

cp "$UPATH"/Branches_List "$UPATH"rcpthese."$rcp_num"

print "$1" " " "$2" " " "$rcp_num"  > "$ULOGPATH"
print "rcopydef process started..."  >> "$ULOGPATH"
print "rcopydef number is " "$rcp_num" "." >> "$ULOGPATH"

while [ -a "$UPATH"rcpthese."$rcp_num" ]
do
  notrcp_flg=0

  date >> "$ULOGPATH"
  print "The files will be copied to the following branches" >>
  ➤  "$ULOGPATH"
  print "____________________" >> "$ULOGPATH"
  cat rcpthese."$rcp_num" >> "$ULOGPATH"
  print "____________________" >> "$ULOGPATH"

  for branch_name in `cat "$UPATH"rcpthese."$rcp_num"`
  do
    rcp -p "$1" "$branch_name":"$2" 2> "$UPATH"rcopydef.err
    rcp_rc="$?"

    cat "$UPATH"rcopydef.err >> "$UPATH"rcopydef."$rcp_num".err
```

```
    if [ "$rcp_rc" -ne 0 ]
    then
      notrcp_flg=1
      print "$branch_name" >> "$UPATH"notrcped."$rcp_num"
      print "rcp command has failed for ""$branch_name""." >>
      ➤  "$ULOGPATH"
    else
      print "$branch_name" >> "$UPATH"success."$rcp_num"
      print "Files have been rcped to ""$branch_name"" successfully."
      ➤  >> "$ULOGPATH"
    fi
  done

  if [ "$notrcp_flg" -eq 0 ]
  then
    rm "$UPATH"rcpthese."$rcp_num"
    rm "$UPATH"success."$rcp_num"
    break
  else
    mv "$UPATH"notrcped."$rcp_num" "$UPATH"rcpthese."$rcp_num"

    if [ -a "$UPATH"success."$rcp_num" ]
    then
      rm "$UPATH"success."$rcp_num"
    fi
  fi

  print "I will sleep 10 minutes ..." >> "$ULOGPATH"
  sleep 600

  print >> "$ULOGPATH"
done

rm "$UPATH"rcopydef.err

print "rcopydef numbered " "$rcp_num" "are completed successfully..."
➤  >> "$ULOGPATH"
```

RCOPYDEF.CONT

```
#!/bin/ksh

rcp_num="$3"
UPATH='/home/pmkdbp2/mntdir/'
ULOGPATH="$UPATH"rcopydef."$rcp_num".log

if [ -a /home/pmkdbp2/mntdir/notrcped."$rcp_num" ]
then
  rm /home/pmkdbp2/mntdir/notrcped."$rcp_num"
```

```
fi

while [ -a /home/pmkdbp2/mntdir/rcpthese."$rcp_num" ]
do
  notrcp_flg=0

  date >> "$ULOGPATH"

  print "The files will be copied to the following branches" >>
  ➤  "$ULOGPATH"
  print "____________________" >> "$ULOGPATH"
  cat rcpthese."$rcp_num" >> "$ULOGPATH"
  print "____________________" >> "$ULOGPATH"

  for branch_name in `cat /home/pmkdbp2/mntdir/rcpthese."$rcp_num"`
  do
    grep $branch_name success."$rcp_num" > /dev/null 2>&1
    already_rcped="$?"

    if [ "$already_rcped" -ne 0 ]
    then
      rcp -p "$1" "$branch_name"\:"$2" 2>
      ➤  /home/pmkdbp2/mntdir/rcopydef.err
      rcp_rc="$?"
    else
      rcp_rc=0
    fi

    cat rcopydef.err >> /home/pmkdbp2/mntdir/rcopydef."$rcp_num".err

    if [ "$rcp_rc" -ne 0 ]
    then
      notrcp_flg=1
      print "$branch_name" >> /home/pmkdbp2/mntdir/notrcped."$rcp_num"
      print "rcp command has failed for ""$branch_name""." >>
      ➤  "$ULOGPATH"
    else
      if [ "$already_rcped" -ne 0 ]
      then
        print "$branch_name" >> /home/pmkdbp2/mntdir/success."$rcp_num"
        print "Files have been rcped to ""$branch_name"" successfully."
        ➤  >> "$ULOGPATH"
      else
        print "Files have already been rcped to ""$branch_name""
        ➤  successfully." >> "$ULOGPATH"
      fi
    fi
  done

  if [ "$notrcp_flg" -eq 0 ]
  then
```

```
    rm /home/pmkdbp2/mntdir/rcpthese."$rcp_num"
    rm /home/pmkdbp2/mntdir/success."$rcp_num"
    break
  else
    mv /home/pmkdbp2/mntdir/notrcped."$rcp_num"
    ➤  /home/pmkdbp2/mntdir/rcpthese."$rcp_num"

    if [ -a /home/pmkdbp2/mntdir/success."$rcp_num" ]
    then
      rm /home/pmkdbp2/mntdir/success."$rcp_num"
    fi
  fi

  print "10 dk uyuyacagimdir.     " >> "$ULOGPATH"
  sleep 600
done

rm /home/pmkdbp2/mntdir/rcopydef.err

print "rcopydef numbered " "$rcp_num" "are completed successfully..."
➤  >> "$ULOGPATH"
```

RCOPYDEF.CONT.RESTART

```
#!/bin/ksh

cd /u/pmkdbp2/mntdir

for i in `ls rcpthese.*|sed 's/rcpthese.//'`
do
  nohup head -1 rcopydef.$i.log | xargs  rcopydef.cont &
done
```

Abdullah Ongul
DB2 DBA
Pamukbank (Turkey)

A colourful aixterm

While most think of **aixterm** as nothing more than a simple character-based terminal emulator that supports both the high function terminal (HFT) standard and the not-so-new Digital Equipment Corp (DEC)

VT102 standard, it also has its colourful side. Using colour not only makes the display look more interesting, it can also help the information displayed to be understood more easily.

There are several parameters that are used by **aixterm** to control the display colour. These parameters can be set from the command line, in the X Windows resource file, using **aixterm**'s escape sequences, or a combination of these methods.

COMMAND LINE

To set **aixterm**'s background colour to blue, with white text (the foreground colour), and a red cursor, you need to issue the following command at the command line:

```
$ aixterm -fg white -bg blue -cr red
```

Colours can also be specified using their X resource names, which is a somewhat more lengthy procedure.

```
$ aixterm -xrm 'aixterm.foreground:  white' \
          -xrm 'aixterm.background:  blue'  \
          -xrm 'aixterm.cursorColor: red'
```

The **-xrm** option is standard with every X command, and it supplies a value to the X resource manager (**xrm**). Each X application consists of a hierarchy of named objects known as 'widgets'. Each widget has a set of resources that can be set both internally from the program and from outside the program. Once you understand its rules, you can tailor each X application to your own needs.

X WINDOWS' RESOURCE .XDEFAULTS FILE

aixterm's resources can also be set in the X Windows resource file *.Xdefaults*. To set the same colours as in the preceding example, add the following lines to the *.Xdefaults* file in your home directory:

```
aixterm.foreground:  white
aixterm.background:  blue
aixterm.cursorColor: red
```

Make sure that you leave no redundant spaces at the end of the line. Your system can find the colour 'red' but won't find the colour 'red '.

Unfortunately, it won't complain if it does not find the desired colour. If you are using the Common Desktop Environment (CDE), there is a button called 'Reload Resources' in the Application Manager under *Desktop_Tools*. After you edit your *.Xdefaults* file, you need to run this application to ensure that your X server has the new settings. If you're not using CDE, then you need to use the following command to reload resources:

```
xrdb -load .Xdefaults
```

Now all your **aixterm** windows have white characters, a blue background, and a red cursor. Though this may be useful in its own right, you may wish to have different colours on your various **aixterm** windows. For example, one window may display the local system, while another may be connected to a remote system. Using **aixterm**'s **-name** parameter has two effects: the 'name' is displayed in the window's title bar and is also used when reading resources. First change your *.Xdefaults* to:

```
local.foreground:  white
local.background:  blue
remote.foreground: yellow
remote.background: darkblue
aixterm.cursorColor: red
```

Then reread these resources with:

```
$ xrdb -load .Xdefaults
```

followed by the two **aixterm** commands below.

```
$ aixterm -name local &
$ aixterm -name remote &
```

You can probably tell what effect the above commands have. Notice that both windows have cursors of the same colour as their text. This is because we did not replace the name of the *aixterm.cursorColor* resource.

AIXTERM'S ESCAPE SEQUENCES

aixterm's datastream support allows you to set foreground and background colours using escape sequences. These colour codes are defined by the ISO 6427 standard, shown below.

