<A>APPENDIX D

<B>DCL AND SHELL SCRIPTS

In ORACLE7.0 Administration and Management I provided numerous DCL and a few shell scripts o help in the management of Oracle databases. I provided more and better DCL scripts simply because I was spending a majority of my time supporting databases in the OpenVMS arena. Well, as they say, the worm has turned. Since that volume was published I have worked less with VMS and more with UNIX (of various denominations, Dynix, SCO, BSD, HPUX, Solaris). I'm afraid I don't have any new DCL scripts but I will include the ones from the first book. However, I have written many useful UNIX shell scripts so the number, and I hope, the quality of UNIX scripts has increased.

The first set of scripts we will show are Digital Command Language (DCL) scripts. It is logical to provide a central file to provide logical and symbol definitions. Therefore the first script will be a file that does just this. The file is called ORACLE_LOG.COM.

```
$! This command procedure DEFINES logicals used by the Oracle Management
$! menu.
$! MRA, REV 0, 10/23/92
$!
$! First, get logicals defined:
$!
$ DEFINE/NOLOG ORACLE$COM
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.COM]
$ DEFINE/NOLOG ORACLE$SQL
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.SQL]
$ DEFINE/NOLOG ORACLE$MENU
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.MENUS]
$ DEFINE/NOLOG ORACLE$DOC     DUA2:[NM91263.ORACLE6.DOC]
$ DEFINE/NOLOG ORACLE$ODDS
      M_ORA_DISK0:[m_oracle.dba_tools.ddview.sql]
$ DEFINE/NOLOG oracle$log           DUA2:[nm91263.oracle6.log]
$ DEFINE/NOLOG ORACLE$EXP     m_ora_disk2:[m_ORACLE6.db_case.exportS]
$ DEFINE/NOLOG ORA$DBA_REP
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.reports]
$ DEFINE/NOLOG ORA$EXP_COM    DUA2:[NM91263.ORACLE6.COM.EXPORTS]
$ DEFINE/NOLOG ORA$ODDSFRM
      M_ORA_DISK0:[m_oracle.dba_tools.ddview.sql]
$ define/nolog/TRANS=CONC           ORA$ARCHIVE M_ORA_DISK0:
$ DEFINE/NOLOG                ORA_ANSI MH5102P
$ DEFINE/NOLOG                ORA_LAND PRT_HBC5_R1_L
$ define/nolog ORA$IOUG
      M_ORA_DISK0:[m_oracle.dba_tools.dbutil.sql]
```

```
$ define/nolog ora$com_rep
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.reports]
$ define/nolog ora_status_reports
      M_ORA_DISK0:[m_oracle.dba_tools.dbstatus.reports]
$ define/nolog CASE_HP_CMD    "print /queue=plt_hbc5_draftpro"
$ define/nolog CASE_PS_CMD    "print /queue=prt_hbc5_r1_p"
$ define/nolog CASE_SDPRINT   "SYS$PRINT"
$ define/nolog  SDD$PRINT           "print /queue=prt_hbc5_r1_a"
$ define/nolog  SDD$WPRINT          "print /queue=prt_hbc5_r1_l"
$ define/nolog  SDD_QUEUE           "PRT_HBC5_r1_A"
$ define/nolog adhoccon             mmrd01"""sqlnet
sqlnet"""::"""task=ordnadhoc"""
$ define/nolog casecon              mmrd01"""sqlnet
sqlnet"""::"""task=ordnkcgc"""
$ define/nolog DEVcon               mmrd15"""sqlnet
sqlnet"""::"""task=ordnDEV"""
$ define/notran elcon               "ELWOOD:oracle"
$ define tcpcase              "157.206.11.15:"""CASE""":4096,5,YES"
$!
$! Symbol definitions:
$!
$! Report related symbols:
$!
$ PREP            :== "@ORACLE$COM:PRINT_REPORTS.COM"
$ GREP            :== "@ORACLE$COM:GEN_REPORTS.COM"
$!
$! Menu related symbols:
$!
$ ddview          :== "''runmenu' ddview -m f"
$ dbstatus        :== "''runmenu' dbstatus -m f"
$ DBUTILS         :== "''RUNMENU' DBUTILS -M F"
$ DB_TOOLS        :== "''RUNMENU' DBTOOLS -M F"
$ DBTOOLS         :== "''RUNMENU' DBTOOLS -M F"
$ CASE_MENU       :== "''RUNMENU' CASE_MENU -M F"
$!
$! Specific Instance Startup and Shutdown symbols:
$! The instance specific ORAUSER must be run for symbol to work.
$!
$ start_kcgc            :== "@ORA_INSTANCE:startup_EXCLUSIVE_CASE.com"
$ stop_kcgc             :== "@ORA_INSTANCE:shutdown_CASE.com"
$ start_APPL            :== "@ORA_INSTANCE:startup_EXCLUSIVE_PROD.com"
$ stop_APPL             :== "@ORA_INSTANCE:shutdown_PROD.com"
$ start_ADHOC     :== "@ORA_INSTANCE:startup_EXCLUSIVE_ADHOC.com"
$ stop_ADDHOC     :== "@ORA_INSTANCE:shutdown_ADHOC.com"
$ START_RECDEV    :== "@ORA_INSTANCE:STARTUP_EXCLUSIVE_RECDEV.COM"
$ STOP_RECDEV     :== "@ORA_INSTANCE:SHUTDOWN_RECDEV"
$ START_RECRUN    :== "@ORA_INSTANCE:STARTUP_EXCLUSIVE_RECRUN.COM"
$ STOP_RECRUN     :== "@ORA_INSTANCE:SHUTDOWN_RECRUN"
$!
$! Management Menu related symbols:
$!
$ MAN             :=="@ORACLE$COM:MANAGE_ORACLE.COM"
$ DEV_DIR         :=="ORACLE.DEV"
$ LIMS_DIR        :=="ORACLE.LIMS"
$ KCGC_DIR        :=="M_ORACLE6.DB_CASE"
$ APPL_DIR        :=="M_ORACLE6.DB_APPL"
$ ADHOC_DIR       :=="M_ORACLE6.DB_PROD"
$ GORECDEV        :== -
"@M_ORA_DISK0:[M_ORACLE.ORACLE6.DB_RECDEV]ORAUSER_RECDEV.COM
$ GORECRUN        :== -
 "@M_ORA_DISK0:[M_ORACLE.ORACLE6.DB_RECRUN]ORAUSER_RECRUN.COM
$!
$EXIT
```

This script is run by all users who will be using the menu system.

