Quick Tips File

Last Updated 25-JULY-07

UNIX Tips

Cron Jobs

If you want to see all your cron jobs just type *crontab -l*
If you want to edit a cron jobs just type *crontab -e*
Lines that begin with "#" are considered comments and are ignored.
An environment setting is of the form
NAME = VALUE
Cron will look at a special environment variable, MAILTO. Any output generated by your cron jobs will be sent to the address specified by MAILTO (if it is not specified it will be sent to the owner of the crontab). If MAILTO is defined by empty (MAILTO=""), no mail will be sent.
The format of the cron table entry includes five (5) time fields followed by a command. Commands are executed when the time specified by the date fields matches the current time. The five time fields are as follows:

```
Field        Allowed Values
minute         0-59
hour           0-23
day of month   0-31
month          1-12 (or names, see below)
day of week    0-7 (0 or 7 is Sun, or use names)
```

A field may be an asterisk (*), which indicates all values in the range are acceptable. Ranges of numbers are allowed, i.e. "2-5" or "8-11", and lists of numbers are allowed, i.e. "1,3,5" or "1,3,8-11". Step values can be represented as a sequence, i.e. "0-59/15", "1-31/3", or "*/2".
Names can be used for the "month" and "day of week" fields. Use the first three letters (case-insensitive) of the particular month or day. Ranges or lists of names are not allowed.
Examples:
MAILTO="someone@somewhere.com"
15 1 * * * [COMMAND]
Explanation: executes the command [COMMAND] at 1:15 AM every day

40 23 1 * * [COMMAND]
Explanation: executes the command [COMMAND] on the first of every month at 11:40 PM

0-30/10 9,17 * * 1-5 [COMMAND]
Explanation: executes the command [COMMAND] on Monday-Friday (1-5) every 10 minutes for the first half hour (0-30/10) of the 9 AM and 5 PM hours (9,17)

0 */4 * jan sun [COMMAND]
Explanation: executes the command [COMMAND] on each Sunday in January at midnight, 4 AM, 8 AM, noon, 4 PM, and 8 PM

30 4 1,15 * fri [COMMAND]
Explanation: executes the command [COMMAND] at 4:30 AM on the 1st and 15th of each month, plus every Friday

0 0 19 8 * [COMMAND] or
0 0 19 aug * [COMMAND]

esta es una linea de ejemplo:
00 20 * * * /usr/bin/find / -name core -exec rm -f {} \;
minute hour monthday month weekday command

Deleting Files by 'n' Dates

There are three times associated with a file
- atime - last access time
- ctime - last status change time
- mtime - last modify time

To remove all files from directory /home/dpafumi that have ctime greater than 5 days, enter
find /home/dpafumi -type f -ctime +5 -exec rm {} \;

To test this, first use something like the above command to print out what files have a ctime greater than 5 days with
find /home/dpafumi -type f -ctime +5 -print

Both commands will go down recursively through subdirectories of /home/dpafumi.

To only go search /home/dpafumi, you have to use the GNU version of find
/usr/local/bin/find /home/dpafumi -type f -ctime +5 -maxdepth 1 -print

or

gnufind instead of find (depends on the system)

Switching ORACLE_HOME and ORACLE_SID in Unix.

Have you ever had the need to change your environment settings to connect to another database on the same server? The following Unix shell script will ask you which database you want to connect to, and export your new environment variables. You may need to add additional variables, depending on your system and application.

```
#!/bin/ksh
echo " "
echo "Enter the Oracle database you want: "
echo " "
echo " 1. PROD 8.0.5.2.1 "
echo " "
echo " 2. TEST 8.1.6.0.0 "
echo " "
echo " 3. DEV 7.3.4.1.0 "
echo " "
read TermType
case $TermType in
1) export ORACLE_SID=prod
export ORACLE_HOME=/u01/app/oracle/product/8.0.5
export LD_LIBRARY_PATH=/u01/app/oracle/product/8.0.5/lib;;
2) export ORACLE_SID=test
export ORACLE_HOME=/u01/app/oracle/product/8.1.6
export LD_LIBRARY_PATH=/u01/app/oracle/product/8.1.6/lib;;
3) export ORACLE_SID=dev
export ORACLE_HOME=/u01/app/oracle/product/7.3.4
export LD_LIBRARY_PATH=/u01/app/oracle/product/7.3.4/lib;;
esac
echo
echo Setting ORACLE_SID to $ORACLE_SID
echo Setting ORACLE_HOME to $ORACLE_HOME
```

```
echo Setting LD_LIBRARY_PATH to $LD_LIBRARY_PATH
echo
echo
```

- How to find O/S VERSION or system name

uname -ap

- Different uses of 'ps -ef'

ps -ef | grep smon &nb sp; (Checks for databases)
ps -ef | grep ows (Checks fo r web listeners)
ps -ef | grep f45 (Checks f or Form listeners)

- Finding process id for trace or other process

ps -fu username

- Zipping up a directory for backup

1. zip -r file [Directory], or
2. zip -r file *fresh* (This would zip up all files with fresh in the name, plus any directories, and all dirs underneath with fresh in the name.

- How to tar a directory

tar -cvf /tmp/filename *
do this from the top direct ory that you want to bundle into the tar'd file

- using 'tar' to list contents of tar file, without extracting

tar -tvf [filename]

SETUP Display

If you have problems with the display, try the following in your startup file:

```
if ( $?DISPLAY == 0 ) setenv DISPLAY `who -m | cut -f 2 -d "(" | cut -f 1 -d ")"`:0.0
```

- To copy a directory structure (Directory, sub-directories, and all their files)

cp -rp

- Checking space on a disk

df -tk

- Show size of directories (All files under a specific directory)

du -sk

- Useful grep options

grep -c *This counts the number of lines that contain the pattern*
grep -i *Ignore upper/lower case*
grep -l *Print only the name of the files that contain the pattern. Does not repeat file names.*

- Count the number of files in a directory

ls | wc -l

- How to send a message to everyone on a UNIX machine

1. Create a file that contains the message
2. $ wall < [filename] (Ex: wall < message)
3. You may need to specify the directory 'wall' is in. /usr/sbin/wall < message

- Removing some files to free up space

1. Go to the destination directory
2. Run 'find . -name "*.*O" -print'. Make sure this returns files you want to delete
3. Run 'find . -name "*.*O" -exec rm -f {} \;

WARNING! Make sure you do this right or you can remove more files that you want.