```
ISO 6427 colour codes
Foreground colours   Background colours   Other codes

30 black             40 black            0 restore default
31 red               41 red              1 brighter colour
32 green             42 green            4 underline
33 yellow            43 yellow           5 blink
34 blue              44 blue
35 purple            45 purple
36 cyan              46 cyan
37 white             47 white
```

The colour code must be preceded by the two characters <ESC> and '[' and followed by the character 'm'. Try:

```
$ echo "\033[32m"
```

This escape sequence results in a green foreground. Note that '\033' is the octal code for the <ESC> character.

COLOURIZED FILE LISTINGS

Unlike GNU's **ls** version, AIX's **ls** does not have a **--color** parameter. This means that we must use **awk** scripting to produce a colourful listing. Create a file named *ls.awk* with the following contents. Note that the characters:

```
"^["
```

are the escape sequence. If you're using the **vi** editor, then the escape sequence can be input using *Ctrl-v* followed by *Ctrl-[*.

LS.AWK

```
function normal() {
    printf("^[[0m")
}
function foregroundRed() {
    printf("^[[31m")
}
function foregroundGreen() {
    printf("^[[32m")
}
function foregroundBlue() {
    printf("^[[34m")
}
function foregroundPurple() {
```

```
    printf("^[[35m")
}
function foregroundCyan() {
    printf("^[[36m")
}
function backgroundRed() {
    printf("^[[41m")
}

BEGIN {
    firstline=1
}

{
    if ( firstline == 1 ) {
        print $0
        firstline = 0
    } else {
        backgroundRed()
        printf("%s", $1)
        normal()
        foregroundBlue()
        printf(" %3d", $2)
        normal()
        foregroundGreen()
        printf(" %-8s %-8s", $3, $4)
        normal()
        foregroundPurple()
        printf(" %7d", $5)
        normal()
        foregroundCyan()
        printf(" %3s %2s %5s ", $6, $7, $8)
        normal()
        foregroundRed()
        printf("%-24s\n", $9)
        normal()
    }
}
```

Now pipe your listing through this **awk** script using:

```
ls -al | awk -f ls.awk
```

This may not be as simple as GNU's **ls**, and you may find the results somewhat gaudy, but at least we've put some colour into **ls**'s output.

SETTING BANNERS

While setting **aixterm**'s banner doesn't have much to do with colour

I think it's appropriate enough to mention it. Again, **aixterm**'s data stream supports this with the escape sequence:

```
ESC ] 0 ; text \007
```

Try:

```
$ echo "^[]0;aixterm's new title\007"
```

Notice that this also becomes the icon's name.

AIXTERM, XTERM, AND DTTERM

These colourful examples are valid not only for AIX's **aixterm**, but also for Unix's generic **xterm** and CDE's **dtterm**.

Werner Klauser
Klauser Informatik (Switzerland)

SSCCARS (part 2)

This month's instalment continues this series of articles on utilities that comprise a source code management system.

SSCCARS_LIB.SH

ssccars_lib.sh is the library module that contains all the common functions and initializes all the global messages and variables. This library is 'included' in the main module **ssccars.sh**. Variables defined here are exported for access by the child process. As all the module calls are initiated in the current shell using 'dot notation', their variables are set in different modules.

SSCCARS_LIB.SH

```
#! /bin/ksh
##############################################################################
# Name     : ssccars_lib.sh
#
```



```
# Overview : The script contains all the global variables and
#            common functions used by ssccars.
#
# Notes    1 The library contains following functions:
#            o DisplayMessage
#            o MoveCursor
#            o GetSourceName
#            o GetSourceNameFromLov
#            o ExtractSourceName
#            o ProcessFileExtension
#            o GetFileExtension
#            o CheckFileExtension
#            o GetDirectoryName
#            o CheckLock
#            o FreeLock
#            o LockSource
#            o CopySource
#            o SmartDatabaseStatus
#            o UpdateSmart
#            o ExecuteSuperUser
#            o GetLatestVersion
#            o SeekConfirmation
#            o CheckRootUserId
#            o CheckAuthorisedUserId
#            o GetReleaseLogFileNameFromLov
#            o LeapYear
#            o ValidateLastPatchBuiltDate
#            o FormatUnderscores
#            o PerformSanityCheck
##############################################################################
# global variables
# exit flag
GLOBAL_EXIT=""  ; export GLOBAL_EXIT
OPTION=""       ; export OPTION
# interrupt facility
FUNCTION_INTERRUPTED=""        # interrupt flag
FUNCTION_ABORTED=""            # user quitting the program flag
# escape sequences
ESC="\0033["   ; export ESC
RVON=_[7m      ; export RVON            # reverse 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
# menu titles
SSCCARS="${RVON}SERVER SOURCE CODE CONTROL AND RELEASE SYSTEM${RVOFF}"
CHKIN="${RVON}SOURCE CHECKIN UTILITY${RVOFF}"
CHKOUT="${RVON}SOURCE CHECKOUT UTILITY${RVOFF}"
# global message used in SeekConfirmation
MESSAGE=""    ; export MESSAGE
```

```
DUMMY=""      ; export DUMMY
# define ssccars bin directory
SSCCARS_BIN="/usr/bin"              ; export SSCCARS_BIN
# define source directories
SSCCARS_DIR="/home/ecatmgr/ssccars" ; export SSCCARSS_DIR
SOURCE_DIR=${SSCCARS_DIR}/source    ; export SOURCE_DIR
TRIG_DIR=${SOURCE_DIR}/trig         ; export TRIG_DIR
PROC_DIR=${SOURCE_DIR}/proc         ; export PROC_DIR
VIEW_DIR=${SOURCE_DIR}/view         ; export VIEW_DIR
SH_DIR=${SOURCE_DIR}/sh             ; export SH_DIR
PC_DIR=${SOURCE_DIR}/pc             ; export PC_DIR
C_DIR=${SOURCE_DIR}/c               ; export C_DIR
AWK_DIR=${SOURCE_DIR}/awk           ; export AWK_DIR
SQL_DIR=${SOURCE_DIR}/sql           ; export SQL_DIR
LOG_DIR=${SSCCARS_DIR}/log          ; export LOG_DIR
LOCK_DIR=${SSCCARS_DIR}/lock        ; export LOCK_DIR
TARGET_DIR=                         ; export TARGET_DIR
CUR_DIR=                            ; export CUR_DIR
TEMP_DIR="${SSCCARS_DIR}/temp"      ; export TEMP_DIR
LOG_FILE="ssccars.log"              ; export LOG_FILE
LOV_FILE="ssccars.dat"              ; export LOV_FILE
DATA_FILE="ssccars.dat"             ; export DATA_FILE
# restricted directory list
RESTRICTED_DIR_LIST="${SSCCARS_DIR} \
                     ${SOURCE_DIR}  \
                     ${TRIG_DIR}    \
                     ${PROC_DIR}    \
                     ${VIEW_DIR}    \
                     ${SH_DIR}      \
                     ${PC_DIR}      \
                     ${AWK_DIR}     \
                     ${C_DIR}       \
                     ${SQL_DIR}     \
                     ${LOCK_DIR}    \
                     ${LOG_DIR}"
export RESTRICTED_DIR_LIST
# define variables for log
LOG_DAY=`date +%d/%m/%y`                    ; export LOG_DAY
LOG_TIME=`date +%H:%M:%S`                   ; export LOG_TIME
LOG_MESSAGE=""                              ; export LOG_MESSAGE
USERID=`id | tr "()" "::" | cut -d':' -f2`  ; export USERID
# define all file extensions allowed
FILE_EXTS="sql trig proc view pc c sh awk"  ; export FILE_EXTS
# define release-related variables
ROOT_RELEASE_DIR="${SSCCARS_DIR}/release"   ; export ROOT_RELEASE_DIR
RELEASE_LOG_DIR="${ROOT_RELEASE_DIR}/log"   ; export RELEASE_LOG_DIR
BUILT_RELEASE_LOG_DIR="${RELEASE_LOG_DIR}"  ; export
➤  BUILT_RELEASE_LOG_DIR
RELEASED_SOURCE_LIST="${TEMP_DIR}/br_source_list_$$.tmp"
export RELEASED_SOURCE_LIST
RELEASE_DESC=""                             ; export RELEASE_DESC
```

```
LOG_FILE_NAME=""                                ; export LOG_FILE_NAME
LAST_PATCH_BUILT_DATE=""
LAST_PATCH_BUILT_DATE_FILE="${ROOT_RELEASE_DIR}/.last_patch_built"
export LAST_PATCH_BUILT_DATE
export LAST_PATCH_BUILT_DATE_FILE
UNDERSCORE=""   ; export UNDERSCORE
HEADER=""       ; export HEADER
# error handling variables
PROG="ssccars_lib.sh"          ; export PROG
INFO="${RVON}\${PROG}:INFO:"   ; export INFO
WM="$PROG WARNING:"            ; export WM
ERROR="${RVON}\${PROG}:ERROR:" ; export ERROR
DM="$PROG DEBUG:"              ; export DM
# file handling variables
SOURCE_NAME=""
SOURCE_NAME_WITHOUT_VERSION=""
SOURCE_EXT=""
SOURCE_NAME_WITHOUT_EXT=""
REQ_DIR=""
TARGET_DIR=""
REQ_SOURCE=""
REQ_FILE=""
LOCKED_FILE=""
ACTION=""
# export variables
export SOURCE_NAME
export SOURCE_NAME_WITHOUT_VERSION
export SOURCE_EXT
export SOURCE_NAME_WITHOUT_EXT
export REQ_DIR
export REQ_SOURCE
export REQ_FILE
export LOCKED_FILE
export ACTION
DEFAULT_SLEEP_DURATION=3
SLEEP_DURATION=${DEFAULT_SLEEP_DURATION}
export DEFAULT_SLEEP_DURATION
export SLEEP_DURATION
# define return code
RC=""   ; export RC
TRUE=0  ; export TRUE
FALSE=1 ; export FALSE
SEC=0   ; export SEC   # success exit code
FEC=1   ; export FEC   # failure exit code
# define an authorized user for menu option '3'
AUTHORISED_USER=ecatmgr ; export AUTHORISED_ROOT
# 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
```