Once the logicals and symbols are set, the users should be sent to a central menu that accesses all other menus. This ensures that the menus are properly invoked and things are done in a standard form. The central management menu for DBA activities in these examples is called MANAGE_ORACLE.COM appropriately enough. Its listing follows.

```
$!
$! NAME       : MANAGE_ORACLE.COM
$! PURPOSE  : ALLOW EASE OF ORACLE SYSTEM MANAGEMENT
$! USE           : @ORACLE$COM:MANAGE_ORACLE
$! Limitations    : None
$! Revisions:
$!          Date        Modified By Reason For change
$!          03-AUG-1991 MIKE AULT   INITIAL CREATE
$!          23-AUG-1991 MIKE AULT   ADDED DBA*ASSIST
$!          04-SEP-1992 MIKE AULT   ADDED CASE ACCESS
$!          23-OCT-1992 MIKE AULT   REP FP WITH LOGS
$!
$ ON CONTROL_Y THEN GOTO START
$!
$! Define local use logicals
$!
$ DEFINE/NOLOG REPORT$LOCATION oracle$sql
$ define/nolog report$dest ora_status_reports
$!
$! define local symbols
$!
$ DELETE :== ""
$ SEE :== "DEFINE/USE SYS$INPUT SYS$COMMAND"
$ say :== "WRITE SYS$OUTPUT"
$ CON_STR:==""
$ node = f$getsyi("NODENAME")
$ USER == F$GETJPI("","USERNAME")
$ USER == F$EDIT("''USER'","TRIM")
$ define node 'node'
```

```
$!
$ START:
$!
$ CLS       ! If not defined by SYS. Manager, set to SET/TERM/WID=80
$!
$! The next line displays a static menu file, can be generated via FMS
or by editor.
$!
$ TYPE ORACLE$MENU:MANAGE_oracle.MEN
$!
$ DB == F$TRNLNM("ORA_SID")
$!
$ READ/PROMPT=-
"      USER: ''USER' NODE: ''NODE' DB: ''DB' ENTER CHOICE:" SYS$COMMAND
ANS
$!
$! The next several if constructs check for special commands
$!
$ IF (ANS .EQS. "LO" .OR. ANS .EQS. "lo")
$     then
$     lo
$ ENDIF
$!
$ IF (ANS .EQS. "EX" .OR. ANS .EQS. "ex")
$     then
$     cls
$     goto END_IT
$ ENDIF
$!
$ IF (ANS .EQS. "SLEEP" .OR. ANS .EQS. "sleep")
$     then
$     @ORACLE$COM:idle.com
$     GOTO START
$ ENDIF
$!
$!
$ IF (ANS .EQS. "spawn" .OR. ANS .EQS. "SPAWN")
$     then
$     SEE
$     SPAWN
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "ed" .OR. ANS .EQS. "ED")
$     then
$     SAY " "
$     @ORACLE$COM:EDIT_IT
$     WAIT 00:00:03
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "MAIL" .OR. ANS .EQS. "mail")
$     then
$     SAY " "
$     SAY "INVOKING VMS MAIL ...."
$     SEE
$     MAIL
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "PHN" .OR. ANS .EQS. "phn")
$     then
$     SAY " "
$     SAY "INVOKING VMS PHONE ...."
$     SEE
$     PHONE
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "PRT" .OR. ANS .EQS. "prt")
$     then
$     SAY " "
$     @oracle$com:prt_it
$     GOTO START
$ ENDIF
$!
$! the next set of if constructs allow changing the database instances
accessed
$!
$ IF (ANS .EQS. "PROD" .OR. ANS .EQS. "prod")
$     then
$     SAY " "
$     see
$     @M_ORA_DISK0:[M_ORACLE.ORACLE6.DB_PROD]ORAUSER_PROD
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "adhoc" .OR. ANS .EQS. "ADHOC")
$     then
$     SAY " "
$     see
$     @M_ORA_DISK0:[M_ORACLE.ORACLE6.DB_ADHOC]ORAUSER_ADHOC
$     GOTO START
$ ENDIF
$!
$ IF (ANS .EQS. "case" .OR. ANS .EQS. "CASE")
$     then
$     SAY " "
$     see
$     @M_ORA_DISK0:[M_ORACLE.ORACLE6.DB_CASE]ORAUSER_CASE
$     GOTO START
$ ENDIF
$!
$! Actual Menu section
$!
$ ON ERROR THEN GOTO START
$ ANS = F$INTEGER(ANS)
$!
```

```
$ IF (ANS.LT.1) .OR. (ANS.GT.14)    ! This line changes as menu items
added/deleted
$ THEN
$     GOTO START
$ ENDIF
$!
$! Process Menu options, wish DCL had a CASE statement like UNIX!
$! Notice all options call other procedures, this allows ease of
maintenance
$!
$ IF (ANS.EQ.1)
$ THEN
$     @ORACLE$COM:MANAGE_USER.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.2)
$ THEN
$     @ORACLE$COM:MANAGE_TABLESPACE.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.3)
$ THEN
$     @ORACLE$COM:QUEUE_MENU.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.4)
$ THEN
$     @ORACLE$COM:EXPORT_MENU.COM
$!
$ GOTO START
$ ENDIF
$!
$! This option calls the Oracle supplied report writer administration
menu.
$!
$ IF (ANS.EQ.5)
$ THEN
$     SRW_ADMIN
$!
$ GOTO START
$ ENDIF
$!
$! This option calls the Oracle supplied installation menu.
$!
$ IF (ANS.EQ.6)
$ THEN
$     ORACLEINS
$!
```

```
$ GOTO START
$ ENDIF
$!
$! This option uses the startup symbols createdin ORACLE_LOG.COM
$!
$ IF (ANS.EQ.7)
$ THEN
$ ORA_SID == F$TRNLNM("ORA_SID")
$     READ/PROMPT = "STARTUP INSTANCE ''ORA_SID'? (Y OR N): ->" -
            SYS$COMMAND INS
$     INS = F$EDIT("''INS'","UPCASE")
$     IF ("''INS'" .EQS. "Y")
$ THEN
$     START_'ORA_SID'
$ ENDIF
$!
$ GOTO START
$ ENDIF
$!
$! The next option uses the shutdown symbols defined in ORALCE_LOG.COM
$!
$ IF (ANS.EQ.8)
$ THEN
$ ORA_SID == F$TRNLNM("ORA_SID")
$     READ/PROMPT = "SHUTDOWN INSTANCE ''ORA_SID'? (Y OR N): ->"
SYS$COMMAND INS
$     INS = F$EDIT("''INS'","UPCASE")
$     IF ("''INS'" .EQS. "Y")
$ THEN
$     STOP_'ORA_SID'
$ ENDIF
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.9)
$ THEN
$     @ORACLE$COM:ARCHIVE_MENU.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.10)
$ THEN
$     @ORA_install:ora_insutl
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.11)
$ THEN
$     ora_sid == f$trnlnm("ORA_SID")
$     ora_sid:== "''ora_sid'"
$     @ORACLE$COM:ORACLE_LOG_MENU.COM
```

```
$!
$ GOTO START
$ ENDIF
$!
$! The next few options call SQL*Menu menus, the symbols are defined in
ORACLE_LOG.COM
$!
$ IF (ANS.EQ.12)
$ THEN
$     define/nolog report$location ORACLE$COM
$     define/nolog report$DEST ORA_STATUS_REPORTS
$     ORA_SID == F$TRNLNM("ORA_SID")
$     ORA_SID:=='ORA_SID'
$           GOSUB GET_USER_DATA
$           SEE
$           DBSTATUS 'un'/'pw'
$!
$ goto start
$ ENDIF
$!
$ IF (ANS.EQ.13)
$ THEN
$     define/nolog report$location ora$ioug
$     define/nolog report$dest ora_STATUS_REPORTS
$     ORA_SID == F$TRNLNM("ORA_SID")
$     ORA_SID:=='ORA_SID'
$           see
$           dbutils
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.14)
$ THEN
$     DEFINE/NOLOG REPORT$LOCATION ora$oddsfrm
$     DEFINE/NOLOG REPORT$DEST ora_STATUS_REPORTS
$     ORA_SID == F$TRNLNM("ORA_SID")
$     ORA_SID:=='ORA_SID'
$           SEE
$           DDVIEW
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.15)
$ THEN
$     GOSUB GET_USER_DATA
$     SEE
$     DBTOOLS 'UN'/'PW'
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.16)
$ THEN
$     GOSUB GET_USER_DATA
$     SEE
$     RUNMENU CASE_MENU 'UN'/'PW' -M F
$!
$ endif
$ GOTO START
$!
$ GET_USER_DATA:
$!
$ READ/PROMPT = "INPUT USER NAME: ---->" SYS$COMMAND UN
$ SET TERM/NOECHO
$ READ/PROMPT = "INPUT USER PASSWORD: ---->" SYS$COMMAND PW
$ SET TERM/ECHO
$ RETURN
$!
$ end_it:
$!
$ EXIT
```

As you can see, this main menu does very few direct actions. It allows the user access to system features such as the editor, printing, spawn to the OS, mail and phone, but still keeps them within the menu structure. Ifdesired the spawn option can be removed and a "captive" user will result.

The menu also allows the user to switch between available instances, in this case "CASE", "ADHOC" and "PROD". Options can also be added to "go" to different nodes using "SET HOST".

One of the drawbacks of the DCL system is lack of a "case" command such as in UNIX shell languages. This requires the developer to result to the IF-THEN-ELSE type structure shown. In spite of this lack, some very easy to use scripts can be built.