Find Memory in CPU

/usr/sbin/prtconf
or
dmesg | more

- To find the printers available on a server

lpstat -p

- rcp: Setting up UNIX system to do a remote copy from another box

1. Create or modify file '.rhosts' in your login's home directory
2. Add a line such as...
   machinename.us.oracle.com loginname

Example
apptest15.us.oracle.com applmgr

3. telnet to the machine, and login as user specified in .rhosts
4. issue following command:
   rcp -r * login_to_your_machine@yourmachine:/directory_to_copy_to

example:
rcp -r * stuck@apptest9:/u01/stuck/

- How to do a search and replace in vi

:%s,[string to replace], [replacing string], g
example: :%s,oracle,stuck,gc {would replace oracle with stuck}

The above command does this for the whole document without confirming, if you would like to confirm each change use the command:
:%s,[string to replace], [replacing string], gc

This will stop after each search, type a y to confirm the change

- Stripping control character (^M) from an ascii file

Sometimes, in a patch, you will receive a file that was created in NT, so on Solaris it has a ^M at the end of each line. This can cause problems running the script sometimes. To strip out the ^M use the following

cat filename | tr -d ^M > newfilename

NOTE: When typing the ^M in the command above use Cntl V Cntl M to type it

- Cleaning up memory and semaphores when a db process is killed, rather than being shut down via svrmgrl

If a database process id is killed, rather than being shutdown properly via svrmgrl, then it will leave memonry and semaphores,
which may prevent the db from being recreated. Do the following to clean it up.

1. Run the UNIX command, 'ipcs -a'. You may want to stretch your screen as wide as it will go to get all the data returned to be on one line.
2. Look for the line where the value for the column NATTCH is 0. This is the Memory for your killed process that you need to delete.
3. Get the id# for that line from the ID column (first column)
4. Run the command 'ipcrm -m id#' to delete this memory

5. Figure out which line in the Semaphores zone is for your database.
If there are multiple lines, you may be able to determine which one by comparing the value in the NSEMS column to value of 'processes=XX' in your init.ora.
If only one line matches, then that is the line you need to delete.
If you delete the wrong line you will crash someone else's database.
If you cannot find a distinct line, you may want to try shutting down databases on the system until you can get one distinct line.
Once the line is identified, you need to remove it also.
6. Again, get the id# for that line from the first column.
7. Run 'ipcrm -s id#

- To lock archive file for updating

co -l tele.html

- To unlock

ci -u tele.html

Performing Loop Actions with Files

- Deleting Trace Files

```
for FILE in *.trc
do
  echo $FILE
  rm $FILE
done
```

- Rename Files

```
OLDSUFFIX=aaa
NEWSUFFIX=bbb
for FILE in *."$OLDSUFFIX"
do
  NEWNAME=´echo "$FILE" | sed -e "s/${OLDSUFFIX}\$/$NEWSUFFIX/"´
  mv "$FILE" "$NEWNAME"
done
```

- Set my filenames to lowercase

```
for FILE in *
do
  mv -i "$FILE" ´echo "$FILE" | tr '[A-Z]' '[a-z]'´ 2> /dev/null
done
```

Run Programs in background even if you disconnect

As a DBA, you have more than likely been faced with the need to run a command in UNIX that you know will take a long time to complete. Most often, you will want to run the command and "exit" from the terminal. When the shell exits though, it sends its children a SIGHUP signal, which by default causes them to be killed. All stopped, running and background jobs will ignore SIGHUP and continue running if their invocation is preceded by the nohup or if the process programmatically has chosen to ignore SIGHUP.
The nohup utility invokes the named command with the arguments supplied. When the command is invoked, nohup arranges for the SIGHUP signal to be ignored by the process.
Syntax

```
/usr/bin/nohup command [ argument ...]
```

Processes run by /usr/bin/nohup are immune to SIGHUP (hangup) and SIGQUIT (quit) signals.

Examples:
To run a command in the background after you log off, enter:

$ nohup find / -print &

To run a command in the background and redirect the standard output to a different file, enter:
$ nohup find / -print >filenames &
This example runs the find / -print command and stores its output in a file named filenames.Now only the process ID and prompt are displayed:
677
$
Wait before logging off because the nohup command takes a moment to start the command specified by the Command parameter. If you log off too quickly, the command specified by the Command parameter may not run at all. Once the command specified by the Command parameter starts, logging off does not affect it.

- Finding the IP address for different systems

grep -i finsun /etc/hosts

- How to find the tcpip address for a machine

nslookup [machine name]

- Different examples of the FIND command

find . -exec grep -lis fdugpi.lpc \{\} \\;
(searches in this directory down)

find / -exec grep -lis fdugpi.lpc \{\} \\;
(searches from root directory down)

find . -exec grep -lis file {} \;
(searches this directory down)

find . -follow -name bug734234 -print
(use if there are linked sub-directories, such as in tcpatch)

find . -name "*.pls" -exec egrep -il ARP_STANDARD {} \;
(This searches all .pls files for a file that contains the string ARP_STANDARD)

Install or Application Setup Tips

- Checking OS block_size. Oracle Block_Size must be equal or multiple of this one

df -g

- Pin Objects

1- As Internal Run:
@dbmspool.sql
@prvtpool.plb

2-Create the following Trigger

```
CREATE OR REPLACE TRIGGER db_startup_keep
AFTER STARTUP ON DATABASE
BEGIN
  sys.dbms_shared_pool.keep('SYS.STANDARD');
  sys.dbms_shared_pool.keep('SYS.DBMS_STANDARD');
  sys.dbms_shared_pool.keep('SYS.DBMS_UTILITY');
  sys.dbms_shared_pool.keep('SYS.DBMS_DESCRIBE');
  sys.dbms_shared_pool.keep('SYS.DBMS_OUTPUT');
END;
```

3- The following Oracle core packages owned by user SYS should be pinned in the shared PL/SQL area:

DUTIL
STANDARD
DIANA
DBMS_SYS_SQL
DBMS_SQL
DBMS_UTILITY
DBMS_DESCRIBE
DBMS_JOB
DBMS_STANDARD
DBMS_OUTPUT
PIDL

4- Run the following Script to check pinned/unpinned packages

```
SELECT substr(owner,1,10)||'.'||substr(name,1,35) "Object Name",
             '  Type: '||substr(type,1,12)||
             '  size: '||sharable_mem ||
             ' execs: '||executions||
             ' loads: '||loads||
             ' Kept: '||kept
  FROM v$db_object_cache
  WHERE type in ('TRIGGER','PROCEDURE','PACKAGE BODY','PACKAGE')
--   AND  executions > 0
  ORDER BY executions desc,
         loads desc,
         sharable_mem desc;
```

How to delete a database midway through creation

1. Shutdown DB
2. delete lk.. and control files from $ORACLE_HOME/
3. delete log and dbf files
4. recreate db

Start DB, Listener at Boot Time

To make the database and listeners start up automatically when the server reboots and shut down automatically when the server shuts down, you'll need to create a **dbora** file in /etc/init.d and link it to /etc/rc2.d and /etc/rc0.d. You'll need to do this as the root user. First create a file called dbora in /etc/init.d as follows as root:
vi /etc/init.d/dbora

```
#!/bin/sh
# description: Starts and stops Oracle processes
```

```
#
ORA_BASE=/u01/app/oracle
ORA_HOME=/u01/app/oracle/OraHome_1
ORA_OWNER=oracle
if [ ! -f $ORA_HOME/bin/dbstart ]
then
  echo "Oracle startup: cannot start"
  exit
fi
case "$1" in
     'start')
          # Start the Oracle database, listener and Web Control
          su - $ORA_OWNER -c "$ORA_HOME/bin/dbstart"
          su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl start"
          su - $ORA_OWNER -c "$ORA_HOME/bin/emctl start dbconsole"
          ;;
     'stop')
          # Stop the Oracle database, listener and Web Control
          su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl stop"
          su - $ORA_OWNER -c "$ORA_HOME/bin/emctl stop dbconsole"
          su - $ORA_OWNER -c $ORA_HOME/bin/dbshut
          ;;
esac
```