```
SIGTSTP=18  ; export SIGTSTP    # ctrl-z
# define temporary files
TEMP_FILE1="${TEMP_DIR}/ssccars_lib_1_$$.tmp" ; export TEMP_FILE1
TEMP_FILE2="${TEMP_DIR}/ssccars_lib_2_$$.tmp" ; export TEMP_FILE2
DIR_MISSING="Directory \${DIR_NAME} is missing${RVOFF}"
export DIR_MISSING
DIR_NOT_WRITABLE="Directory \${DIR_NAME} is not writable by
➤  user${RVOFF}"
export DIR_NOT_WRITABLE
FILE_MISSING="File \${FILE_NAME} is missing${RVOFF}"
export FILE_MISSING
FILE_NOT_EXECUTABLE="File \${FILE_NAME} is not executable by
➤  user${RVOFF}"
export FILE_NOT_EXECUTABLE
NOT_YET_IMPLEMENTED="Function not yet implemented${RVOFF}"
NO_LOCK_FILE="No lock file found${RVOFF}"
export NO_LOCK_FILE
NO_LOG_FILE="No log file found${RVOFF}"
export NO_LOG_FILE
NO_DATA_FILE="No data file found${RVOFF}"
export NO_DATA_FILE
NO_RELEASE_LOG="Release log \${FILE_NAME} does not exist${RVOFF}"
export NO_RELEASE_LOG
LAST_PATCH_BUILT_DATE_FILE_INITIALIZED="Successfully initialized
➤  file \${LAST_PATCH_BUILT_DATE_FILE}${RVOFF}"
NO_LAST_PATCH_BUILT_DATE_FILE="File \${LAST_PATCH_BUILT_DATE_FILE}
➤  not found${RVOFF}"
export LAST_PATCH_BUILT_DATE_FILE_INITIALIZED
export NO_LAST_PATCH_BUILT_DATE_FILE
WORKING="Working... do not acknowledge this message${RVOFF}"
FUNCTION_NOT_IMPLEMENTED="Function not yet implemented${RVOFF}"
LOG_INITIALIZED="Successfully initialized the log file${RVOFF}"
MAP_FILE_INITIALIZED="Successfully initialized the source name
➤   mapping file${RVOFF}"
NOT_AUTHORISED_USER="Must be \${AUTHORISED_USER} to perform
➤  this task${RVOFF}"
NOT_ROOT_USER="Must be root to perform this task${RVOFF}"
NOT_INTERRUPTABLE="This process cannot be interrupted${RVOFF}"
FUNCTION_ABORTING="Aborting process${RVOFF}"
SOURCE_ERROR="Source file \${TARGET_FILE} does not
➤  exist${RVOFF}"
RESTRICTED_DIR="Not allowed to copy source in directory \${CUR_DIR}
➤  ${RVOFF}"
DIR_NOT_EXIST="Directory \${DIR_NAME} does not exist${RVOFF}"
NOT_AUTHORISED="This option is only available to
➤  \${AUTHORISED_USER}${RVOFF}"
INVALID_EXT="Invalid extension for the source file${RVOFF}"
INVALID_ENTRY="Invalid entry${RVOFF}"
INVALID_FILE_EXT="\${SOURCE_EXT} is an invalid file extension${RVOFF}"
NO_FILE_EXT="\${SOURCE_NAME} is an invalid file name\; file
➤  extension missing${RVOFF}"
```

```
INVALID_DIR="Directory, \${DIR_NAME} does not exist${RVOFF}"
DIR_NOT_WRITEABLE="Directory \${DIR_NAME} is not writeable by
➤  user${RVOFF}"
SOURCE_EXISTS="File \${FILE_NAME} exists in directory
➤  \${DIR_NAME}${RVOFF}"
COPY_FAILED="Failed to copy file \${REQ_FILE} to
➤  \${TARGET_FILE}${RVOFF}"
SOURCE_COPIED="Successfully copied the source${RVOFF}"
NO_SOURCE_FOUND="Source file \${FILE_NAME} does not exist in directory
➤  \${DIR_NAME}${RVOFF}"
SOURCE_ALREADY_LOCKED="Source file \${TARGET_SOURCE} is checked out
➤  for modification${RVOFF}"
SOURCE_NOT_LOCKED="Failed to lock the source \${SOURCE_NAME}${RVOFF}"
LOG_NOT_WRITTEN="Failed to write information to log file${RVOFF}"
READ_LATEST_CHECKOUT_FAILED="Failed to check out latest read-only
➤  copy of \${SOURCE_NAME}${RVOFF}"
READ_LATEST_CHECKOUT_DONE="Successfully checked out latest read-
➤  only copy of \${SOURCE_NAME}${RVOFF}"
READ_SPECIFIC_CHECKOUT_FAILED="Failed to check out specific read-
➤  only copy of \${REQ_SOURCE}${RVOFF}"
READ_SPECIFIC_CHECKOUT_DONE="Successfully checked out specific read-
➤  only copy of \${REQ_SOURCE}${RVOFF}"
UPDATE_CHECKOUT_FAILED="Failed to check out updateable copy of
➤  \${SOURCE_NAME}${RVOFF}"
UPDATE_CHECKOUT_DONE="Successfully checked out updateable copy of
➤  \${TARGET_SOURCE}${RVOFF}"
CANCELLATION_FAILED="Failed to cancel the booking${RVOFF}"
CANCELLATION_DONE="Successfully cancelled booking for
➤  \${SOURCE_NAME}${RVOFF}"
LOCK_NOT_REMOVED="Failed to remove lockfile \${TARGET_SOURCE} for
➤  cancellation${RVOFF}"
NOT_NUMERIC="Input must be numeric${RVOFF}"
DB_NOT_OK="SMART database is not accessible${RVOFF}"
# export variables
export WORKING
export SOURCE_ERROR
export RESTRICTED_DIR
export DIR_NOT_EXIST
export NOT_AUTHORISED
export NOT_AUTHORISED_USER
export NOT_ROOT_USER
export INVALID_EXT
export INVALID_ENTRY
export INVALID_FILE_EXT
export NO_FILE_EXT
export SOURCE_COPIED
export INVALID_DIR
export DIR_NOT_WRITEABLE
export SOURCE_EXISTS
export COPY_FAILED
export SOURCE_COPIED
```

```
export NO_SOURCE_FOUND
export SOURCE_ALREADY_LOCKED
export SOURCE_NOT_LOCKED
export LOG_NOT_WRITTEN
export READ_LATEST_CHECKOUT_FAILED
export READ_LATEST_CHECKOUT_DONE
export READ_SPECIFIC_CHECKOUT_FAILED
export READ_SPECIFIC_CHECKOUT_DONE
export UPDATE_CHECKOUT_FAILED
export UPDATE_CHECKOUT_DONE
export CANCELLATION_FAILED
export CANCELLATION_DONE
export NO_LOCK_FILE
export LOCK_NOT_REMOVED
export NOT_NUMERIC
# message for chkin.sh
CHECKED_IN="Successfully checked in the source \${SOURCE_NAME}${RVOFF}"
NOT_CHECKED_IN="Failed to check in the source \${SOURCE_NAME}${RVOFF}"
# message for br.sh (build release)
EDIT_RELEASE_DOC="Edit the release document as required${RVOFF}"
REL_DIR_EXISTS="Release directory \${RELEASE_DIR} exists${RVOFF}"
REL_DIR_CREATED="Successfully created release directory
➤  \${RELEASE_DIR}${RVOFF}"
PROCESSING_DIR="Processing source directory \${DIR_NAME}${RVOFF}"
OS_ERROR="\${ERROR_MSG}${RVOFF}"
RELEASE_FAILED="Failed to make the release${RVOFF}"
RELEASE_SUCCEEDED="Successfully built the release${RVOFF}"
COPYING_FILE="Copying file \$REQ_FILE \ - do not acknowledge
➤  message${RVOFF}"
COPIED_FILE="Copied the file \$REQ_FILE \ - do not acknowledge
➤  message${RVOFF}"
FILE_NOT_COPIED="Failed to copy the file${RVOFF}"
INVALID_LOG_FILE="Log file \${RELEASE_LOG_FILE} does not exist${RVOFF}"
PATCH_BUILT_DATE_NOT_NUMERIC="Patch build date must be numeric${RVOFF}"
INVALID_DATE="Invalid patch build date${RVOFF}"
INVALID_MONTH_LEN="Month for patch build date must be two
➤  digit${RVOFF}"
INVALID_DAY_LEN="Day for patch build date must be two digit${RVOFF}"
INVALID_YEAR_LEN="Year for patch build date must be two digit${RVOFF}"
INVALID_HOUR_LEN="Hours for patch build date must be two digit${RVOFF}"
INVALID_MIN_LEN="Minutes for patch build date must be two
➤  digit${RVOFF}"
INVALID_SEC_LEN="Seconds for patch build date must be two
➤  digit${RVOFF}"
INVALID_DAY="Day for patch build date is not valid${RVOFF}"
INVALID_MONTH="Month for patch build date is not valid${RVOFF}"
INVALID_YEAR="Year for patch build date is not valid${RVOFF}"
INVALID_HOUR="Hour for patch build date is not valid${RVOFF}"
INVALID_MINUTES="Minutes for patch build date is not valid${RVOFF}"
INVALID_SECONDS="Seconds for patch build date is not valid${RVOFF}"
export EDIT_RELEASE_DOC
```