The MANAGE_ORACLE.MEN file is shown below.

```
________________________________________________
              DATABASE MANAGEMENT MENU
________________________________________________
ED-EDIT  PHN-PHONE  PRT-PRINT MAIL-VMS MAIL EX-EXIT MENU
________________________________________________
```

```
1. MANAGE ORACLE USERS MENU   9.   ARCHIVE MENU
2. MANAGE TABLESPACES MENU    10. INSTALL TOOLS SHARED
3. QUEUE MENU                11. MANAGE ORACLE LOGS
4. EXPORT MENU               12. DB STATUS MENU
5. REPORT WRITER ADMIN MENU   13. DB UTILITIES MENU
6. ORACLEINS MENU                  14. DATA DIC. VIEWS
7. STARTUP DATABASE          15. DB TOOLS MENU
8. SHUTDOWN DATABASE         14. CASE ACCESS MENU
_______________________________________________________
```

Menu items 12, 13, and 14 can be combined into a DBA Tools SQL*Menu menu. Item 15's SQL*Menu menu calls SQL*Forms, SQL*Plus, SQL*ReportWriter and SQLDBA. Item 16's SQL*Menu menu calls CASE*Dictionary, CASE*Designer and CASE*Generator.

Menu Item 12 calls a SQL*Menu menu that allows the generation, display and print of most of the DBA related reports shown in Chapter 4. Menu Item 13 calls a SQL*Menu menu that allows the generation, display and print of several useful reports gleaned from the Oracle IOUG (International Oracle User's Group) distribution disks. Menu Item 14 calls a SQL*Menu menu that runs Tim Olesen's Oracle Data Dictionary (ODDS) system of forms and reports.

The scripts used in menu items 13 and 14 can be obtained from the IOUG. The menus developed to run the scripts are provided on the disk that is the companion to this book.

Let's look at a few of the secondary menus and see how processing is handled in them. The first sub-menu does user related tasks. It would probably be better todo this through a combination of SQL*Menu and SQL*Forms, but this method is simpler and it works.

The DCL script is called MANAGE_USERS.COM. The listing of this script follows.

```
$!
$! NAME     : MANAGE_USER.COM
$! PURPOSE  : ASSIST DBA IN MANAGING ORACLE USERS
$! USE            : FROM MANAGE_ORACLE.COM
$! Limitations    : None
$! Revisions:
$!         Date        Modified By Reason For change
$!         21-AUG-1991 MIKE AULT   INITIAL CREATE
$!         23-OCT-1992 MIKE AULT   REPLACE FULLPATH WITH LOGICALS
$!
$ DELETE :== ""
$!
$ START:
$!
$ CLS
$!
$! Display menu
$!
$ TYPE ORACLE$MENU:USER_MENU.MEN
$!
$ READ/PROMPT="           ENTER CHOICE:" SYS$COMMAND ANS
$!
$ ANS = F$INTEGER(ANS)
$!
$ IF (ANS.EQ.99) THEN GOTO END
$ IF (ANS.LT.1) .OR. (ANS.GT.6) THEN GOTO START
$!
$ IF (ANS.EQ.1)
$ THEN
$ @ORACLE$COM:ADD_USER.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.2)
$ THEN
$ @ORACLE$COM:SWITCH_USER.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.3)
$ THEN
$ @ORACLE$COM:GRANT.COM
```

```
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.4)
$ THEN
$ @ORACLE$COM:REVOKE.COM
$!
$ GOTO START
$!
$ ENDIF
$!
$ IF (ANS.EQ.5)
$ THEN
$ @ORACLE$COM:REVOKE_CONNECT.COM
$!
$ GOTO START
$ ENDIF
$!
$ IF (ANS.EQ.6)
$ THEN
$ @ORACLE$COM:USER_REPORT.COM
$!
$ GOTO START
$ ENDIF
$!
$ END:
$!
$ EXIT
```

Again, as with the main menu, this menu does no actual processing. This menu calls sub-routines that actually do the work. This allows wholesale replacement of sub-routines without effecting the calling menu. The menu actually displayed is shown below.

```
________________________________________
     DATABASE USER MANAGEMENT MENU
________________________________________

1. ADD ORACLE USER
2. CHANGE ORACLE USERS DEFAULT TABLESPACE
3. GRANT RESOURCE TO ORACLE USER
4. REVOKE RESOURCE FROM ORACLE USER
5. REMOVE ORACLE USER
6. GENERATE USER REPORT

99. RETURN TO MAIN MENU

________________________________________
```

Let's look at one of this menus sub-routines. The first menu item allows a user to create a new Oracle account. The script is called ADD_USER.COM and is listed below.

```
$!
$! NAME     : add_user.com
$! PURPOSE  : add an oracle user to the database
$! USE            : use from command line
$! Limitations    : NONE
$! parameters     : NONE
$! Revisions:
$!          Date        Modified By Reason For change
$!          5 DEC 90    Mike Riggs  Initial Checkin
$!          23 OCT 92   MIKE AULT   REPLACE FULLPATHS WITH LOGICALS
$!
$ INQUIRE S_USER   "Enter DBA userid--->"
$!
$! To protect the DBA password from prying eyes, set so terminal doesn't
echo
$!
$ SET TERM/NOECHO
$ INQUIRE S_PW     "Enter DBA password->"
$ SET TERM/ECHO
$!
$ cls
$!
$ INQUIRE U_ID     "Enter new account name---------->"
$ INQUIRE PW       "Enter new account password------>"
$ INQUIRE TS       "Enter user default tablespace--->"
$ INQUIRE TST      "Enter user Temporary tablespace->"
$!
$ sqlplus -s 'S_USER'/'S_PW' -
      @ORACLE$SQL:add_user 'u_id' 'PW' 'TS' 'TST'
$!
$ INQUIRE ANS "Is this user a developer?"
$!
$ IF ANS .NES. "Y" THEN GOTO ADD_USER_DONE
$!
$! add sqlreportwriter developer's access
$!
$ WRITE SYS$OUTPUT "For SQL*Reportwriter access, enter user name here."
$ SEE
$ sqlplus 'S_USER'/'S_PW' @ORA_ROOT:[SQLREPORTWRITER.BIN]SRW_GRNT
$!
$! add sqlmenu 50 developer's access
$!
$ 'GENMENU'  's_user'/'s_pw' -gd 'u_id' -s
$!
$ADD_USER_DONE:
```

```
$	EXIT
```

Finally, we see some processing done. Note the call to the SQL script to actually insert the user. This allows upgrades to the SQL script without modifications to the calling DCL script.

```
REM
REM NAME            :add_user.sql
REM PURPOSE         : adds a developer user to the database
REM                 includes access for all products
REM
REM USE             : called by add_user.com
REM Limitations     : None
REM Revisions       :
REM        Date          Modified By    Reason For change
REM        5 DEC 90      Mike Riggs     Initial Checkin
REM        3 AUG 91      MIKE AULT      DONT USE SYSTEM AS TEMP!
CREATE USER &1 IDENTIFIED BY &2
DEFAULT TABLESPACE &3
TEMPORARY TABLESPACE &&4;
QUOTA UNLIMITED ON &&4
EXIT
```