Link the script:

```
ln -s /etc/init.d/dbora /etc/rc2.d/S99dbora
ln -s /etc/init.d/dbora /etc/rc0.d/K10dbora
```

Test the script created. To test the script created above, without rebooting do the following:
su - root
/etc/init.d/dbora start (for startup)
/etc/init.d/dbora stop (for shutdown)

- Umask and Permission

You need to change the "umask" to the required 022 and set the permissions on all relevant directories.
Example:

umask 022
cd $ORACLE_HOME
chmod 755 (on all sub-dirs)
cd $ORACLE_BASE/admin/(SID NAME)
chmod 755 (on all sub-dirs)

- TWO_TASK to another database

1. Find Values from system you are connecting to...
   echo $ORACLE_SID, echo $FNDNAM, echo $GWYUID
   from the system you will be joining to.
2. Set Environment Variables
   TWO_TASK=[net_alias]; export TWO_TASK
   FNDNAM=value from $FNDNAM; export FNDNAM
   GWYUID=value from $GWYUID; export GWYUID

- Listener

Starting
lsnrctl start [Listener name from listener.ora]
Stopping
lsnrctl stop [Listener name from listener.ora]
Stopping it will not affect other users that have already connected, because their connection has been established.

- To find a PORT that is available when installing...
netstat -a | grep port#
ex.
netstat -a | grep 8068

- Recreating a DB from Backup and Changing the database name
Following is an example of how to modify the recontrol script:
The key is to use the resetlogs command, otherwise it will expect the original database name to be used when it reads it from the .dbf files

```
STARTUP NOMOUNT
CREATE CONTROLFILE
SET DATABASE "FRESHUPG"
  MAXLOGFILES 16
  MAXLOGMEMBERS 2
  MAXDATAFILES 30
  MAXINSTANCES 1
  MAXLOGHISTORY 100
LOGFILE
  GROUP 1 '/u04/oracle/dbf/FRESHUPG/log/log1.dbf' SIZE 1M,
  GROUP 2 '/u04/oracle/dbf/FRESHUPG/log/log2.dbf' SIZE 1M,
  GROUP 3 '/u04/oracle/dbf/FRESHUPG/log/log3.dbf' SIZE 1M resetlogs
DATAFILE
  '/u04/oracle/dbf/FRESHUPG/system.dbf',
  '/u04/oracle/dbf/FRESHUPG/ROLLBACK.dbf',
  '/u04/oracle/dbf/FRESHUPG/aolm.dbf',
  '/u04/oracle/dbf/FRESHUPG/aoli.dbf',
  '/u04/oracle/dbf/FRESHUPG/modm.dbf',
  '/u04/oracle/dbf/FRESHUPG/modi.dbf',
  '/u04/oracle/dbf/FRESHUPG/temp.dbf';
RECOVER DATABASE
ALTER DATABASE OPEN RESETLOGS;
```

Change The Owner Of Oracle Database Software On Unix

1) Shutdown the database.
2) Shut down all processes running out of the ORACLE_HOME. (Listener, EM etc.)
3) There are some files under Oracle Home which are owned by “root” or other users. For Example :
The file ‘nmo’, ’nmb’ , ‘oradism’ etc ... in the ORACLE_HOME/bin directory are owned by “root” user in 10g.In 9i the list may be different.

We can execute the following command to find the files, which are not owned by oracle owner

```
$ cd $ORACLE_HOME
$ find . ! -user <oracle_owner> | xargs ls -l | tee /tmp/list.txt
```

(Replace the <oracle_owner> with the current owner of installations)

For Example:

```
$ find . ! -user oracle | xargs ls -l | tee /tmp/list.txt
```

```
-rwsr-x--- 1 root root 65319 Jan 8 19:00 ./bin/extjob
-rwsr-s--- 1 root root 18801 Jan 8 18:47 ./bin/nmb
-rwsr-s--- 1 root root 19987 Jan 8 18:47 ./bin/nmo
-r-sr-s--- 1 root root 14456 Feb 4 2006 ./bin/oradism
-rw-r----- 1 root root 1534 Dec 22 2005 ./rdbms/admin/externaljob.ora
```

Check and make a note of the ownership of the files, which are listed as the result of the above command. The list will be also redirected to the file /tmp/list.txt.

4) Change the ownership of oracle database software by using recursive “chown” command.
Consider if the current owner of oracle software is "oracle92" who is from "dba" group, you want to change it to a new user "oracle" and your oracle software is installed in the directory "/app/oracle/product/ora92" then you should do:

```
$ cd /app/oracle/product
$ chown -R oracle:dba ora92
```

(You require root access or help of your System administrator for the same.)

The new owner should be also from the same group as like the current owner. You can verify it by using the “id” command.

```
$ id <user_name >
```

Consider your current user is "oracle92" and the group is "dba" then "id oracle92" will give output as uid=1003 (oracle92) gid=103(dba) where 1003, “oracle92” is the userid and the user name respectively, 103 and “dba” is the groupid and group name. The new owner “oracle” should be also from the same group “dba”

5) The recursive chown command will change the ownership of all the files in the Oracle Home. So as “root” user ,change the ownership of the files which are listed in the step 3 to the respective users using a chown.
For example:
After recursive chown

```
$ ls -l $ORACLE_HOME/bin/extjob
-rwsr-x--- 1 oracle dba 65319 Jan 8 19:00 ./bin/extjob
```

Change the ownership using chown.

```
$chown root:root $ORACLE_HOME/bin/extjob
-rwsr-x--- 1 root root 65319 Jan 8 19:00 ./bin/extjob
```

6) All the folders that belong to oracle installation are found in ORACLE_HOME except the Oracle Central Inventory.The location of the Central Inventory usually found from the oraInst.loc file, which contains the path as

```
"inventory_loc=< path >/oraInventory"
```

You have to change the ownership of the oraInventory also to the new owner.
(oraInst.loc file exists in /var/opt/oracle or /etc/oracle by default)

In 9i Oracle Universal Installer (OUI) is also found outside the ORACLE_HOME.The location of OUI is found in the oraInventory from a file called "oui.loc" which contails the path for

```
"InstLoc=< path >/oui".
```

You have to change the ownership of this also to the new owner. (In 10g OUI is found inside Oracle Home by default.)