```
export REL_DIR_EXISTS
export REL_DIR_CREATED
export PROCESSING_DIR
export OS_ERROR
export RELEASE_FAILED
export RELEASE_SUCCEEDED
export COPYING_FILE
export COPIED_FILE
export FILE_NOT_COPIED
export INVALID_LOG_FILE
export PATCH_BUILT_DATE_NOT_NUMERIC
export INVALID_MONTH_LEN
export INVALID_DAY_LEN
export INVALID_YEAR_LEN
export INVALID_HOUR_LEN
export INVALID_MIN_LEN
export INVALID_SEC_LEN
export INVALID_DAY
export INVALID_MONTH
export INVALID_HOUR
export INVALID_MINUTES
export INVALID_SECONDS
# common message
SOURCE_NOT_CHECKED_OUT="Source \${SOURCE_NAME} is not checked out
➤  for modification${RVOFF}"
export SOURCE_NOT_CHECKED_OUT
ORACLE_ERROR="\${ERROR_MSG}${RVOFF}"; export ORACLE_ERROR
EXECSU_ERROR="\${ERROR_MSG}${RVOFF}"; export OEXECSU_ERROR
# smart database interface
SQL_COMMAND_FILE="${TEMP_DIR}/br_$$.sql" ; export SQL_COMMAND_FILE
SMART_USER="ecatmgr"
RELEASE_INITIATOR="AZ"
SMART_UPDATED="Successfully updated SMART database${RVOFF}"
SMART_NOT_UPDATED="Failed to update SMART database${RVOFF}"
# common functions
################################################################################
# Name     : HandleInterrupt
#
# Overview : Displays a message and returns $TRUE
################################################################################
HandleInterrupt ()
{
FUNCTION_INTERRUPTED=Y
DisplayMessage I "${NOT_INTERRUPTABLE}"
return $TRUE
}
################################################################################
# Name     : MoveCursor
#
# Input    : Y and X coordinates
#
```

```
# Overview : Moves the cursor to the required location (Y, X).
################################################################################
MoveCursor ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
YCOR=$1
XCOR=$2
echo "${ESC}${YCOR};${XCOR}H"
}
################################################################################
# Name     : DisplayMessage
#
# Overview : Displays message
#
# Input    : 1 Message type (E = Error, I = Information).
#            2 Error code defined in DefineMessages ().
#            3 Y or N flag to indicate whether the message is to be
#              acknowledged.
#
# Notes    1 If the third argument is null, the default is Y.
################################################################################
DisplayMessage ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
MESSAGE_TYPE="$1"
MESSAGE_TEXT=`eval echo $2`
MSG_ACK_FLAG="$3"
if [ "${MSG_ACK_FLAG}" = "" ]
then
    MSG_ACK_FLAG="Y"
fi
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
# establish whether the user has acknowledged the message
if [ "${MSG_ACK_FLAG}" = "Y" ]
then
    # wait for user input
    read DUMMY
else
    :
fi
return ${TRUE}
}
################################################################################
```

```
# Name     : GetSourceName
#
# Overview : The function gets a source name from the user.
#
# Input    : Mode  CI  = check in existing source
#                  CIN = check in new source
#                  CO  = check out source
#
# Notes    1 The message displayed depends on the mode.
#          2 The function calls GetSourceFileNameFromLov
################################################################################
GetSourceName ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_MODE="$1"
SOURCE_NAME=""
while true
do
    clear
    if [ "${P_MODE}"  = "CI" ]
    then
        echo "Enter short source file name with version number
        ➤  and file extension"
        echo "(q to quit):\c"
    elif [ "${P_MODE}"  = "CO" ]
    then
        echo "Enter short source file name with extension
        ➤  (l=list of values)"
        echo "(q to quit):\c"
    elif [ "${P_MODE}"  = "CIN" ]
    then
        echo "Enter short source file name, including extension,"
        echo "of up to 20 characters (q to quit):\c"
    fi
    read SOURCE_NAME
    case $SOURCE_NAME in
        "") if [ "${FUNCTION_INTERRUPTED}" = "Y" ]
            then
                 FUNCTION_INTERRUPTED=N ;
            else
                DisplayMessage E "${INVALID_ENTRY}" ;
            fi ;;
       l|L) if [ "${P_MODE}" != "CO" ]
            then
                DisplayMessage E "${INVALID_ENTRY}" ;
                continue;
            fi;
            GetSourceNameFromLov ;
            if [ -z "${SOURCE_NAME}" ]
            then
                   : ;
```

```
            else
                 clear ;
                 break ;
            fi ;;
       q|Q) FUNCTION_ABORTED="Y" ; return $FALSE ;;
         *) SOURCE_NAME="`echo $SOURCE_NAME | cut -c1-20`" ;
            break ;;
    esac
done
}
################################################################################
# Name     : GetSourceNameFromLov
#
# Overview : The function displays a list of souce file names from
#            which one is selected.
#
# Notes    1 $SOURCE_NAME is not checked in this function.
################################################################################
GetSourceNameFromLov ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# prepare list
echo "                  List of source files" > ${TEMP_FILE1}
echo "                  ====================" >> ${TEMP_FILE1}
echo "\t Select source by deleteing line containing required
➤  source name
      \t and saving file in usual manner\n" >> ${TEMP_FILE1}
cat ${TEMP_FILE1}  > ${TEMP_FILE2}
cat $LOG_DIR/${LOV_FILE} >> ${TEMP_FILE1}
cat $LOG_DIR/${LOV_FILE} >> ${TEMP_FILE2}
vi  ${TEMP_FILE1}
LINE=`diff ${TEMP_FILE1} ${TEMP_FILE2}`
SOURCE_NAME=`echo $LINE | awk {'print $3'}`
rm -f ${TEMP_FILE1}
rm -f ${TEMP_FILE2}
}
################################################################################
# Name     : GetReleaseLogFileNameFromLov
#
# Overview : Displays a list of all release-related log files
#            for the user to select one.
#
# Input    : Type of Log file:
#            B = log file for built release
#            I = log file for installed release
#
# Notes    1 The function populates the global variable
#            $LOG_FILE_NAME.
################################################################################
GetReleaseLogFileNameFromLov ()
{
```

```
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
LOG_FILE_TYPE="$1"
# get a list of all log file
CURDIR=`pwd`
# switch to release log dir
cd ${RELEASE_LOG_DIR}
# initialize the LOV file
if [ "${LOG_FILE_TYPE}" = "B" ]
then
    echo "   List of values for log files for built release" >
    ➤  ${TEMP_FILE1}
    echo "   ------------------------------------------" >>
    ➤  ${TEMP_FILE1}
else
    echo "   List of values for log files for installed release" >
    ➤  ${TEMP_FILE1}
    echo  "   ------------------------------------------------">>
    ➤  ${TEMP_FILE1}
fi
echo "     Select file name by deleting corresponding line" >>
➤  ${TEMP_FILE1}
echo  "         and saving the file \n\n" >> ${TEMP_FILE1}
#  select appropriate log files
if [  "${LOG_FILE_TYPE}" = "B" ]
then
    ls -lt fr_* 2>/dev/null | awk {'print $9'} >> ${TEMP_FILE1}
    ls -lt pr_* 2>/dev/null | awk {'print $9'} >> ${TEMP_FILE1}
else
    ls -lt iron_* | awk {'print $9'} >> ${TEMP_FILE1}
fi
cp ${TEMP_FILE1} ${TEMP_FILE2}
# allow user to select a file from the list
vi ${TEMP_FILE2}
# pick the selected log file name
LOG_FILE_NAME=`diff ${TEMP_FILE1} ${TEMP_FILE2} | tail |
➤  awk {'print $2'}`
LOG_FILE_NAME=`echo $LOG_FILE_NAME | sed s/'\n'//`
cd $CURDIR
return $TRUE
}
##############################################################################
# Name     : GetDirectoryName
#
# Overview : The function gets a directory name (where the
#            source is to be checked in from or checked out to).
#
# Input    : Mode  CI = check in
#                  CO = check out
#
# Notes    1 If the directory name is null, the default is the
#            current directory.
```