Several utility scripts written in DCL need mentioning. If we wanted to do several different printout styles, say portrait, landscape or to a terminal we would either have to come up with the command each time we wanted to do a different style or create a DCL routine we could pass a minimum of parameters and have it do the work. PRINT_REPORT.COM that is referenced by the PREP symbol in ORACLE_LOG.COM is one such DCL routine. This routine is used by several of the support menus to specify how a report should be printed. Of course the report has to be generated first. This is accomplished with GEN_REPORTS.COM and GEN_BATCH_REPORTS.COM. These two report generation scripts work together to allow the user to generate a report whether it be from ORA*Report or SQL*Plus. The listing for these utility scripts follow.

```
$! GEN_REPORTS.COM
$!
$! PROCEDURE TO ALLOW FOR THE GENERATION OF REPORTS FROM THE
$! DATABASES
$!
```

```
$! REV. 0 12-MAY-1992 M. AULT
$!
$!  REV:    DATE:       REASON:                PERSON:
$!  ----   --------     ---------------------  -------
$!   0     12-MAY-1992  INITIAL CREATE         M. AULT
$!   1     23-OCT-1992  REP FP WITH LOGS       M. AULT
$!------------------------------------------------------------------
$! PARAMETERS:
$!
$! P1 : REPORT TO RUN
$! P2 : USERNAME (ORACLE)
$! P3 : PASSWORD (ORACLE)
$! P4 : TYPE OF REPORT 1-RPT, 2-REPORTWRITER, 3-SQL
$! P5 : B FOR BATCH, N FOR NORMAL
$! P6 : Y TO USE PARAMETER FORM, N TO NOT USE FORM
$! P7 : TO PASS VARIABLE TO SQLPLUS REPORT
$!__________________________________________________________________
$!
$! ASSUMPTIONS:
$!
$! 1. EACH NON-RPT REPORT MUST SPECIFY ITS OWN DESTINATION
$! 2. REPORT$LOCATION IS A LOGICAL POINTING TO THE DIRECTORY FOR THE
$!    REPORT SPECIFICATION FILE
$! 3. IF REPORT$LOCATION IS NOT SPECIFIED, IT WILL DEFAULT TO SYS$LOGIN
$! 4. REPORT$DEST IS A LOGICAL POINTING TO THE LOCATION OF THE REPORT
$!    OUTPUT
$!    IF IT IS NOT SPECIFIED IT DEFAULTS TO THE REPORT$LOCATION VALUE
$!------------------------------------------------------------------
$!
$ INQUIRE ANS "CONTINUE?"
$ SET NOON
$!
$ NODE = F$GETSYI("NODENAME")
$!
$! CHECK VALUE OF REPORT$LOCATION
$!
$ IF F$TRNLNM("REPORT$LOCATION") .EQS. ""
$ THEN
$     DEFINE/NOLOG REPORT$LOCATION SYS$LOGIN
$ ENDIF
$!
$! CHECK VALUE OF REPORT$DEST
$!
$ IF F$TRNLNM("REPORT$DEST") .EQS. ""
$ THEN
$    DEFINE/NOLOG REPORT$DEST REPORT$LOCATION
$ ENDIF
$!
$! VERIFY SPECIFICATION FILE EXISTS
$!
$ FILE = F$SEARCH("REPORT$LOCATION:''P1'")
$ IF (FILE .EQS. "")
$ THEN
$     WRITE SYS$OUTPUT "BAD FILE SPECIFICATION"
$     GOTO GET_OUT
$ ENDIF
$!
$ WRITE SYS$OUTPUT "GENERATING REPORT FOR ''FILE'"
$!
$! CHECK IF BATCH REQUESTED:
$!
$ IF "''P5'" .EQS. "B"
$ THEN
$     if "''p2'".eqs."N" then p2 = " "
$     if "''p3'".eqs."N" then p3 = " "
$   SUBMIT/NOTIFY/NOPRINT/QUEUE='NODE'$BATCH -
 ORACLE$COM:GEN_BATCH_REPORTS.COM/-

PARAMETERS=('FILE','P2','P3','P4','P1','P7')/LOG=SYS$LOGIN:ORA_BATCH_REP
ORT.LOG
$   GOTO GET_OUT
$ ENDIF
$!
$! RPT FILES:
$!
$ IF F$INTEGER(P4) .EQ. 1
$ THEN
$ if "''p2'".eqs."N" then p2 = " "
$ if "''p3'".eqs."N" then p3 = " "
$ DEFINE/USE/NOLOG SYS$INPUT SYS$COMMAND
$ RPT REPORT$LOCATION:'P1' REPORT$DEST:'P1' 'P2'/'P3'
$ RPF REPORT$DEST:'P1' REPORT$DEST:'P1' -F
$ GOTO GET_OUT
$ ENDIF
$!
$! SQL*REPORTWRITER FILES:
$!
$ IF F$INTEGER(P4) .EQ. 2
$ THEN
$     if "''p2'".eqs."N" then p2 = " "
$     if "''p3'".eqs."N" then p3 = " "
$ IF "''P6'" .EQS. "" THEN P6 = "NO"
$ IF "''P6'" .EQS. "Y" THEN P6 = "YES"
$ IF "''P6'" .EQS. "N" THEN P6 = "NO"
$ DEFINE/USE/NOLOG SYS$INPUT SYS$COMMAND
$ RUNREP 'FILE' PARAMFORM='P6' USERID='P2'/'P3'
$ GOTO GET_OUT
$ ENDIF
$!
$! SQLPLUS REPORTS:
$!
$ IF F$INTEGER(P4) .EQ. 3
$ THEN
$     if "''p2'".eqs."N" then p2 = " "
$     if "''p3'".eqs."N" then p3 = " "
$ SQLPLUS -S 'P2'/'P3' @'FILE' 'P7'
$ GOTO GET_OUT
$ ENDIF
```

```
$!
$ WRITE SYS$OUTPUT "IMPROPER REPORT TYPE SPECIFIED"
$ GOTO GET_OUT
$!
$ GET_OUT:
$!
$ EXIT
```

Since this script calls GEN_BATCH_REPORTS.COM, we will look at it next.

```
$! GEN_BATCH_REPORTS.COM
$!
$! PROCEDURE TO ALLOW FOR THE GENERATION OF REPORTS FROM THE
$! DATABASES IN BATCH MODE
$!
$! REV. 0 12-MAY-1992 M. AULT
$!
$!  REV:    DATE:       REASON:                 PERSON:
$!  ----   --------     ---------------------  -------
$!   0     13-MAY-1992  INITIAL CREATE          M. AULT
$!
$!------------------------------------------------------------------
$! PARAMETERS:
$!
$! P1 : REPORT TO RUN
$! P2 : USERNAME (ORACLE)
$! P3 : PASSWORD (ORACLE)
$! P4 : TYPE OF REPORT 1-RPT, 2-REPORTWRITER, 3-SQL
$! P5 : FILE NAME FOR RPT REPORTS
$!__________________________________________________________________
$!
$! ASSUMPTIONS:
$!
$! 1. EACH NON-RPT REPORT MUST SPECIFY ITS OWN DESTINATION
$! 2. REPORT$LOCATION IS A LOGICAL POINTING TO THE DIRECTORY FOR THE
$!    REPORT SPECIFICATION FILE
$! 3. IF REPORT$LOCATION IS NOT SPECIFIED, IT WILL DEFAULT TO SYS$LOGIN
$! 4. IF REPORT$DEST IS NOT SET, IT WILL DEFAULT TO REPORT$LOCATION
$!------------------------------------------------------------------
$!
$ SET NOON
$!
$! RPT FILES:
$!
$ IF F$INTEGER(P4) .EQ. 1
$ THEN
$ DEFINE/USE/NOLOG SYS$INPUT SYS$COMMAND
$ RPT REPORT$LOCATION:'P1' REPORT$DEST:'P1' 'P2'/'P3'
$ RPF REPORT$DEST:'P1' REPORT$DEST:'P1' -F
$ GOTO GET_OUT
$ ENDIF
```

```
$!
$! SQL*REPORTWRITER FILES:
$!
$ IF F$INTEGER(P4) .EQ. 2
$ THEN
$ P6 = "NO"
$ DEFINE/USE/NOLOG SYS$INPUT SYS$COMMAND
$ RUNREP 'FILE' PARAMFORM='P6' USERID='P2'/'P3'
$ GOTO GET_OUT
$ ENDIF
$!
$! SQLPLUS REPORTS:
$!
$ IF F$INTEGER(P4) .EQ. 3
$ THEN
$ SQLPLUS -S 'P2'/'P3' @'FILE'
$ GOTO GET_OUT
$ ENDIF
$!
$ WRITE SYS$OUTPUT "IMPROPER REPORT TYPE SPECIFIED"
$ GOTO GET_OUT
$!
$ GET_OUT:
$!
$ EXIT
```