OS Authentication

OS authentication allows Oracle to pass control of user authentication to the operating system. Non-priviliged OS authentication connections take the following form.

```
sqlplus /
sqlplus /@service
```

When a connection is attempted from the local database server, the OS username is passed to the Oracle server. If the username is recognized, the Oracle the connection is accepted, otherwise the connection is rejected. This article presents the configuration steps necessary to set up OS authentication on UNIX/Linux and Windows platforms.
First, create an OS user, in this case the user is called "diego". Once you created that, if you try to login as sqlplus "/ as sysdba" it will fail:
The connections failed because we have not told Oracle the users are OS authenticated. To do this, we must create an Oracle user, but first we must check the value of the Oracle `OS_AUTHENT_PREFIX` initialization parameter.

```
SQL> SHOW PARAMETER os_authent_prefix
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
os_authent_prefix                    string      ops$
SQL>
```

As you can see, the default value is "ops$". If this is not appropriate it can be changed using the `ALTER SYSTEM` command, but for now we will use this default value.
Now we know the OS authentication prefix, we can create a database user to allow an OS authenticated connection. To do this, we create an Oracle user in the normal way, but the username must be the prefix value concatenated to the OS username. So for the OS user "tim_hall", we would expect an Oracle username of "ops$tim_hall" on a UNIX or Linux platform.

```
-- UNIX
CREATE USER ops$tim_hall IDENTIFIED EXTERNALLY;
GRANT CONNECT TO ops$tim_hall;
```

The situation is complicated slightly on Windows platforms as the domain or machine name forms part of the username presented to Oracle. On Windows platforms you would expect an Oracle username of "OPS$DOMAIN\TIM_HALL" for the Windows user "tim_hall".

```
-- Windows
CREATE USER "OPS$ORACLE-BASE.COM\TIM_HALL" IDENTIFIED EXTERNALLY;
GRANT CONNECT TO "OPS$ORACLE-BASE.COM\TIM_HALL";
```

When using a Windows server, there is an additional consideration. The following option must be set in the "%ORACLE_HOME%\network\admin\sqlnet.ora" file.

```
SQLNET.AUTHENTICATION_SERVICES= (NTS)
```

With the configuration complete, now you can connect as that "diego" user.

Deleting Files from Windows

Delete files more than X days old in Windows "forfiles" is part of the resource kit, but ships with Windows 2003 Server and later.

forfiles -p d:\mssql_backup -s -m *.bak -d -2 -c "cmd /C del @Path"

1. -p: Start in the directory d:\mssql_backup
2. -s: process subdirectories
3. -m: match files using *.bak
4. -d: Find files more than 2 days old
5. -c: Execute the del command to delete the file. @Path has double-quotes around it already.

Improving SQL*Net Performance

Load Balancing tnsnames.ora

First setup multiple listeners on listener.ora. Example:

```
LSNR1666 =
(DESCRIPTION_LIST =
(DESCRIPTION =
 (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = jake)(PORT = 1666))
 )
)
)

LSNR2666 =
(DESCRIPTION_LIST =
(DESCRIPTION =
 (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = jake)(PORT = 2666))
 )
)
)

SID_LIST_LSNR1666 =
(SID_LIST =
(SID_DESC =
 (GLOBAL_DBNAME = lx10r2.us)
 (ORACLE_HOME = /u01/app/oracle/product/10.2.0.1)
 (SID_NAME = lx10r2)
)
)

SID_LIST_LSNR2666 =
(SID_LIST =
(SID_DESC =
 (GLOBAL_DBNAME = lx10r2.us)
 (ORACLE_HOME = /u01/app/oracle/product/10.2.0.1)
 (SID_NAME = lx10r2)
)
)
```

lsnrctl start lsnr1666

Next, we configure the client. The client naming is controlled by tnsnames.ora

```
LX10R2.US =
(DESCRIPTION =
  (ADDRESS_LIST=
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = jake)(PORT = 1666))
     (ADDRESS = (PROTOCOL = TCP)(HOST = jake)(PORT = 2666))
  )
  (CONNECT_DATA = (SID = lx10r2))
)
```

Here, the database alias is "lx10r2.us". Normally, when you specify multiple address lines for an alias, Oracle will attempt the first and if that fails will attempt the second. This might actually work for the poster, but would still pound port 1666 until it couldn't handle any connections and THEN try port 2666. The key to having a somewhat even distribution over the two listeners is by using the (LOAD_BALANCE=ON) parameter.

Connect to a system regardless of machine failure

You can place multiple address entries for a single connection alias in the TNSNAMES.ORA file. This means that you can connect to a database, even if some kind of physical failover occurred. Look at the following example:

```
10gexpress =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = Machine01)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = Machine02)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = XE)
    )
  )
```

Suppose Machine01 is down, then every new SQL*NET connection using service `10gexpress` will automatically login to Machine02. However, there is one restriction, the SID must be the same on both machines. This feature can provide guaranteed login for application servers and for the Oracle Parallel Server.

What can be done to increase SQL*Net performance?

While a SQL statement is running SQL*Net polls the client continuously to catch CONTROL-C situations. This results into a lot of *poll* and *fstat* system calls.
The following SQLNET.ORA parameter can be specified to reduce polling overhead on your system:

```
# Number of packets to skip between checking for breaks (default=4)
BREAK_POLL_SKIP=10000
```

Prespawned server sessions. You can tell the listener to start up a pool of idle server processes. When a connection request is made, it doesn't have to start a server process; it just hands one of the idle processes to the client (and then starts a new connection in its own time). This is configured in LISTENER.ORA, in the SID_LIST_LISTENER section, as follows:

```
SID_LIST_LISTENER =
   (SID_LIST =
     (SID_DESC =
       (SID_NAME = yourSID)
       (PRESPAWN_MAX = 50)
       (PRESPAWN_LIST =
         (PRESPAWN_DESC = (PROTOCOL = TCP) (POOL_SIZE = 5)
         (TIMEOUT = 2))))
)
```

PRESPAWN_MAX: if there are over 50 sessions connected to the database, the listener won't prespawn any more.

POOL_SIZE: the listener will maintain an idle pool of 5 server processes.

TIMEOUT: after a client disconnects, the listener will keep the freed-up server process around for two minutes, waiting for a new connection request, before killing that process.

Multiple listeners with load balancing. You can start multiplelisteners on a server, and reference all of the listeners in the TNSNAMES.ORA file. When a client makes a connection request, the SQL*Net client will randomly pick one of the listeners to contact.

In LISTENER.ORA, specify multiple listeners as in:

```
# Define listener A...
STARTUP_WAIT_TIME_LISTENER_A = 0
CONNECT_TIMEOUT_LISTENER_A = 10
LISTENER_A=
  (ADDRESS_LIST =
     (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = yourHost.domain)
        (PORT = 1521)))
SID_LIST_LISTENER_A =
   (SID_LIST =
      (SID_DESC =
         (SID_NAME = yourSID)
         (PRESPAWN_MAX = 50)))
# Define the second listener...
STARTUP_WAIT_TIME_LISTENER_B = 0
CONNECT_TIMEOUT_LISTENER_B = 10
LISTENER_B=
  (ADDRESS_LIST =
     (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = yourHost.domain)
        (PORT = 1522)))
SID_LIST_LISTENER_B =
   (SID_LIST =
      (SID_DESC =
         (SID_NAME = yourSID)
         (PRESPAWN_MAX = 50)))
```