```
#############################################################################
GetDirectoryName ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_MODE="$1"
while  true
do
    if [ "${P_MODE}"  = "CI" ]
    then
        clear
        echo "Enter a directory name (the location where the source
        ➤  is copied from)"
        echo "Press return for current directory (q to quit):\c"
    elif [ "${P_MODE}"  = "CO" ]
        then
        clear
        echo "Enter a directory name (the location where the source
        ➤  is copied to)"
        echo "Press return for current directory (q to quit):\c"
    fi
    read DIR
    case $DIR in
         "") if [ "${FUNCTION_INTERRUPTED}" = "Y" ]
             then
                 FUNCTION_INTERRUPTED=N ;
             else
                 DIR="." ; # define as current directory
                 break ;
             fi ;;
        q|Q) FUNCTION_ABORTED=Y ; return $FALSE ;;
          *) break ;;
    esac
done
# check the directory exists
if [ -d $DIR ]
then
     :
else
     DIR_NAME="${DIR}"
     DisplayMessage E "${DIR_NOT_EXIST}"
     return $FALSE
fi
# check the permission
if [ -w $DIR ]
then
     :
else
     DIR_NAME="${DIR}"
     DisplayMessage E "${DIR_NOT_WRITEABLE}"
     return $FALSE
```

```
fi
# assign $DIR to appropriate variable
if [ "${P_MODE}" = "CI" ]
then
    REQ_DIR="${DIR}"
elif [ "${P_MODE}" = "CO" ]
then
    TARGET_DIR="${DIR}"
fi
return $TRUE
}
################################################################################
# Name     : CopySource
#
# Overview : Copies source from one location to another.
#
# Input    : Source File
#            Target File
################################################################################
CopySource ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_SOURCE_FILE="$1"
P_TARGET_FILE="$2"
# copy the source
cp $P_SOURCE_FILE $P_TARGET_FILE 2> /dev/null
RC=$?
if [ $RC -ne 0 ]
then
    DisplayMessage E "${COPY_FAILED}"
    return $FALSE
else
    DisplayMessage I "${SOURCE_COPIED}"
    return $TRUE
fi
}
################################################################################
# Name     : FreeLock
#
# Overview : The function frees a lock on a source.
#
# Input    : Source name
#
# Returns  : $TRUE or $FALSE
################################################################################
FreeLock ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_SOURCE_NAME="$1"
ACTION="RM"
```

```
${SSCCARS_BIN}/execsu ${ACTION} ${P_SOURCE_NAME} ${LOCK_DIR} >
‰  ${TEMP_FILE1} 2>&1
RC=$?
if [ $RC -ne 0 ]
then
    ERROR_MSG=`cat ${TEMP_FILE} |  head -1`
    DisplayMessage E "${EXECSU_ERROR}"
    return $FALSE
else
    return $TRUE
fi
}
##############################################################################
# Name     : CheckLock
#
# Overview : Checks whether a source is already checked out for
#            modification
#
# Input    : Source name
#
# Returns  : TRUE if the file is already locked; FALSE otherwise
##############################################################################
CheckLock ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_SOURCE_NAME="$1"
LOCKED_FILE="${LOCK_DIR}/${P_SOURCE_NAME}"
if [ -f $LOCKED_FILE ]
then
    # file is already locked
    return $TRUE
else
    return $FALSE
fi
}
##############################################################################
# Name     : ProcessFileExtension
#
# Overview : Checks that the file extension for the required
#            source file is valid.
#
# Input    : Call Type (I = Internal, E = External)
#
# Returns  : $TRUE
#            $FALSE
#
# Notes    1 The function calls the following functions:
#            o GetFileExtension
#            o CheckFileExtension
##############################################################################
```

```
ProcessFileExtension ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_CALL_TYPE="$1"
if ! GetFileExtension "${P_CALL_TYPE}"
then
    return $FALSE
fi
if ! CheckFileExtension "${P_CALL_TYPE}"
then
    return $FALSE
else
    return $TRUE
fi
}
##############################################################################
# Name     : GetFileExtension
#
# Overview : The function extracts the file extension from the
#            required source name held in $SOURCE_NAME. In the
#            process, it also extracts the source name without
#            the extension.
#
# Input    : Call type (I = Internal E = External)
#
# Returns  : $TRUE or $FALSE
#
# Notes    1 The function allows the file to have more than
#            one period in file name and assumes the last
#            period as the separator of file extension.
#
#          2 The function uses the global variable $SOURCE_NAME.
#
#          3 If the call type is I, do not display message.
##############################################################################
GetFileExtension ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# assign parameter
P_CALL_TYPE="$1"
SOURCE_EXT=""
SOURCE_NAME_WITHOUT_EXT=""
SOURCE_EXT=`echo $SOURCE_NAME | sed 's/^.*\.//'`
if [ "${SOURCE_EXT}" = "${SOURCE_NAME}" ]
then
    # source file name has no extension
    if ! ([ "${P_CALL_TYPE}" = "I" ])
    then
        DisplayMessage E "${NO_FILE_EXT}"
    fi
```

```
    return $FALSE
fi
SOURCE_EXT_LEN=`expr "$SOURCE_EXT" : '.*'`
SOURCE_EXT_LEN=`expr $SOURCE_EXT_LEN + 1`
SOURCE_NAME_LEN=`expr "$SOURCE_NAME" : '.*'`
REQ_LEN=`expr $SOURCE_NAME_LEN - $SOURCE_EXT_LEN`
SOURCE_NAME_WITHOUT_EXT=`echo $SOURCE_NAME | cut -c 1-$REQ_LEN`
return $TRUE
}
##############################################################################
# Name     : CheckFileExtension
#
# Overview : Checks the file extension is a valid one for the
#            source file.
#
# Input    : Call type (I = Internal, E = External)
#
# Returns  : $TRUE or $FALSE
#
# Notes    1 If the file extension of the requested source file
#            doesn't match the allowed file extensions, an error
#            message is displayed and $FALSE is returned.
#
#          2 If the call type is I, no message is displayed.
##############################################################################
CheckFileExtension ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# assign parameter
P_CALL_TYPE="$1"
for EXT in `echo $FILE_EXTS`
do
    if [ "$EXT" = "$SOURCE_EXT" ]
    then
        SOURCE_EXT_OK=Y
        break
    fi
done
# check the SOURCE_EXT_OK flag
if [ "$SOURCE_EXT_OK" = "Y" ]
then
    return $TRUE
else
    if ! ([ "${P_CALL_TYPE}" = "I" ])
    then
        DisplayMessage E "${INVALID_FILE_EXT}"
    fi
    return $FALSE
fi
}
```

```
##############################################################################
# Name     : LockSource
#
# Overview : Locks a specific source by creating an empty file with
#            the same name in the $LOCK_DIR directory.
#
# Input    : Source name
#
# Notes    1 It runs execsu to create a file in $LOCK_DIR that is
#            only writable by root.
##############################################################################
LockSource ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_SOURCE_NAME="$1"
ACTION="CR"
${SSCCARS_BIN}/execsu  ${ACTION} ${P_SOURCE_NAME} ${LOCK_DIR} >
➤  /dev/null 2>&1
RC=$?
if [ $RC -ne 0 ]
then
    DisplayMessage E "${SOURCE_NOT_LOCKED}"
    return $FALSE
else
    return $TRUE
fi
}
##############################################################################
# Name     : CheckLocation
#
# Overview : Checks that the user is not in the restricted
#            directories (ie all the source directories listed
#            in RESTRICTED_DIR_LIST).
#
# Returns  : $TRUE if the user is not in a restricted directory
#            $FALSE if the user is in a restricted directory
##############################################################################
CheckLocation ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
CUR_DIR=`pwd`
# check the current directory in RESTRICTED_DIR_LIST
for DIR in $RESTRICTED_DIR_LIST
    do
        if [ "$DIR" = "$CUR_DIR" ]
        then
            DIR_NAMR="${CUR_DIR}"
            DisplayMessage E "${RESTRICTED_DIR}"
            return $FALSE
        fi
```

```
    done
}
################################################################################
# Name     : ExtractSourceName
#
# Overview : Extracts the source name from the source file name
#            that contains version number (test_lecar is
#            extracted from file name test_lecar_11.trig).
#
# Input    : Source name with version number (eg valmi_11.proc)
#
# Notes    1 Populates global variable SOURCE_NAME_WITHOUT_VERSION.
#
#          2 If the call type is I, do not display a message.
################################################################################
ExtractSourceName ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# assign the parameter
P_SOURCE_NAME="$1"
P_CALL_TYPE="$2"
INDEX=1
SOURCE_NAME_WITHOUT_VERSION=`echo "${P_SOURCE_NAME}" |
➤  cut -d'_' -f$INDEX`
if [ "${SOURCE_NAME_WITHOUT_VERSION}" = "${P_SOURCE_NAME}" ]
then
    # file does not have version number
    # this file should not be here
    return $FALSE
fi
# extract source name
while true
do
    # extract segment separated by _
    SOURCE_NAME_WITHOUT_VERSION=`echo "${P_SOURCE_NAME}" |
    ➤  cut -d'_' -f${INDEX}`
    if [ "${SOURCE_NAME_WITHOUT_VERSION}" = "" ]
    then
        # all segments have been extracted
        # extract the relevant segmnet for file name
        INDEX=`expr $INDEX - 2`
        IND=1
        while [ !  $IND -gt ${INDEX} ]
        do
            if [ "${SOURCE_NAME_WITHOUT_VERSION}"  = "" ]
            then
                SOURCE_NAME_WITHOUT_VERSION=
                ➤  "${SOURCE_NAME_WITHOUT_VERSION}"`echo
                ➤  "${P_SOURCE_NAME}" | cut -d'_' -f${IND}`
            else
```