Once the report is ready, we will want to get a look at it, either in hard copy or on screen. PRINT_REPORTS.COM does this for us. This script is referenced by the PREP symbol in the ORACLE_LOG.COM procedure. Its listing follows.

```
$! PRINT_REPORTS.COM
$!
$! PROCEDURE TO ALLOW FOR THE PRINTING OF REPORTS FROM THE
$! DATABASES
$!
$! REV. 0 12-MAY-1992 M. AULT
$!
$!  REV:    DATE:       REASON:                PERSON:
$!  ----   --------     --------------------   -------
$!   0     12-MAY-1992  INITIAL CREATE         M. AULT
$!
$!-------------------------------------------------------------------
$! PARAMETERS:
$!
$! P1 : REPORT TO PRINT (FULL PATH UNLESS GENERATED IN DEFAULT
DIRECTORY)
$! P2 : WHERE TO PRINT REPORT (QUEUE NAME FROM MMD_QUEUES OR SPECIFY)
$!      "TERM" WILL DISPLAY TO TERMINAL
$! P3 : SIZE OF REPORTS LINES 1-80, 2-132 (DEFAULTS TO 80), 3->132
$!_________________________________________________________________
$!
$! ASSUMPTIONS:
$!
$! EACH REPORT IS ANSI FORMAT
$!
$!-----------------------------------------------------------------
$!
$ SET NOON
$ DEFINE/USE/NOLOG SYS$INPUT SYS$COMMAND
$!
$ IF F$TRNLNM("REPORT$LOCATION") .EQS. ""
$ THEN
$     DEFINE/NOLOG REPORT$LOCATION SYS$LOGIN
$ ENDIF
$!
$ IF F$TRNLNM("REPORT$DEST") .EQS. ""
$ THEN
$     DEFINE/NOLOG REPORT$DEST REPORT$LOCATION
$ ENDIF
$!
$ FILE = F$SEARCH("REPORT$DEST:''P1'")
$ IF (FILE .EQS. "")
$ THEN
$     WRITE SYS$OUTPUT "BAD FILE SPECIFICATION"
$     WAIT 00:00:03
$ GOTO GET_OUT
$ ENDIF
$!
$ IF "''P2'" .EQS. "TERM"
$ THEN
$ GOTO DISPLAY_REPORT
$ ENDIF
$!
$ QUE_TEST = F$GETQUI("DISPLAY_QUEUE","QUEUE_NAME","''P2'")
$ IF "''QUE_TEST'" .EQS. ""
$ THEN
$     WRITE SYS$OUTPUT "BAD QUEUE SPECIFICATION"
$     WAIT 00:00:02
$     GOTO GET_OUT
$ ENDIF
$!
$ WRITE SYS$OUTPUT "PRINTING REPORT ''FILE'"
$!
$ IF P3 .EQS. ""
$ THEN
$     P3 = "1"
$     WRITE SYS$OUTPUT "DEFAULTING TO 80 COLUMNS"
$     WAIT 00:00:02
$ ENDIF
$!
$ IF F$INTEGER(P3) .EQ. 1
$ THEN
$     PRINT/QUEUE='P2'/param=(numb=1) 'FILE'
$     GOTO GET_OUT
$ ENDIF
$!
$ IF F$INTEGER(P3) .EQ. 2 .OR. F$INTEGER(P3) .EQ. 3
$ THEN
$     PRINT/QUEUE='P2'/PARAMETERS=(numb=1,PAGE_O=LANDSCAPE) 'FILE'
$     GOTO GET_OUT
$ ENDIF
$!
$ WRITE SYS$OUTPUT "IMPROPER LINE SIZE SPECIFIED"
$ WAIT 00:00:03
$ GOTO GET_OUT
$!
$ DISPLAY_REPORT:
$ IF F$INTEGER(P3) .EQ. 2 .OR. F$INTEGER(P3) .EQ. 3
$ THEN
$     SET TERM/WIDTH=132
$ ENDIF
$ TYPE/PAGE 'FILE'
$ READ/PROMPT="PRESS ENTER WHEN FINISHED" SYS$COMMAND NULL_IN
$ set term/wid=80
$ GOTO GET_OUT
$!
$ GET_OUT:
$!
$ EXIT
```

Another form of this script is called PRINT_IT.COM and is used by some of the menus. The listing for PRINT_IT.COM follows.

```
$!
$! NAME     : PRINT_IT.COM
$! PURPOSE  : AUTOMATE THE PRINTING OF STATUS REPORTS FOR THE DBA
$! USE            : FROM DBSTATUS MENU
$! Limitations    : None
$! Revisions:
$!    Date        Modified By Reason For change
$!    07-AUG-1992 MIKE AULT   INITIAL CREATE
$!              23-OCT-1992   MIKE AULT   REPLACE FULLPATHS WITH
LOGICALS
$ START:
$ HOST = F$GETSGI("NODENAME")
$!
$ PRINT_IT:
$ purge/nolog ora_status_reports
$ READ/PROMPT = "DO YOU WANT A PRINTOUT?: " SYS$COMMAND YN
$ YN = F$EDIT(YN,"UPCASE")
$ IF "''YN'" .EQS. "Y"
$ THEN
$  READ/PROMPT = "ENTER QUEUE NAME: " SYS$COMMAND PRT
$  PRT = F$EDIT(PRT,"UPCASE")
$   PRINT/QUE='PRT'/PARAM=(num=1) ORA_status_reports:'P1'
$ ENDIF
$!
$ END:
$!
$ EXIT
```

As you can see, PRINT_IT.COM isn't nearly as sophisticated as PRINT_REPORTS.COM, but for non-critical applications it works just fine.

The final example script is one to do Hot Backups. This is just a bare bones script and lots of error checking and self protection can be added, but as a starting point it is sufficient

```
$!**********************************************************************
$! Name     : Hot_backup.com
$! Purpose  : Perform a hot backup of an Oracle Database
$! Use      : @Hot_backup.com
$! Limitations    : Creates a read consistent image, but doesn't backup
in process transactions
$!
$! Revision History:
$! Date           Who         What
$! --------       ----------- ----------------------------
$! June 1993      K. Loney    Featured in Oracle Mag. Article
$! 29-Jun-93      M. Ault     Modified, commented
$!**********************************************************************
$!
$! Define symbol for backup command
$ dup_it = "backup/ignore=(noback,interlock,label) /log"
$ !
$ SVRMGRL
     connect internal
     alter tablespace system begin backup;
     exit
$ dup_it m_ora_disk2:[m_oracle.oracle6.db_example]ora_system_1.dbs
   mua0:ora_system.bck/save
$!
 SVRMGRL
   connect internal
   alter tablespace system end backup;
   alter tablespace tools begin backup;
   exit
$ dup_it m_ora_disk3:[m_oracle.oracle6.db_example]ora_tools_1.dbs
  mua0:ts_tools.bck/save
$!
SVRMGRL
```

```
   connect internal
   alter tablespace tools end backup;
   alter tablespace user_tables begin backup;
   exit
$ dup_it m_ora_disk3:[m_oracle.oracle6.db_example]ora_user_tables_1.dbs
  mua0:ts_tools.bck/save
$!
   SVRMGRL
   connect internal
   alter tablespace user_tables end backup;
   exit
$! force write of all archive logs
$!
$ SVRMGRL
   connect internal
   alter system switch logfile;
   archive log all;
   exit
$!
$ rename m_ora_disk5:[m_oracle.oracle6.db_example.archives]*.arc
*.oldarc
$! Now backup a control file
$!
$ SVRMGRL
   connect internal
   alter database example
   backup controlfile to
   'm_ora_disk1:[m_oracle.oracle6.db_example]ora_control.bac
   reuse;
   exit
$ dup_it m_ora_disk1:[m_oracle.oracle6.db_example]ora_control1.con
mua0:ora_control.bac/save
$! now backup all archive logs
$!
$! you don't want to delete logs if an error causes them not to be
backed up
$ on error goto end_it
$!
$ dup_it m_ora_disk5:[m_oracle.oracle6.db_example.archives]*.oldarc
mua0:ora_archives.bck/save
$! Now delete logs
$!
$ delete/log
m_ora_disk5:[m_oracle.oracle6.db_examples.archives]*.oldarc;*
$ end_it:
$ exit
```

Using the example scripts shown, a DBA, perhaps with the assistance of the System Manager or someone familiar with DCL, can create their own menu system using DCL, SQL*menus and other utilities. A complete menu system based on the

MANAGE_ORACLE.COM procedure is available on the disk that is the companion to this book and comes with complete installation instructions.