The TNSNAMES.ORA service for this database would be something like:

```
oradb1.world =
  (description_list=
     (description=
        (address_list=
           (address=
              (protocol=tcp)
              (host=yourHost.domain)
              (port=1521)))
           (connect_data =
              (sid = yourSID)))
     (description =
        (address_list =
           (address=
                (protocol=tcp)
                (host=yourHost.domain)
                (port=1522)))
           (connect_data =
              (sid = yourSID))))
```

Changing Oracle UID and GID

In Unix, the file system only contains numeric UID and GID values, they only get converted to a name via lookup in /etc/passwd and /etc/group. (system calls getpwnam(), getpwuid(), getgrnam(), and getgrgid() perform this task). Oracle software does not know or care anything about the numeric UID/GID, only the names. So the change is pretty easy, just like changing the description for a unique ID in a lookup table in the database.
Here is a sample scenario. Assumes new UID and GID are not already in use, of course.

users:
oracle change 101 => 103
groups:
dba change 101 => 21

First, run pwck and grpck commands to clean up any problems with the respective files. Optional, but recommended (you'd be surprised what you might find).

```
# get "before" list of files to be changed for logging purposes
find / -user oracle -exec ls -ld {} \; > /tmp/ora_owned_files.lst
# find files which don't have DBA group, if any (shouldn't be any)
find / -user oracle \! -group dba -exec ls -ld {} \; >>
\ /tmp/ora_owned_files.lst
```

Shut down all oracle software (confirm with "ps -fu oracle" command).

```
# make the change
find / -user oracle -exec chown 103:21 {} \;

# make backups using RCS or your favorite method
cd /etc
ci -l passwd
ci -l group

# change lookups
vipw [...change oracle UID to 103, GID to 21]
vi /etc/group [change dba GID to 21]

# re-run listing to check for consistency
# check output to see what's changed...should be the same as "before"
# listing
find / -user oracle -exec ls -ld {} \; > ora_owned_files.lst.new
find / -user oracle \! -group dba -exec ls -ld {} \;
>> /tmp/ora_owned_files.lst.new &
```

It might be a little slow, you can experiment with the recursive option of chown instead of using find. Or, instead of -exec option of 'find', pipe output to xargs command. Just be sure you handle symbolic links correctly. (Your SA should understand all of this, in case you don't).

How to use rlwrap to get a command history in sql*plus

SQL*Plus does not have a command history function under Linux and Unix.
rlwrap is a readline wrapper for shell commands which uses input from the controlling terminal.
It adds a persistent input history for each command and supports user-defined completion.
You can download the sources for the rpm for rlwrap from HERE. The most recent version I could find is version 0.26.
There you also find a README and the manpage for rlwrap.

After downloading and unpacking the tar.gz
gunzip rlwrap-0.26.tar.gz
tar -xvf rlwrap-0.26.tar

I ran as root (#)
./configure
make
make check
make install

and that was it.

Now I could call sqlplus this way:
$ rlwrap sqlplus user/password@sid.

Finally I created an alias
alias q="rlwrap sqlplus"
or
alias sqlplus="rlwrap sqlplus"

I would avoid the last one. rlwrap does not support non-interactive mode (echo xxx | sqlplus), is not an oracle support tool, and crashes occasionnaly.
So that I can always use sqlplus to run important script, and keep rlwrap for not-productive stuff.

Now I can simply call sqlplus as I always have done and have a commend history with the keys on my keyboard.

Database Tips

Writing to the alert log
If you want to write a message to the alert log, you can use the undocumented KSDWRT procedure of the DBMS_SYSTEM package.
This procedure has two parameters, the first one must be "2" to write to the alert file, the second one is the message you want to write.
Here is an example:

```
    execute sys.dbms_system.ksdwrt(2,to_char(sysdate)|| ' -- ');
```

Use 1 instead of 2 to write to the trace file
Use 3 to write to both.

Example:

```
exec sys.dbms_system.ksdwrt(2, 'CUS-00001: message from me'||chr(10)||'On another line');
```

--> In the alert.log file:

```
Wed Jun 25 19:49:30 2003
CUS-00001: message from me
On another line
```

Connect as SYSDBA
For most scripts it should be sufficient to replace:

```
 CONNECT INTERNAL
  with
 CONNECT / AS SYSDBA
```

but it would be sensible to review the alternatives before making such a change. Additionally , scripts which call 'svrmgrl' should be ammended thus:

Old form of command	New form of command
svrmgrl	sqlplus /nolog connect / as sysdba

To connect from SQL*Plus:
FROM Solaris to Solaris DB
```
sqlplus "/ as sysdba"
```

From my PC to Filerver DB
```
sqlplus /nolog
conn sys/passwd@fileserver as sysdba
```
or
```
sqlplus "sys@fileserver as sysdba"
```
or
```
sqlplus "sys/passwd@fileserver as sysdba"
```

Administrative Users
There are two main administrative privileges: SYSOPER and SYSDBA
SYSOPER privilege allows operations such as:
Instance startup, mount & database open ;
Instance shutdown, dismount & database close ;
Alter database BACKUP, ARCHIVE LOG, and RECOVER.
This privilege allows the user to perform basic operational tasks without the ability to look at user data.

SYSDBA privilege includes all SYSOPER privileges plus full system privileges (with the ADMIN option), plus 'CREATE DATABASE' etc..
This is effectively the same set of privileges available when previously connected INTERNAL.

Password Authentication
Unless a connection to the instance is considered 'secure' then you must use a password to connect with SYSDBA or SYSOPER privilege. Users can be added to a special 'password' file using either the 'ORAPWD' utility, or 'GRANT SYSDBA to USER' command. Such a user can then connect to the instance for administrative purposes using the syntax:
```
CONNECT username/password AS SYSDBA
```
or
```
CONNECT username/password AS SYSOPER
```

Users with SYSDBA or SYSOPER privilege can be seen in the view V$PWFILE_USERS.

Operating System Authentication
If the connection to the instance is local or 'secure' then it is possible to use the operating system to determine if a user is allowed SYSDBA or SYSOPER access. In this case no password is required. The syntax to connect using operating system authentication is:
```
CONNECT / AS SYSDBA
```
or
```
CONNECT / AS SYSOPER
```

Outer joins - Usage and efficiency

Outer joins enable rows to be returned from a join where one of the tables does not contain matching rows for the other table.

eg. Suppose we have two tables:

```
     Person
     ------
         Person_id   Name                  Address_id
         ---------   ----------------      ----------
         00001       Fred Bloggs           00057
         00002       Joe Smith             00092
         00003       Jane Doe
         00004       Sue Jones             00111
     Address
     -------
         Address_id   Address_Desc
         ----------   ------------------------
         00057        1, Acacia Avenue, Anytown
         00092        13, High Street, Anywhere
         00113        52, Main Road, Sometown
```

Then the simple join:

```
         SELECT PERSON.NAME, ADDRESS.ADDRESS_DESC
           FROM PERSON, ADDRESS
          WHERE PERSON.ADDRESS_ID = ADDRESS.ADDRESS_ID
returns:
    NAME          ADDRESS_DESC
    ----------    ------------
    Fred Bloggs   1, Acacia Avenue, Anytown
    Joe Smith     13, High Street, Anywhere
```