```
                SOURCE_NAME_WITHOUT_VERSION=
                ➤  "${SOURCE_NAME_WITHOUT_VERSION}_"`echo
                ➤  "${P_SOURCE_NAME}" | cut -d'_' -f${IND}`
            fi
            IND=`expr $IND + 1`
        done
        return $TRUE
    else
        INDEX=`expr $INDEX + 1`
    fi
done
}
#############################################################################
# Name     : UpdateSmart
#
# Overview : The function updates the SMART database.
#
# Input    : Mode: CIN    = Check in new
#                  CIE    = Check in existing
#                  CO     = Check out
#                  CANCEL = Cancel booking
#                  SRS    = Store released sources
#
# Returns  : $TRUE or $FALSE
#############################################################################
UpdateSmart ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_MODE="$1"
ORACLE_SID=smart ; export ORACLE_SID
APP_CODE="EUROCAT"
if [ "${P_MODE}" = "CIN" ]
then
    sqlplus -s / <<! > ${TEMP_FILE1} 2>&1
    WHENEVER SQLERROR EXIT 99
    ---
    --- create record in status_prog table
    ---
    INSERT into status_prog (sp_app_code,sp_program,sp_prog_language,
    ➤  sp_orginauth_init,sp_update_date,sp_update_time,
    ➤  create_sysdate, update_sysdate)
    VALUES ('${APP_CODE}',upper('${SOURCE_NAME}'),
    ➤  upper('${SOURCE_EXT}'),'JS',sysdate,sysdate,sysdate,sysdate);
    ---
    --- create record in status_prog_hist table
    ---
    INSERT into status_prog_hist (sph_app_code,sph_program,
    ➤  sph_program_version,sph_program_status,sph_prog_status_date,
    ➤  sph_prog_status_time,create_sysdate,update_sysdate)
    VALUES ('${APP_CODE}',upper('${SOURCE_NAME}'),1,'IN',
```

```
    ➤  sysdate,sysdate,sysdate, sysdate);
!
elif [ "${P_MODE}" = "CIE" ]
then
    sqlplus -s / <<!  > ${TEMP_FILE1}  2>&1
    WHENEVER SQLERROR EXIT 99
    ---
    --- create record in status_prog_hist table
    ---
    INSERT into status_prog_hist (sph_app_code,sph_program,
    ➤  sph_program_version,sph_program_status,sph_prog_status_date,
    ➤  sph_prog_status_time,create_sysdate,update_sysdate)
    VALUES ('${APP_CODE}',upper('${SOURCE_NAME_WITHOUT_VERSION}
    ➤  .${SOURCE_EXT}'),${SOURCE_VERSION},'IN',sysdate,sysdate,
    ➤  sysdate,sysdate);
    commit;
!
elif [ "${P_MODE}" = "CO" ]
then
    sqlplus -s / <<! > ${TEMP_FILE1} 2>&1
    WHENEVER SQLERROR EXIT 99
    ---
    --- create record in status_prog_hist table
    ---
    INSERT into status_prog_hist (sph_app_code,sph_program,
    ➤  sph_program_version,sph_program_status,sph_prog_status_date,
    ➤  sph_prog_status_time,create_sysdate,update_sysdate)
    VALUES ('${APP_CODE}',upper('${SOURCE_NAME_WITHOUT_EXT}
    ➤  .${SOURCE_EXT}'),${SOURCE_VERSION},'OUT',sysdate,sysdate,
    ➤  sysdate, sysdate);
    commit;
!
elif [ "${P_MODE}" = "CANCEL" ]
then
    sqlplus -s / <<!  > ${TEMP_FILE1}  2>&1
    WHENEVER SQLERROR EXIT 99
    DELETE  from status_prog_hist
    where sph_program = upper('${SOURCE_NAME}') and
    sph_app_code = '${APP_CODE}'                and
    sph_program_version = ${SOURCE_VERSION}     and
    sph_program_status  = 'OUT' ;
    commit  ;
!
elif [ "${P_MODE}" = "SRS" ]
then
    # build sql command file to store the released source names
    # in the database
    # build statement for STATUS_RELEASE table
    echo "
        insert into status_release(sr_app_code,sr_release,sr_who_init,
```

```
        ➤  sr_date,sr_desc,create_sysdate,update_sysdate)
        values('${APP_CODE}','${RELEASE_NAME}','${RELEASE_INITIATOR}',
        ➤  sysdate,'${RELEASE_DESC}',sysdate,sysdate);
        " > ${SQL_COMMAND_FILE}
    # build statements for STATUS_REL_PROG table
    # from the file $RELEASED_SOURCE_LIST
    cat ${RELEASED_SOURCE_LIST} | while read LINE
    do
         # extract source name
         if ! ExtractSourceName $LINE
         then
             continue
         fi
         # extract version number
         SOURCE_VERSION=`echo $LINE | sed 's/.*\_//' | cut -d'.' -f1`
         # extract file extension
         SOURCE_EXT=`echo $SOURCE_NAME | sed 's/^.*\.//'`
         echo "
             insert into status_rel_prog(srp_app_code,srp_release,
             ➤  srp_program,srp_prog_version,create_sysdate,
             ➤  update_sysdate)
             values('${APP_CODE}','${RELEASE_NAME}',
             ➤  '${SOURCE_NAME_WITHOUT_VERSION}.${SOURCE_EXT}',
             ➤  ${SOURCE_VERSION},sysdate,sysdate);
             " >> ${SQL_COMMAND_FILE}
    done
    echo " commit ; " >> ${SQL_COMMAND_FILE}
    # run the command file
    chmod 777 ${TEMP_FILE1}
    sqlplus -s / <<! > ${TEMP_FILE1} 2>&1
    @${SQL_COMMAND_FILE}
    exit
!
fi
# check Oracle error
if grep "ORA-" ${TEMP_FILE1} > /dev/null 2>&1
then
    # Oracle error found
    ERROR_MSG=`grep "ORA-" ${TEMP_FILE1} | head -1`
    DisplayMessage E "${ORACLE_ERROR}"
    view ${TEMP_FILE1}
    # delete inserted records
    if [ "${P_MODE}" = "CIN" ]
    then
        sqlplus -s / <<! > ${TEMP_FILE2} 2>&1
        WHENEVER SQLERROR EXIT 99
        ---
        --- delete record from status_prog and status_prog_hist table
        ---
         DELETE  from status_prog
```

```
          where sp_program = upper('${SOURCE_NAME}')    and
                sp_app_code = '${APP_CODE}' ;
          ----
          DELETE  from status_prog_hist
          where sph_program = upper('${SOURCE_NAME}')   and
                sph_app_code = '${APP_CODE}' ;
!
    elif [ "${P_MODE}" = "CIE" ]
    then
        sqlplus -s / <<! > ${TEMP_FILE2} 2>&1
        WHENEVER SQLERROR EXIT 99
        DELETE from status_prog_hist
        where sph_program = upper('${SOURCE_NAME}')    and
              sph_app_code = '${APP_CODE}'             and
              sph_program_version = ${SOURCE_VERSION}  and
              sph_program_status  = 'IN' ;
!
    elif [ "${P_MODE}" = "CO" ]
    then
        sqlplus -s / <<! > ${TEMP_FILE2} 2>&1
        WHENEVER SQLERROR EXIT 99
        DELETE from status_prog_hist
        where sph_program = upper('${SOURCE_NAME}')    and
              sph_app_code = '${APP_CODE}'             and
              sph_program_version = ${SOURCE_VERSION}  and
              sph_program_status = 'OUT' ;
!
    fi
    # check for Oracle error
    if grep "ORA-" ${TEMP_FILE2} > /dev/null 2>&1
    then
        ERROR_MSG=`grep "ORA-" ${TEMP_FILE2} | head -1`
        DisplayMessage E "${ORACLE_ERROR}"
        view ${TEMP_FILE2}
    fi
    return $FALSE
else
    # no Oracle error
    return $TRUE
fi
}
##############################################################################
# Name     : SmartDatabaseStatus
#
# Overview : The function checks the status of the SMART database.
#
# Returns  : $TRUE  - database is fully accessible
#            $FALSE - database is not accessible
##############################################################################
SmartDatabaseStatus ()
```