UNIX Shell Scripts:

One of the first assignments I had after leaving the safe haven of being a house DBA and moving into consulting was the video on demand project that TeleTV out of Virginia put together. The pilot was in Rome, Italy on Sequent systems using N-Cubes as the video servers (I know, it was a tough job but somebody had to do it). Anyway, while the systems had good system management tools and good backup tools, they had no DBA support tools to speak of. Pulling on my vast UNIX knowledge (right) I put together a menu system to allow the execution, printing and viewing of DBA reports. Here it is, I hope you find it useful. At the end of the appendix I have also included a few shell scripts that do such things as kill Oracle operating side processes (other than core processes), allow grep and sed of the alert log for errors, list running instances and so forth.

```
#!/bin/sh
# dba_menu
# Oracle Database DBA Menu. Allows simple generation, viewing
# and printing of Oracle Database Reports and DBA activities
# Rev 0. 30/9/95 MRA - RevealNet
#
# first full path the report files
#
REP=/home/oracle/sql_scripts
#
# now set report output full path
#
REP_OUT=/home/oracle/sql_scripts/rep_out
SH_FILES=/home/oracle/sh_files
x="0"
while [ $x != "99" ]
do
#
tput clear
echo ""
echo ""
echo "                      Oracle Database DBA Menu"
echo "                   ------------------------------"
echo ""
echo "                    1. Generate a report, execute script"
echo "                    2. View a report"
echo "                    3. Print a report"
echo "                    4. See a reports function statement"
echo "                   5. grep Against the sessions report"
echo "                   6. Enter SQLPLUS"
echo "                   7. Enter SVRMGR (Must be from Oracle user)"
echo "                   8. tail the Alert log"
echo "                   9. Check SQLNet V1 TCPIP status"
echo "                  10. tail SQLNet V1 TCPIP log"
echo "                  11. Check SQLNet V2 TCPIP status"
echo "                  12. tail SQLNET V2 TCPIP log"
echo "                  13. Start/Stop SQLNET"
echo ""
echo "                  99. Exit menu"
echo ""
echo "                  -----------------------------"
echo ""
echo "                        Enter Choice: \c"
read x1
if [ -n "$x1" ]
      then
            x=$x1
fi
case $x in

1)
cd $REP
tput clear
rep_name="none.sql"
echo ""
echo ""
echo "Please enter the name of the report to run from the following
list:"
echo ""
echo
"********************************************************************* "
ls -C *.sql
echo
"********************************************************************* "
echo ""
echo "\n                      Enter choice: \c"
read rep_name2
        echo ""
        if [ -s "$rep_name2" ]
        then
                rep_name="$rep_name2"
        fi
echo "\n Enter Oracle user name to run report under : \c"
read orauser
stty -echo
echo "\n                 Enter Oracle password: \c"
read pw
stty echo
#
# get into SQLPLUS and run report, normally report will terminate
# automatically with output directed to REP_OUT
#
tput clear
sqlplus -s $orauser/$pw @$rep_name
#
;;
```

```
2)
cd $REP_OUT
tput clear
rep_name="none.lis"
echo ""
echo ""
echo "Please enter the name of the report to view from the following
list:"
echo ""
echo
"**********************************************************************"
ls -C
echo
"**********************************************************************"
echo ""
echo "\n                Enter choice: \c"
read rep_name2
            echo ""
      if [ -s "$rep_name2" ]
      then
            rep_name="$rep_name2"
      fi
       pg $rep_name
#
;;

3)
cd $REP_OUT
tput clear
rep_name="none.lis"
echo ""
echo ""
echo "Please enter the name of the report to print from the following
list:"
echo ""
echo
"**********************************************************************"
ls -C
echo
"**********************************************************************"
echo ""
echo "\n                Enter choice: \c"
read rep_name2
                echo ""
        if [ -s "$rep_name2" ]
        then
                rep_name="$rep_name2"
           lp $rep_name
      else
      pg $rep_name
        fi
echo ""
;;

4)
cd $REP
tput clear
```

```
rep_name="none.sql"
echo ""
echo ""
echo "Enter the name of the report to see the function for from
following list:"
echo ""
echo
"*********************************************************************"
ls -C *.sql
echo
"*********************************************************************"
echo ""
echo "\n Enter choice: \c"
read rep_name2
        if [ -s "$rep_name2" ]
        then
                rep_name="$rep_name2"
        fi
echo ""
grep -i FUNCTION $rep_name
echo "                  Press enter to continue"
read nada
;;

5)
$SH_FILES/grep_sessions
;;

99)
tput clear
exit
;;

6)
tput clear
echo "Enter Oracle User name for SQLPLUS session: \c"
read un
stty -echo
echo ""
echo "Enter Oracle Password: \c"
stty echo
read pw
sqlplus $un/$pw
;;

7)
svrmgrl
;;

8)
tput clear
$SH_FILES/ck_alrt
;;

9)
tput clear
tcpctl status>status.lis
pg status.lis
```

```
rm status.lis
;;

10)
tput clear
echo "Enter number of lines to display: \c"
read lines
tail -$lines /home/oracle/product/7.1.4.1.0/tcp/log/orasrv.log>tail.lis
pg tail.lis
rm tail.lis
;;

11)
tput clear
lsnrctl status>status.lis
pg status.lis
rm status.lis
;;

12)
tput clear
echo "Enter number of lines to display: \c"
read lines
tail -$lines
/home/oracle/product/7.1.4.1.0/network/log/listener.log>tail.lis
pg tail.lis
rm tail.lis
;;

13)
tput clear
echo " S - Start SQLNET V1 and V2, X - Stop SQLNET V1 and V2: \c"
read strt_stp
if [ $strt_stp = "s|S" ]
then
      tput clear
      echo " Has the re-start of the SQLNET protocols been authorized by
STREAM? (Y or N):\c"
      read yn
      if [ $yn = "y|Y" ]
      then
            /home/oracle/start_tcpctl
      fi
fi
if [ $strt_stp = "x|X" ]
then
      echo " Has the stop of SQLNET protocols been authorized by STREAM?
(Y or N): \c"
      read yn
      if [ $yn = "y|Y" ]
      then
            /home/oracle/stop_tcpctl
      fi
fi
;;
*)
tput clear
echo ""
echo "                        Invalid Selection try again"
echo "                        Press enter to continue"
read nada
pw="0"
;;
esac
pw="0"
done
```

The next shell script wasused to grep from the list of sessions generated by a sessions report, specific values. The script has been modified to grep a specific report in the listing that follows this one.

```
# grep_sessions shell script MRA
#!/bin/sh
yn=y
while [ yn -eq y or yn -eq Y ]
do
tput clear
echo "Enter value to grep the sessions report for:\c"
read search_value
echo "Enter name for output file from grep operation:\c"
read output_file
if [ -r report_output/$output_file ]; then
      echo "File exists - do you want to remove it? (Y or N):\c"
      read yn
      case $yn in
      y|Y)
            rm report_output/$output_file
            echo "File " $output " Removed"
      ;;

      n|N)
            echo "Enter a different file name for the output file:\c"
            read output_file
      ;;
      esac
fi
#
egrep -i "$search_value" rep_out/sessions.lst>report_output/$output_file
echo "Do you wish to see the result file using the page utility?(Y or
N)?:\c"
read yn
case $yn in
      y|Y)
            tput clear
            pg report_output/$output_file
      ;;

      n|N)
            tput clear
      ;;
      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac

echo "Do you want to remove the result file? (Y or N):\c"
read yn
case $yn in
      y|Y)
            tput clear
            rm report_output/$output_file
      ;;

      n|N)
            tput clear
      ;;

      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac

echo "Do you wish to exit the utility? (Y or N):\c"
read yn
case $yn in
      n|N)
            tput clear
            ;;

      y|Y)
            tput clear
            exit
      ;;
esac
done
```

Here is the above script made to grep any selected report output:

```
#!/bin/sh
cd /home/oracle/admin/imp/adhoc/report_output
yn=y
while [ yn -eq y or yn -eq Y ]
do
  tput clear
  rep_name="none.lst"
  echo ""
  echo ""
  echo "Please enter the the report to use from the following list:"
  echo ""
  echo "**************************************************************"
  ls -C *
```

```
echo "***************************************************************"
echo ""
echo "\n                               Enter choice: \c"
read rep_name2
      echo ""
      if [ -s "$rep_name2" ]
      then
              rep_name="$rep_name2"
      fi
#
tput clear
echo "Enter value to grep the report for:\c"
read search_value
echo "Enter name for output file from grep operation:\c"
read output_file
if [ -r $output_file ]; then
    echo "File exists - do you want to remove it? (Y or N):\c"
    read yn
    case $yn in
    y|Y)
          rm $output_file
          echo "File " $output " Removed"
    ;;

    n|N)
          echo "Enter a different file name for the output file:\c"
          read output_file
    ;;
    esac
fi
#
egrep -i "$search_value" $rep_name >$output_file
echo "View the result file using the page utility?(Y or N)?:\c"
read yn
case $yn in
    y|Y)
          tput clear
          pg $output_file
    ;;

    n|N)
          tput clear
    ;;

    *)
          echo "Invalid Response"
          echo "Press enter to continue"
          read nada
    ;;
esac
echo "Do you want to remove the result file? (Y or N):\c"
read yn
case $yn in
    y|Y)
          tput clear
          rm $output_file
    ;;
```

```
      n|N)
            tput clear
      ;;

      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
  esac
  echo "DO you wish to exit the utility? (Y or N):\c"
  read yn
  case $yn in
      n|N)
            tput clear
            ;;

      y|Y)
            tput clear
            exit
      ;;
  esac
  done
```