But the outer join:

```
         SELECT PERSON.NAME, ADDRESS.ADDRESS_DESC
           FROM PERSON, ADDRESS
          WHERE PERSON.ADDRESS_ID = ADDRESS.ADDRESS_ID(+)
returns:
    NAME          ADDRESS_DESC
    ----------    ------------
    Fred Bloggs   1, Acacia Avenue, Anytown
    Joe Smith     13, High Street, Anywhere
    Jane Doe
    Sue Jones
```

Note the two new rows for Jane Doe and Sue Jones. These are the people who do not have matching records on the ADDRESS table. Sue Jones had an address_id on her PERSON record, but this didn't match an address_id on the ADDRESS table. (Probably a data inconsistency). Jane Doe had NULL in her PERSON.ADDRESS_ID field, which obviously doesn't match any address_id on the ADDRESS table.
Note that the outer join is created by including (+) on the WHERE clause which joins the two tables. The (+) is put against the column-name on the *deficient* table, ie. the one with the missing rows. It is very important to put the (+) on the correct table: putting it on the other table will give different results. eg. the query:

```
         SELECT PERSON.NAME, ADDRESS.ADDRESS_DESC
           FROM PERSON, ADDRESS
          WHERE PERSON.ADDRESS_ID(+) = ADDRESS.ADDRESS_ID
returns:
    NAME          ADDRESS_DESC
    ----------    ------------
    Fred Bloggs   1, Acacia Avenue, Anytown
    Joe Smith     13, High Street, Anywhere
                  52, Main Road, Someplace
```

Managing Dates

You can use fractions with sysdate (or any date column) to add hours, minutes and/or seconds. For hours, use a

denominator of 24; for minutes use 1440; for seconds: 86400.
For example "sysdate + 3/24" will add 3 hours, "sysdate + 5/1440" will add 5 minutes; "sysdate + 45/86400" will add 45 seconds. You can use values like "sysdate + 30/24" to add one day +6 hours if you need to. Example:

```
SELECT to_char(sysdate,'dd-mon-yyyy hh:mi:ss'),
        TO_CHAR(SYSDATE + 10/1440,'dd-mon-yyyy hh:mi:ss') FROM DUAL;
```

Using Oracle functions for dates
An alternative to this method is to use the **numtodsinterval** function. Example to add 2 hours:

```
SELECT to_char(sysdate, 'DD/MON/YY HH24:MI:SS'), to_char(sysdate +
numtodsinterval(2, 'HOUR'), 'DD/MON/YY HH24:MI:SS')
FROM dual;

TO_CHAR(SYSDATE,'D TO_CHAR(SYSDATE+NU
----------------- -----------------
17/JAN/06 12:50:53 17/JAN/06 14:50:53
```

Here the numtodsinterval function is doing the work of dividing 2/24 for hours.
Valid options for the numtodsinterval are 'DAY', 'HOUR', 'MINUTE', or 'SECOND'. Here is an example using 'MINUTE'.
When working with minutes the numtodsinterval function is much more readable.

```
SELECT to_char(sysdate, 'DD/MON/YY HH24:MI:SS'), to_char(sysdate +
numtodsinterval(45, 'MINUTE'), 'DD/MON/YY HH24:MI:SS')
    FROM dual;

TO_CHAR(SYSDATE,'D TO_CHAR(SYSDATE+NU
----------------- -----------------
17/JAN/06 12:50:09 17/JAN/06 13:35:09
```

Adjusting Months and Years
To work with months and years (either of which may have a varying number of days) Oracle has provided the function **numtoyminterval**.
This works much like the numtodsinterval function mentioned above by taking a number and a string. Valid options for the string are 'YEAR' or 'MONTH'.

```
SELECT sysdate, sysdate + numtoyminterval(5, 'MONTH')
FROM dual;
SYSDATE             SYSDATE+NUMTOYMINT
------------------ ------------------
17/JAN/06 12:53:49 17/JUN/06 12:53:49

SELECT sysdate, sysdate + numtoyminterval(2, 'YEAR')
FROM dual;
SYSDATE             SYSDATE+NUMTOYMINT
------------------ ------------------
17/JAN/06 12:54:08 17/JAN/08 12:54:08
```

Truncate minutes and seconds

```
select to_char(TRUNC(sysdate,'HH24'),'dd-mon-yyyy hh:mi:ss') from dual;
select to_char(TRUNC(sysdate,'MI'),'dd-mon-yyyy hh:mi:ss') from dual;
```

How can I get the time difference between two date columns
Look at this example query:

```
define date1 = sysdate+3;
```

```
define date2 = sysdate ;
select floor(((&date1-&date2)*24*60*60)/3600) || ' HOURS ' ||
floor((((&date1-&date2)*24*60*60) - floor(((&date1-&date2)*24*60*60)/3600)
*3600)/60) || ' MINUTES ' ||
round((((&date1-&date2)*24*60*60) - floor(((&date1-&date2)*24*60*60)/3600)*3600
-(floor((((&date1-&date2)*24*60*60) -
floor(((&date1-&date2)*24*60*60)/3600)*3600)/60)*60))) || ' SECS '
time_difference
from DUAL;
undef date1;
undef date2;
```

Extract Date Information

```
select sysdate from dual;
SYSDATE
------------------
06/DEC/06 11:06:12

select extract(day from sysdate) from dual;
EXTRACT(DAYFROMSYSDATE)
-----------------------
                      6

select extract(month from sysdate) from dual;
EXTRACT(MONTHFROMSYSDATE)
-------------------------
                       12

select extract(year from sysdate) from dual;
EXTRACT(YEARFROMSYSDATE)
------------------------
                    2006
```

Adding the Oracle SID to the SQL Prompt

Just add the following statements to the GLOGIN.SQL to populate the SQL command prompt with the Oracle SID:

```
--  Add any sqlplus commands here that are to be executed when a user
--  starts SQL*Plus on your system

-- Used by Trusted Oracle
column ROWLABEL format A15

-- OLD Added by me to show the SID
-- column sid new_value osid noprint;
-- select upper(substr(global_name,1,(instr(global_name,'.')-1))) sid
-- from global_name;
-- set sqlprompt '&osid SQL> '

set serveroutput on size 1000000
alter session set nls_date_format = 'DD/MON/YY hh24:mi:ss';

-- Added by me to show the SID
set head off
```

```
set feed off
set term off
column C01 format A30 new_value DIA
column C02 format A8 new_value NOMBASE
column C03 format A8 new_value NOMUSER
column C04 format A10 new_value TERMINAL
column C05 format A5 new_value SESSION

select decode(upper(substr(global_name,1,(instr(global_name,'.')-1))),
null,global_name, upper(substr(global_name,1,(instr(global_name,'.')-1)))) C02,

       USERENV('TERMINAL') C04, USERENV('SESSIONID') C05
from global_name;

select USER C03, to_char(sysdate,'DD Month YYYY - HH24:MI') C01 from dual;

set term on
set verify off
select '****************************************'||CHR(10)||
       ' &DIA                                  '||CHR(10)||CHR(10)||
       ' Connected to  : &NOMBASE'              ||CHR(10)||
       ' As User       : &NOMUSER.'             ||CHR(10)||
       ' Terminal      : &TERMINAL '            ||CHR(10)||
       ' Session_ID    : &SESSION'              ||CHR(10)||
       '****************************************' from dual
/

undef DIA
set verify on
set feed on
set head on
set sqlprompt "&NOMBASE/&NOMUSER> "
undef NOMUSER
undef NOMBASE
clear buffer
clear COLUMNS

-- Used for the SHOW ERRORS command
column LINE/COL format A8
column ERROR    format A65  WORD_WRAPPED

-- Used for the SHOW SGA command
column name_col_plus_show_sga format a24

-- Defaults for SHOW PARAMETERS
column name_col_plus_show_param format a36 heading NAME
column value_col_plus_show_param format a30 heading VALUE

-- For backward compatibility
set pagesize 14

-- Defaults for SET AUTOTRACE EXPLAIN report
column id_plus_exp format 990 heading i
column parent_id_plus_exp format 990 heading p
```

```
column plan_plus_exp format a60
column object_node_plus_exp format a8
column other_tag_plus_exp format a29
column other_plus_exp format a44
```