```
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
ORACLE_SID=smart ; export ORACLE_SID
if [ "${USERID}" = "root" ]
then
    chmod 777 ${TEMP_FILE1}
    su ${SMART_USER} -c "sqlplus -s / <<! > ${TEMP_FILE1} 2>&1
    WHENEVER SQLERROR EXIT 99
    exit
!"
else
    sqlplus -s / <<! > ${TEMP_FILE1} 2>&1
    WHENEVER SQLERROR EXIT 99
    exit
!
fi
if [ $? -eq 0 ]
then
    # database is fully accessible
    return $TRUE
else
    return $FALSE
fi
}
##############################################################################
# Name     : RemoveCopiedSourceFile
#
# Overview : Removes a file from a directory.
#
# Input    : Source file name
#            Source directory name
#
# Returns  : $TRUE or $FALSE
##############################################################################
RemoveCopiedSourceFile ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
P_FILE_NAME="$1"
P_DIR_NAME="$2"
ACTION="RM"
${SSCCARS_BIN}/execsu ${ACTION} ${P_FILE_NAME} ${P_DIR_NAME} >
➤  ${TEMP_FILE1} 2>&1
RC=$?
if [ $RC -ne 0 ]
then
    ERROR_MSG=`cat ${TEMP_FILE} | head -1`
    DisplayMessage E "${EXECSU_ERROR}"
    return $FALSE
else
    return $TRUE
```

```
fi
}
#############################################################################
# Name     : ExecuteSuperUser
#
# Overview : If the user agrees to go ahead with a task that
#            requires root privilege, the function obtains the
#            root password.
#
# Returns  : $TRUE or $FALSE
#############################################################################
ExecuteSuperUser ()
{
# get superuser execution confirmation from user
while true
do
clear
    echo "Root password is required for this option"
    echo "Do you want to proceed (Y/N):\c"
    read REPLY
    case $REPLY in
        y|Y) return $TRUE ;;
        n|N) return $FALSE ;;
         "") if [ "${FUNCTION_INTERRUPTED}" = "Y" ]
             then
                 FUNCTION_INTERRUPTED=N;
             else
                 DisplayMessage E "${INVALID_ENTRY}" ;
             fi ;;
          *) DisplayMessage E "${INVALID_ENTRY}" ;;
    esac
done
}
#############################################################################
# Name     : GetLatestVersion
#
# Overview : The function checks that the required source exists
#            and selects the latest version for copying to
#            $REQ_SOURCE.
#
# Argument : CHECKOUT_MODE (READ or UPDATE)
#
# Notes    1 If the parameter is READ, the name of the latest
#            version of the source is assigned to $REQ_SOURCE
#            and $TARGET_SOURCE.
#
#          2 If the parameter is UPDATE, the name of the latest
#            version of the source is assigned to $REQ_SOURCE
#            and $TARGET_SOURCE is assigned a file name with
#            version number equal to the latest version number + 1.
```

```
##############################################################################
GetLatestVersion ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
DisplayMessage I "${WORKING}" N
# assign the parameter
CHECKOUT_MODE=$1
# construct a match for required source
FILE_MATCH="${SOURCE_NAME_WITHOUT_EXT}_*"
# prepare the directory name
REQ_DIR="${SOURCE_DIR}/${SOURCE_EXT}"
# prepare the file name
REQ_FILE_LIST="${FILE_MATCH}"
# obtain a list of file matching the above
# $FILE_MATCH in $TEMP_FILE file
CURDIR=`pwd`
cd $REQ_DIR 2> /dev/null
if [ $? -ne 0 ]
then
    DisplayMessage E "${DIR_NOT_EXIST}"
    return ${FALSE}
fi
ls -1 ${REQ_FILE_LIST} 1> ${TEMP_FILE1} 2>/dev/null
# rework this list to find the exact match
FILE_COUNT=0
cat ${TEMP_FILE1} | while read LINE
do
    # extract the source name from source file name that
    # contains the version number
    if !  ExtractSourceName $LINE
    then
        continue
    fi
    if [ "${SOURCE_NAME_WITHOUT_VERSION}" =
    ➤  "${SOURCE_NAME_WITHOUT_EXT}" ]
    then
        FILE_COUNT=`expr $FILE_COUNT + 1`
    fi
done
cd $CURDIR
if [ $FILE_COUNT -eq 0 ]
then
    FILE_NAME="${SOURCE_NAME}"
    DIR_NAME="${REQ_DIR}"
    DisplayMessage E "${NO_SOURCE_FOUND}"
    return $FALSE
fi
# rework version number according to check-out mode
if [ "$CHECKOUT_MODE" = "READ" ]
then
```

```
    REQ_SOURCE="${SOURCE_NAME_WITHOUT_EXT}_${FILE_COUNT}.${SOURCE_EXT}"
    REQ_FILE="${REQ_DIR}/${REQ_SOURCE}"
    TARGET_SOURCE="${REQ_SOURCE}"
    # check the existence of required source
    if [ ! -f ${REQ_FILE} ]
    then
        DisplayMessage E "${SOURCE_ERROR}"
        return $FALSE
    fi
else
    REQ_SOURCE="${SOURCE_NAME_WITHOUT_EXT}_${FILE_COUNT}.${SOURCE_EXT}"
    FILE_COUNT=`expr ${FILE_COUNT} + 1`
    TARGET_SOURCE="${SOURCE_NAME_WITHOUT_EXT}_${FILE_COUNT}.
    ➤  ${SOURCE_EXT}"
fi
return $TRUE
}
##############################################################################
# Name     : SeekConfirmation
#
# Overview : The function seeks confirmation for a message in the
#            variable $MESSAGE
#
# Returns  : $TRUE or $FALSE
##############################################################################
SeekConfirmation ()
{
while true
do
    clear
    echo "$MESSAGE"
    read REPLY
    case  $REPLY in
       y|Y) return $TRUE ;;
       n|N) return $FALSE ;;
        "") if [ "${FUNCTION_INTERRUPTED}" = "Y" ]
            then
                FUNCTION_INTERRUPTED=Y ;
            else
                DisplayMessage E "${INVALID_ENTRY}" ;
            fi ;;
         *) DisplayMessage E "${INVALID_ENTRY}" ;;
    esac
done
}
##############################################################################
# Name     : CheckRootUserId
#
# Overview : Checks that the user is root.
##############################################################################
```

```
CheckRootUserId ()
{
if [ `id | tr "()" "::" | cut -d':' -f2` != "root" ]
then
    PROG="ssccars_lib.sh"
    DisplayMessage I "${NOT_ROOT_USER}"
    return  $FALSE
else
    return $TRUE
fi
}
##############################################################################
# Name     : CheckAuthorisedUserId
#
# Overview : Checks that the user is $AUTHORISED_USER.
##############################################################################
CheckAuthorisedUserId ()
{
if [ `id | tr "()" "::" | cut -d':' -f2` != "${AUTHORISED_USER}" ]
then
    PROG="ssccars_lib.sh"
    DisplayMessage  I "${NOT_AUTHORISED_USER}"
    return  $FALSE
else
    return $TRUE
fi
}
##############################################################################
# Name     : LeapYear
#
# Overview : Establishes whether a year is a leap year.
#
# Input    : Year
#
# Returns  : $TRUE for leap year
#            $FALSE otherwise
##############################################################################
LeapYear ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# assign the parameter
P_YEAR="$1"
# divide $P_YEAR by 4 to test for a leap year
RESULT=`bc <<!
scale=2
$YEAR/4
!`
if [ "`echo $RESULT | cut -d'.' -f2`" = "00" ]
then
    # year is a leap year
```

```
    return $TRUE
else
    # year is not a leap year
    return $FALSE
fi
}
##############################################################################
# Name     : ValidateLastPatchBuiltDate
#
# Overview : Checks that the last patch build date in the global
#            variable $LAST_PATCH_BUILT_DATE.
#
# Returns  : $TRUE or $FALSE
#
# Notes    1 The function calls following function:
#            o LeapYear
##############################################################################
ValidateLastPatchBuiltDate ()
{
trap "HandleInterrupt" $SIGINT $SIGTERM $SIGHUP $SIGTSTP
# check patch build date is numeric
if ([ `expr $LAST_PATCH_BUILT_DATE + 0` \
    -eq $LAST_PATCH_BUILT_DATE ]) > /dev/null 2>&1
then
    : # ok
else
    DisplayMessage E "${PATCH_BUILT_DATE_NOT_NUMERIC}"
    return $FALSE
fi
# verify overall string length
LEN=`expr "$LAST_PATCH_BUILT_DATE" : '.*'`
if ! ([ $LEN -eq 14 ])
then
    DisplayMessage E "${INVALID_DATE}"
    return $FALSE
fi
# verify the date format which should be <ddmmyyyyhhmiss>
DAY=`echo  "$LAST_PATCH_BUILT_DATE" | cut -c1-2`
MON=`echo  "$LAST_PATCH_BUILT_DATE" | cut -c3-4`
YEAR=`echo "$LAST_PATCH_BUILT_DATE" | cut -c5-8`
HOUR=`echo "$LAST_PATCH_BUILT_DATE" | cut -c9-10`
MIN=`echo  "$LAST_PATCH_BUILT_DATE" | cut -c11-12`
SEC=`echo  "$LAST_PATCH_BUILT_DATE" | cut -c13-14`
# validate day
LEN=`expr "$DAY" : '.*'`
if ! ([ $LEN -eq 2 ])
then
    DisplayMessage E "${INVALID_DAY_LEN}"
    return $FALSE
fi
```