The next shell script, called ck_alrt does a user specified tail of the alert log. The script assumes the alert log (or at least a link to the actual alet log) is stored in the default location $ORACLE_HOME/dbs. If like me, you locate the alert logs in a more instance specific location, just stick a soft link (ln -s) to the actual location for each alert log into the default location.

```
#ck_alrt script to tail last n lines of alert log
#!/bin/sh
tput clear
echo "Enter sid of database(uppercase):\c"
read db_sid
tput clear
echo "Enter numer of lines to display:\c"
read line_no
tput clear
tail -$line_no $ORACLE_HOME/dbs/alert_$db_sid.log|more
```

The next script produces a list of the lines that contain a specified text fragment from the alert log. The script uses grep so only the line with the text is displayed. A more advanced version is shown later which uses sed to also capture the line above and below the line with the error or text. If the script is called grep_alert then usage would be:

$ grep_alert alert_location

Where alert_location is the full path relative to the current location, relative to root or as a symbol to the alert log. I ususally specify a set of symbols that point to the alert logs on the system and just use the symbols.

```
#!/bin/ksh
cd /home/oracle/mon_scripts/report_output
yn="y"
while [ $yn = "y" or $yn = "Y" ]
do
tput clear
#tput clear
echo "Enter value to grep the log for:\c"
read search_value
echo "Enter name for output file from grep operation:\c"
read output_file
if [ -r $output_file ]; then
      echo "File exists - do you want to remove it? (Y or N):\c"
      read yn
      case $yn in
      y|Y)
            rm $output_file
            echo "File " $output_file " Removed"
      ;;
      n|N)
            echo "Enter a different file name for the output file:\c"
            read output_file
      ;;
      esac
fi
#
egrep -i "$search_value" $1 >$output_file
echo "Do you wish to see the result file using the page utility?(Y or
N)?:\c"
read yn
case $yn in
      y|Y)
            tput clear
            pg $output_file
      ;;
      n|N)
            tput clear
      ;;
      *)
```

```
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac
echo "Do you want to remove the result file? (Y or N):\c"
read yn
case $yn in
      y|Y)
            tput clear
            rm $output_file
      ;;
      n|N)
            tput clear
      ;;
      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac
echo "Do you wish to exit the utility? (Y or N):\c"
read yn
case $yn in
      n|N)
            tput clear
            ;;
      y|Y)
            tput clear
            exit
      ;;
esac
done
```

The next shell script creates a set of process kill commands for each database on the machine for non-oracle base processes. This can be useful for doing unattended backups where users aren't exactly disciplined to log out as they should. Killing from the Operating system side usually does a better cleanup than from the Oracle side.

```
#!/bin/ksh
ORATAB=/etc/oratab
trap 'exit' 1 2 3
# Set path if path not set (if called from /etc/rc)
case $PATH in
    "")     PATH=/bin:/usr/bin:/etc
      export PATH ;;
esac
#
# Loop for every entry in oratab
#
rm kill.lis
rm proc.lis
touch kill.lis
touch proc.lis
```

```
cat $ORATAB | while read LINE
do
    case $LINE in
      \#*)        ;;    #comment-line in oratab
      *)
      ORACLE_SID=`echo $LINE | awk -F: '{print $1}' -`
    if [ "$ORACLE_SID" = '*' ] ; then
            ORACLE_SID=""
    fi
       esac
       if [ "$ORACLE_SID" <> '*' ] ; then
            proc_name='oracle'$ORACLE_SID
            ps -ef|grep $proc_name>>proc.lis
       fi
done
cat proc.lis | while read LINE2
do
      command=`echo $LINE2 | awk -F: 'BEGIN { FS = ",[ \t]*|[ \t]+" }
                                { print $2}' -`
        test_it=`echo $LINE2 | awk -F: 'BEGIN { FS = ",[ \t]*|[ \t]+" }
                                 { print $8}' -`
      if [ "$test_it" <> 'grep' ] ; then
            command='kill -9 '$command
            echo $command>>kill.lis
      fi
done
rm proc.lis
```

This next script uses sed to find a specific user input section of text in the alert log (specidied at the command line as argument 1 to the script). If the script is called sed_alert, then the method to use it would be:

$ sed_alert alert_location

Where alert_location is the full path (relative to this directory, root, or using a symbol) to the alert log you wantto examine including the alert log name. I usually set up a set of symbols, one for each alert log, then just use the symbol when I call the script. The script lists the lines surrounding the text as well as the line with the text.

```
#!/bin/ksh
# sed_alert script to list info from alert log
#
cd /users/oracle/monitor_scripts/report_output
yn="y"
```

```
while [ $yn = "y" or $yn = "Y" ]

do
tput clear
#
tput clear
echo "Enter value to search the log for:\c"
read search_value
echo "Enter name for output file from sed operation:\c"
read output_file1
output_file2=$output_file1"2"
if [ -r $output_file1 ]; then
      echo "File exists - do you want to remove it? (Y or N):\c"
      read yn
      case $yn in
      y|Y)
            rm $output_file1
            rm $output_file2
            echo "File " $output_file1 " Removed"
      ;;
      n|N)
      echo "Enter a different file name for the output file:\c"
      read output_file
      ;;
      esac
fi
#
touch output_file2
sed -n "/$search_value/=" $1 >$output_file1
cat $output_file1|while read LINE
do
      num1=`echo $LINE | awk -F: '{print $1}' - `
      num2=`echo  ${num1}+3 | bc -l`
      sed -n "$num1,$num2 l" $1 >>$output_file2
done
echo "Do you wish to see the result file using the page utility?(Y or
N)?:\c"
read yn
case $yn in
      y|Y)
            tput clear
            pg $output_file2
      ;;
      n|N)
            tput clear
      ;;
      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac
echo "Do you want to remove the result file? (Y or N):\c"
read yn
case $yn in
      y|Y)
            tput clear
            rm $output_file1
            rm $output_file2
      ;;
      n|N)
            tput clear
      ;;
      *)
            echo "Invalid Response"
            echo "Press enter to continue"
            read nada
      ;;
esac
echo "Do you wish to exit the utility? (Y or N):\c"
read yn
case $yn in
      n|N)
            tput clear
            ;;
      y|Y)
            tput clear
            exit
      ;;
esac
done
```

The next script produces a list of the databases that are currently running by grepping a process list into a temporary file and then using awk to test for, and strip out, the database sid. Since there should only be one smon process per running database this should work (it has so far....).

```
      rm -f proc.lis
      rm -f dbs.lis
      touch dbs.lis
      touch proc.lis
      ps -ef|grep smon>>proc.lis
      cat proc.lis | while read LINE2
      do
       command=`echo $LINE2 | awk -F: 'BEGIN { FS = ",[ \t]*|[ \t]+" }
                        { print substr($9,10)}' -`
       test_it=`echo $LINE2 | awk -F: 'BEGIN { FS = ",[ \t]*|[ \t]+" }
                        { print substr($8,1)}' -`
            if [ "$test_it" != "grep" ] ; then
                                command='Database: '$command' is up'
                                echo $command>>dbs.lis
            fi
      done
      cat dbs.lis
```

The next script was created to perform remote exports, that is, export databases on remote nodes from a central server. It uses some interestig techniques so I thought I would include it. The script depends on a file set up similar to the oratab file only the fields are:

filesystem name:freespace required:parameter file to use:export file name:log directory:sid:YorN