Move Tables to another Tablespace
ALTER TABLE <table> MOVE TABLESPACE <new_tablespace> ;

Rebuilding Indexes
You can use the ALTER INDEX REBUILD command to change the storage and tablespace parameters for an existing index without having to drop it.
The following is an example of an index rebuild via this command. It's storage parameters are changed to use an initial extent size of 3MB and a next extent size of 500K. It is also being moved from the USR7 tablespace to the IDX7 tablespace.

ALTER INDEX fuzzy_pink_slippers REBUILD
(STORAGE(INITIAL 3M NEXT 500K
PCTINCREASE 0)
TABLESPACE IDX7;)

alter index <index_name> REBUILD TABLESPACE <new_tablespace>;

Rebuild? There is an alternative
From 8i onwards, there is another option you may wish to consider.
ALTER INDEX vmoore COALESCE;

Coalescing an index, de-fragments the leaf blocks for an index as opposed to a full rebuild cycle. In many instances, this may be sufficient remedial action.
Positives
- Less resource used - in fact no additional space is required
- Faster (typically)
- Good for databases with large block sizes (see the negative point below on index height)

Negatives
- You aren't recreating anything, so obviously you cannot move the index or adjust its storage parameters during a coalesce
- Since only the leaf blocks are coalesced, the height of the index tree will not be changed. For databases with larger block sizes, this should be less significant since the indexes will be "flatter" anyway.

Archiving ON
Enabling Automatic Archiving before Instance Startup
Shutdown the instance and change the folowing parameters:
Example:
Log_archive_start = True
Log_archive_format = "LOG_%t%s.ARC"
log_archive_dest_1 = "location=/u01/oradata/TICPBT09/ARCH MANDATORY"
log_archive_dest_state_1 = enable
log_archive_dest_2 = "service=TICPBP02 OPTIONAL reopen=60"

log_archive_dest_state_2 = enable
log_archive_min_succeed_dest = 1

For 10g
The LOG_ARCHIVE_START init.ora parameter has been rendered obsolete. In Oracle 10g, if the LOG_ARCHIVE_DEST is not set, archiving will be directed to the flashback recovery area automatically when the database is switch to ARCHIVELOG mode.
**.log_archive_format='CCOM%t_%s_%r.dbf'*
**.log_archive_dest_1='LOCATION=/u02/oradata/ARCH'*
log_archive_dest_state_1 = enable
log_archive_dest_2 = "service=TICPBP02 OPTIONAL reopen=60"
log_archive_dest_state_2 = enable
log_archive_min_succeed_dest = 1

- Start up a new instance and mount, but do not open the database.
  startup mount

- Switch the database's archiving mode and open it
  alter database archivelog;
  alter database open;

- Verify your database is now in archivelog mode.
  archive log list;
- Archive all your redo logs at this point.
  archive log all;
  or
  alter system switch logfile;
  or
  alter system archive log current;

- Ensure these newly created Archive log files are added to the backup process

Enabling Automatic Archiving After Instance Startup
This step is NOT NEEDED IN 10G, BY DEFAULT IS AUTOMATIC!!!
To enable automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG START parameter;
The following statement enables archiving:
SVRMGRL> ALTER SYSTEM ARCHIVE LOG START;
Anyway, remeber to modify the init.ora parameters.

Archiving OFF
Disabling Automatic Archiving at Instance Startup (best method)
Shutdown the instance and change the "LOG_ARCHIVE_START" parameter
Log_archive_start = False
Then:
startup mount
alter database noarchivelog;
alter database open;

Disabling Automatic Archiving After Instance Startup
To disable the automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG STOP parameter.

The following statement stops archiving:

SVRMGRL> ALTER SYSTEM ARCHIVE LOG STOP;

If ARCH is archiving a redo log group when you attempt to disable automatic archiving, ARCH finishes archiving the current group, but does not begin archiving the next filled online redo log group.
The instance does not have to be shut down to disable automatic archiving. However, if an instance is shut down and restarted after automatic archiving is disabled, the instance is reinitialized using the settings of the parameter file ("LOG_ARCHIVE_START"), which may or may not enable automatic archiving.

NOTE: If you choose to disable automatic archiving and have not disabled archiving altogether, you are responsible to archive
all filled redo log groups or else database operation is temporarily suspended (you will experience a database hang)
until the necessary archiving is performed.

Performing Manual Archiving
If a database is operating in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. You can manually archive groups of the online redo log whether or not automatic archiving is enabled. If automatic archiving is not enabled, you must manually archive groups of filled online redo log files in a timely fashion. If all online redo log groups are filled but not archived, LGWR cannot reuse any inactive groups of online redo log members to continue writing redo log entries. Therefore, database operation is temporarily suspended until the necessary archiving is performed. You can exercise this scenario by executing
alter system switch logfile;
command when automatic archival is disabled.
Attempting to repeat that command with a last redo log group will show hang, and it won't be completed with "statement processed" message until archiving is done.
If automatic archiving is enabled, but you want to rearchive an inactive group of filled online redo log members to another location, you can use manual archiving. (However, the instance can decide to reuse the redo log group before you have finished manually archiving, thereby overwriting the files. If this happens, Oracle will display an error message in the ALERT file.)
To manually archive a filled online redo log group, you must be connected with administrator privileges.
Manually archive inactive groups of filled online redo log members using the SQL command:
SVRMGRL> ALTER SYSTEM ARCHIVE LOG ALL;.