```
if ! ([ $DAY -gt 0 -a $DAY -lt 32 ])
then
    DisplayMessage E "${INVALID_DAY}"
    return $FALSE
fi
# validate month
LEN=`expr "$MON" : '.*'`
if ! ([ $LEN -eq 2 ])
then
    DisplayMessage E "${INVALID_MONTH_LEN}"
    return $FALSE
fi
if ! ([ $MON -gt 0 -a $MON -lt 13 ])
then
    DisplayMessage E "${INVALID_MONTH}"
    return $FALSE
fi
# validate year
LEN=`expr "$YEAR" : '.*'`
if ! ([ $LEN -eq 4 ])
then
    DisplayMessage E "${INVALID_YEAR_LEN}"
    return $FALSE
fi
if ! ([ $YEAR -gt 0 ])
then
    DisplayMessage E "${INVALID_YEAR}"
    return $FALSE
fi
# validate day and month except the month of february
if [ $MON -eq 01 -o $MON -eq 03 -o $MON -eq 05 -o \
     $MON -eq 07 -o $MON -eq 08 -o $MON -eq 10 -o \
     $MON -eq 12 ]
then
    if ! ([ $DAY -gt 0 -a $DAY -lt 32 ])
    then
        DisplayMessage E "${INVALID_DAY}"
        return $FALSE
    fi
elif [ $MON -eq 04 -o $MON -eq 06 -o $MON -eq 09 -o \
       $MON -eq 11 ]
then
    if ! ([ $DAY -gt 0 -a $DAY -lt 31 ])
    then
        DisplayMessage E "${INVALID_DAY}"
        return $FALSE
    fi
fi
# special handling for February
if LeapYear "${YEAR}"
```

```
then
    if [ $MON -eq 02 ]
    then
        if ! ([ $DAY -gt 0 -a $DAY -lt 30 ])
        then
            DisplayMessage E "${INVALID_DAY}"
            return $FALSE
        fi
    fi
else # not a leap year
    if [ $MON -eq  02 ]
    then
        if ! ([ $DAY -gt 0 -a $DAY -lt 29 ])
        then
            DisplayMessage E "${INVALID_DAY}"
            return $FALSE
        fi
    fi
fi
# validate hours
LEN=`expr "$HOUR" : '.*'`
if ! ([ $LEN -eq 2 ])
then
    DisplayMessage E "${INVALID_HOUR_LEN}"
    return $FALSE
fi
if ! ([ $HOUR -eq 0 -o $HOUR -lt 24 ])
then
    DisplayMessage E "${INVALID_HOUR}"
    return $FALSE
fi
# validate minutes
LEN=`expr "$MIN" : '.*'`
if ! ([ $LEN -eq 2 ])
then
    DisplayMessage E "${INVALID_MIN_LEN}"
    return $FALSE
fi
if ! ([ $MIN -eq 0 -o $MIN -lt 60 ])
then
    DisplayMessage E "${INVALID_MINUTES}"
    return $FALSE
fi
# validate seconds
LEN=`expr "$SEC" : '.*'`
if ! ([ $LEN -eq 2 ])
then
    DisplayMessage E "${INVALID_SEC_LEN}"
    return $FALSE
fi
```

```
if ! ([ $SEC -eq 0 -o $SEC -lt 60 ])
then
    DisplayMessage E "${INVALID_SECONDS}"
    return $FALSE
fi
return $TRUE
}
##############################################################################
# Name     : FormatUnderscores
#
# Overview : The function assigns a number of underscores to the
#            variable UNDERSCORE, which is used with a header in
#            $HEADER.
#
# Input    : The line containing the header
##############################################################################
FormatUnderscores ()
{
# assign parameter
LINE="$1"
# initialize UNDERSCORE
UNDERSCORE=
# initialize index
CHAR_IND=1
# get no of characters in  $LINE
NO_CHARS=`echo "$LINE" | wc -c`
# subtract the carriage return
NO_CHARS=`expr $NO_CHARS - 1`
while [ "$CHAR_IND" -le "$NO_CHARS" ]
do
    UNDERSCORE="${UNDERSCORE}="
    CHAR_IND=`expr $CHAR_IND + 1`
done
}
##############################################################################
# Name     : PerformSanityCheck
#
# Overview : The function performs sanity checks to make sure that
#            it's able to run ssccars.sh.
##############################################################################
PerformSanityCheck ()
{
# check SSCCARS directory
if [ ! -d "${SSCCARS_DIR}" ]
then
    DIR_NAME="${SSCCARS_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS source directory
```

```
if [ ! -d "${SOURCE_DIR}" ]
then
    DIR_NAME="${SOURCE_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS release directory
if [ ! -d "${ROOT_RELEASE_DIR}" ]
then
    DIR_NAME="${ROOT_RELEASE_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS release log directory
if [ ! -d "${BUILT_RELEASE_LOG_DIR}" ]
then
    DIR_NAME="${BUILT_RELEASE_LOG_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS log directory
if [ ! -d "${LOG_DIR}" ]
then
    DIR_NAME="${LOG_DIR}"
    DisplayMessage  E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS lock directory
if [ ! -d "${LOCK_DIR}" ]
then
    DIR_NAME="${LOCK_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
# check SSCCARS temporary directory
if [ ! -d "${TEMP_DIR}" ]
then
    DIR_NAME="${TEMP_DIR}"
    DisplayMessage E "${DIR_MISSING}"
    return $FALSE
fi
if [ ! -w "${TEMP_DIR}" ]
then
    DIR_NAME="${TEMP_DIR}"
    DisplayMessage E "${DIR_NOT_WRITABLE}"
    return $FALSE
fi
# check chkout.sh script
if [ ! -f "${SSCCARS_BIN}/chkout.sh" ]
then
```

```
    FILE_NAME="${SSCCARS_BIN}/chkout.sh"
    DisplayMessage E "${FILE_MISSING}"
    return $FALSE
fi
if [ ! -x "${SSCCARS_BIN}/chkout.sh" ]
then
    FILE_NAME="${SSCCARS_BIN}/chkout.sh"
    DisplayMessage E "${FILE_NOT_EXECUTABLE}"
    return $FALSE
fi
# check chkin.sh script
if [ ! -f "${SSCCARS_BIN}/chkin.sh" ]
then
    FILE_NAME="${SSCCARS_BIN}/chkin.sh"
    DisplayMessage E "${FILE_MISSING}"
    return $FALSE
fi
if [ ! -x "${SSCCARS_BIN}/chkin.sh" ]
then
    FILE_NAME="${SSCCARS_BIN}/chkin.sh"
    DisplayMessage E "${FILE_NOT_EXECUTABLE}"
    return $FALSE
fi
# check br.sh  script
if [ ! -f "${SSCCARS_BIN}/br.sh" ]
then
    FILE_NAME="${SSCCARS_BIN}/br.sh"
    DisplayMessage E "${FILE_MISSING}"
    return $FALSE
fi
if [ ! -x  "${SSCCARS_BIN}/br.sh" ]
then
    FILE_NAME="${SSCCARS_BIN}/br.sh"
    DisplayMessage E "${FILE_NOT_EXECUTABLE}"
    return $FALSE
fi
return $TRUE
}
```

This article continues in next month's issue of *AIX Update*, when we'll publish the first of a number of utilities, each of which implements one of SSCCARS's menu items.

Arif Zama
DBA/Administrator
High-Tech Software

IBM has announced SecureWay Host Publisher Version 2.1, which Web-enables existing applications without requiring changes to them. SecureWay integrates multiple sources of host data into a single Web page, giving end-users the appearance of a single new application. It supports any HTML-based browser, not necessarily Java-enabled.

There are two main components: Host Publisher Studio is a collection of task-oriented tools for creating Host Publisher applications and Host Publisher Server comprises WebSphere Application Server and Host Publisher runtime components for executing the applications created with Studio.

Developers can create Web-to-host applications using Studio and publish them to the Server. The Web-to-host applications are based on integration objects, which comprise reusable Java beans that can establish a connection with a host, accept user input if required, navigate to and extract data from an application, and disconnect from the host and end the connection.

Applications created with the Studio will run unchanged in AIX, NT, and Solaris.

Out on November 30, Version 2.1 costs US$15,000.

IBM also announced SecureWay Communications Server for AIX Version 5.0.4. Enhancements provide secure Web-to-host support for Telnet users, Secure Sockets Layer (SSL) data encryption and client/server authentication, and IP redirection. Out on November 5, no prices were given.

For further information contact your local IBM representative.

* * *

Hummingbird Communications has announced Genio, a data transformation tool that's now available for AIX and Solaris servers, adding to NT support. Based on a metadata repository of business rules, it transforms, cleanses, enriches, and directs the flow of information across decision support systems including data marts, data warehouses, and OLAP environments. Other features include object sharing and sub-modules for managing complex transformations, plus native connectivity to Hyperion Essbase. Out now, prices are available on request from the vendor.

For further information contact:
Hummingbird Communications Ltd, 1 Sparks Avenue, North York, Ontario M2H 2W1, Canada
Tel: +1.416 496 2200
Fax: +1.416 496 2207
Web: http://www.hummingbird.com

Hummingbird Communications Ltd, Mulberry Business Park, Fishponds Road, Wokingham, Berkshire RG41 2GY, UK
Tel: +44 118 978 2800
Fax: +44 118 978 2700

* * *