The parameter file user/password entry should have the connection string as well, so if the tnsnames alias for the database to be exported was ORTEST1 then the userid entry would be:

userid= system/manager@ORTEST1

```
# File: REMOTE_EXP */
# Purpose: This file will automatically perform an export */
#         of a database that resides on a remote server. */
# */
# Setup environment */
# ORATAB is the name of the list of sids and parameters
# */
ORATAB=EXPTAB
OLD_IFS=$IFS
IFS=:
# */
# Start outermost loop through EXPTAB entries */
# */
cat $ORATAB | while read LINE
do
case $LINE in
        \#*) # ignore # lines
        ;;
        99) exit ;;
        *:Y) # Proceed if last field is Y #
# */
# Extract first set of variables into x array */
# */
        set -A x $LINE
# */
# Extract variables from x array */
# */
        filesys=${x[0]}
        freespc=${x[1]}
        pfile=${x[2]}
        expdir=${x[3]}
        logdir=${x[4]}
        orasid=${x[5]}
# */
# Set other needed variables */
# */
        dt_time=`date '+%y%m%d'`
```

```
          exit_status=1
          re_try=1
# */
# Cleanup old logs and create new one */
# */
# First cleanup using find on the log file location */
# */
          find $logdir -name 'exp_log*.log' -atime +7 -exec rm {} \;
# */
# Now create new log file */
# */
          fil_cnt=$(($(ls -C1 $logdir'exp_log_'$dt_time*.log|wc -l)+1))
          log=$logdir'exp_log_'$dt_time'_'$fil_cnt.log
          touch $log
          echo 'Start export of '$orasid' on '`date'+%y%m%d%H:%M:%S'`>>$log
# */
# Assign some local file name variables to enhance readability */
# */
          explog=$logdir$orasid'_'$dt_time.log
          expfile=$expdir$orasid'_'$dt_time.exp
          echo 'Export filename: '$expfile.Z>>$log
# */
# Check the number of exports and delete oldest */
# */
          if (($(($(ls $expdir|grep *exp.Z|wc -l)>3)))); then
                    ls -ctr -C1 $expdir$orasid*.exp.Z|read del_file
                    echo 'Removing '$del_file>>$log
                    rm $del_file
          fi
# */
# Check file space, set second value in EXPTAB to appropriate integer */
# */
          if (($(($(df|grep $filesys |cut -c47-54)-$freespc>0)))); then
# */
# Set up dual while loop around entire export and error checking section */
# */
          while [ ${exit_status} -ne 99 ]
          do
                    while [ ${exit_status} -eq 1 ]
                    do
                    $ORACLE_HOME/bin/exp parfile=$pfile FILE=$expfile LOG=$explog
# */
                    compress -f $expfile
# */
# Check for errors in log, if found process */
# For fatal, non-recoverable errors, just list to error log and exit */
# For self correcting errors, retry at 1 hour intervals max of 3 times, */
# then log error and exit. */
# For informational errors, log error and exit. */
# All errors begin with exp- or ora-, each stack will have at least 1 exp- */
# error, since there are only 36 exp- errors, we will use them for */
# processing */
# */
          if (($(grep -F -i -e'ora-' -e'exp-' $explog|wc -l)>0)); then
```

```
                grep -F -i -e'ora-' -e'exp-' $explog>temp.lis
                if (($(grep -F -i -ffatal_er.lis temp.lis|wc -l)>0)); then
                        echo 'Fatal error has occured, terminating'>>$log
                        cat temp.lis>>$log
                        exit_status=99
                fi
                if [ ${exit_status} -ne 99 ]; then
                 if (($(grep -F -i -fretry_er.lis temp.lis|wc -l)>0)); then
                        echo 'A non-fatal error has occured, retrying'>>$log
                        cat temp.lis>>$log
                        rm $expfile.Z
                        re_try=$(($re_try+1))
                        exit_status=1
                        echo 'Retry number: '$re_try>>$log
                        if [ ${re_try} -eq 4 ]; then
                                exit_status=99
                        else
                                sleep 3600
                        fi
                 fi
                fi
                if [ ${re_try} -eq 1 ]; then
                 if [ ${exit_status} -ne 99 ]; then
                  if (($(grep -F -i -fignore_e.lis temp.lis|wc -l)>0)); then
                        echo 'An error  considered a warning has occured'>>$log
                        echo 'Check database tables and connections'>>$log
                        cat temp.lis>>$log
                        exit_status=99
                  fi
                 fi
                fi
        rm temp.lis
        else
                exit_status=99
        fi
        done
done
else
echo 'Not enough file space to perform export'>>$log
fi
;;
esac
cat $explog >>$log
rm $explog
echo 'Ending of '$orasid' export at '`date '+%y%m%d%H:%M:%S'`>>$log
done
IFS=$OLD_IFS
```

Here is the hot backup script from the book:

```
#*************************************************************************
# Name      : hot_backup
# Purpose   : Perform a hot backup of an Oracle Database
# Use : sh hot_backup
# Limitations     : Creates a read consistent image, but doesn't backup
#                 : in process transactions except by archive log
#
# Revision History:
# Date            Who         What
# --------------  ----------- ------------------------------
# June 1993       K. Loney    Featured in Oracle Mag. Article
# 29-Jun-93       M. Ault     Modified, commented
# 02-Aug-93       M.Ault      Converted to UNIX script
# 03-Aug-93       M. Phillips Added error detection
#**********************************************************************
#
ERROR="FALSE"
LOGFILE="$ORACLE_HOME/adhoc/scripts/hot_back_log"
while [ "$error"=FALSE ]
do
svrmgrl << ending1
      connect internal
      alter tablespace system begin backup;
      exit
ending1
      if ( tar cfv /oracle/backup /data/ORA_SYSTEM_1.DBF )
      then
           :
      else
          ERROR="TRUE";
            echo "Tar backup failed for ora_system1.dbf" >$LOGFILE
      fi
svrmgrl << ending2
connect internal
      alter tablespace system end backup;
      exit
ending2

dup_it="tar rv /oracle/backup"
svrmgrl << ending3
      connect internal
      alter tablespace user_tables begin backup;
      exit
ending3
if ( $dup_it /data/ora_user_tables_1.dbf )
then
      :
else
    ERROR="TRUE";echo "Tar backup failed for
ora_user_tables_1.dbf">>$LOGFILE
fi #we must still end backup for tablespaces
svrmgrl << ending4
      connect internal
      alter tablespace user_tables end backup;
      exit
ending4
# force write of all archive logs
```

```
svrmgrl << ending5
      connect internal
      alter system switch logfile;
      archive log all;
      exit
ending5
if ( cp /usr/oracle/oracle7/db_example.archives/*.arc *.oldarc )
then
    :
else
    ERROR="TRUE";echo "Copy of archive logs failed">>$LOGFILE
fi
# Now backup a control file
svrmgrl << ending6
      connect internal
      alter database example
      backup controlfile to
      '/usr/oracle/oracle7/db_example/ora_control.bac
      reuse;
      exit
ending6
if ( $dup_it /usr/oracle/oracle7/db_example/ora_control.bac )
then
    :
else
    ERROR="TRUE";echo "Tar backup failed for control file">>$LOGFILE
fi
# now backup all archive logs
if ( $dup_it /usr/oracle/oracle7/db_example.archives/*.oldarc )
then
    :
else
    ERROR="TRUE";echo "Tar backup failed for archive files">>$LOGFILE
fi
# Now delete logs
if ( rm /usr/m_oracle/oracle7/db_examples.archives/*.oldarc;* )
then
    ERROR="TRUE"
else
    ERROR="TRUE";echo "Delete of archive files failed">>$LOGFILE
fi
done
exit
done
```

The final UNIX command script is just a convienience, the tail -f command can be used to follow alert logs, listener logs and such. I like to set up a couple of small one line scripts that do just that, follow the input file. If I have the luxury of an X window terminal, I do multiple login screens and have a tail running on the alert log for each monitored instance, that way at a glance I can see if there are any problems. The script, like I said all one line of it, looks like this:

```
# fol_alrt
#!/bin/sh
tail -f $1
```

The $1 is the symbol pointing to the alert log (or any log file) to follow. For example:

```
ORTEST1_ALERT= $ORACLE_HOME/dbs/alert_ORTEST1.log

$ fol_alrt $ORTEST1_ALERT
```