Recompiling Invalid Schema Objects

The DBA_OBJECTS view can be used to identify invalid objects using the following query:

```
COLUMN object_name FORMAT A30
SELECT substr(owner,1,18) owner, object_type, substr(object_name,1,30)
object_name, status
FROM   dba_objects
WHERE  status = 'INVALID'
ORDER BY owner, object_type, object_name;
```

DBMS_UTILITY.COMPILE_SCHEMA
The COMPILE_SCHEMA procedure in the DBMS_UTILITY package compiles all procedures, functions, packages, and triggers in the specified schema. The example below shows how it is called from SQL*Plus:

```
EXEC DBMS_UTILITY.compile_schema(schema => 'SCOTT');
```

If you are using 10g, you can use the following to recompile ONLY the invalid ones:

```
EXEC DBMS_UTILITY.compile_schema(schema => 'SCOTT', compile_all => 'FALSE');
```

UTL_RECOMP
The UTL_RECOMP package contains two procedures used to recompile invalid objects. As the names suggest, the

RECOMP_SERIAL procedure recompiles all the invalid objects one at a time, while the RECOMP_PARALLEL procedure performs the same task in parallel using the specified number of threads. The following examples show how these procedures are used:

```
-- Schema level.
EXEC UTL_RECOMP.recomp_serial('SCOTT');
EXEC UTL_RECOMP.recomp_parallel(4, 'SCOTT');

-- Database level.
EXEC UTL_RECOMP.recomp_serial();
EXEC UTL_RECOMP.recomp_parallel(4);
```

There are a number of restrictions associated with the use of this package including:

- Parallel execution is perfomed using the job queue. All existing jobs are marked as disabled until the operation is complete.
- The package must be run from SQL*Plus as the SYS user, or another user with SYSDBA.
- The package expects the STANDARD, DBMS_STANDARD, DBMS_JOB and DBMS_RANDOM to be present and valid.
- Runnig DDL operations at the same time as this package may result in deadlocks.

utlrp.sql and utlprp.sql

The utlrp.sql and utlprp.sql scripts are provided by Oracle to recompile all invalid objects in the database. They are typically run after major database changes such as upgrades or patches. They are located in the $ORACLE_HOME/rdbms/admin directory and provide a wrapper on the UTL_RECOMP package. The utlrp.sql script simply calls the utlprp.sql script with a command line parameter of "0". The utlprp.sql accepts a single integer parameter that indicates the level of parallelism as follows:

- 0 - The level of parallelism is derived based on the CPU_COUNT parameter.
- 1 - The recompilation is run serially, one object at a time.
- N - The recompilation is run in parallel with "N" number of threads.

Both scripts must be run as the SYS user, or another user with SYSDBA, to work correctly.

DBMS_XPLAN

In version 9, Oracle finally provides a utility that formats the contents of the plan table. The plan table is one that is used to hold the results of an "Explain Plan" for a particular SQL statement. The output from the explain plan shows the anticipated optimizer execution path, along with the estimated cost of the statement without actually executing the statement against the database. The DBA or developer first needs to create the plan table. The DDL for this table is in the $ORACLE_HOME/rdbms/admin/utllxplan.sql file. The next step in using dbms_xplan is running Explain Plan for a statement.

*explain plan for select * from flowdocument where amount > 100000;*

The command above will populate the plan table with the data returned from the optimizer. Next, the dbms_xplan utility can be used to view the output

*select * from table(dbms_xplan.display);*

```
-------------------------------------------------------------------------
| Id  | Operation            |  Name         | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |               |     1 |   168 |     3 |
|   1 |  TABLE ACCESS FULL   | FLOWDOCUMENT  |     1 |   168 |     3 |
-------------------------------------------------------------------------
```

Delete Duplicate Records

--Very Efficient, deleted 685 rows over a 5443932 rows table in 3 Min

```
DELETE FROM &&table_name
   WHERE rowid NOT IN (SELECT max(rowid)
                       FROM &&table_name
                       GROUP BY &colums_with_duplicates);
```

Another way

```
SELECT a.rowid
   FROM &&table_name a
   WHERE a.rowid > (SELECT min(b.rowid)
                    FROM &&table_name b
                    WHERE a.&&column_name = b.&&column_name);
```

Then:

```
DELETE from &&table_name
    WHERE a.rowid > (SELECT min(b.rowid)
                    FROM &&table_name b
                    WHERE a.&&column_name = b.&&column_name);
```

**** Most efficient way to remove duplicate rows
This script uses a hash join -- the most efficient way of joining huge tables -- to find duplicate rows.

```
-- Set hash join enabled
DELETE FROM <table>
  WHERE rowid IN (SELECT t1.rowid
                  FROM <table> t1, <same-table> t2
                  -- primary key is (a1, a2)
                  WHERE t1.a1 = t2.a1
                    AND t1.a2 = t2.a2
                    AND t1.rowid < t2.rowid);
```

****Another METHOD (by METALINK) to find duplicates for one field****
To find duplicate keys from a table tx:

```
select key, count(key) no_of_duplicates
   from tx
   group by key
   having count(key) > 1;
```

*** This script will remove duplicate rows. Suppose a table contains 3 columns. To remove the duplicate rows write the following command, where a and b are aliases of the same table.

```
Delete from table_A a
   where a.rowid > any (select rowid
                        from Table_B b
                        where a. = b.
                          and a. = b.
                          and a. = b.);
```

Shrinking Datafiles

For example, if a datafile is 100Meg in size, and 70Meg of the datafile is currently in use. Then atleast 70Meg must be

left in the datafile. The RESIZE parameter of the ALTER DATABASE command is used to reclaim the space.
ALTER DATABASE datafile '/db01/oracle/fix/data03.ora' resize 80M;

Reduce UNDO Tablespace

The Undo tablespace file can grow quickly if you load data in the database. To reduce the size of the file, you have to recreate the default undo tablespace because the command

```
alter database datafile 'XXX' resize XM;
```

will not work.

Create a second undo tablespace

```
CREATE UNDO TABLESPACE UNDOTBS2 DATAFILE '/chemin/undotbs2.dbf' SIZE 50M
AUTOEXTEND OFF ;
```

Set the second undo tablespace as default undo

```
ALTER SYSTEM SET undo_tablespace = UNDOTBS2 ;
```

Drop the first undo tablespace

```
DROP TABLESPACE undotbs1 INCLUDING CONTENTS AND DATAFILES ;
```

Create the first undo tablespace and limit the size to 1Go

```
CREATE UNDO TABLESPACE UNDOTBS1 DATAFILE '/chemin/undotbs2.dbf' SIZE 500M
AUTOEXTEND ON NEXT 5 M MAXSIZE 1000M ;
```

Set the first undo tablespace as default undo

```
ALTER SYSTEM SET undo_tablespace = UNDOTBS1 ;
```

Drop the second undo tablespace

```
DROP TABLESPACE undotbs2 INCLUDING CONTENTS AND DATAFILES ;
```

Reduce TEMP Tablespace

In many database configurations, the DBA will choose to allow their temporary tablespace (actually the tempfile(s) for the temporary tablespace) to autoextend. A runaway query or sort can easily chew up valuable space on the disk as the tempfiles(s) extends to accommodate the request for space. The obvious action would
be to resize the tempfiles using the following statement:

```
SQL> alter database tempfile '/u02/oradata/TESTDB/temp01.dbf' resize 250M;
alter database tempfile '/u02/oradata/TESTDB/temp01.dbf' resize 250M
*
ERROR at line 1:
ORA-03297: file contains used data beyond requested RESIZE value
```

The procedures above document how to drop a temporary tablespace that is *not* the default temporary tablespace for the database. You will know fairly quickly if the tablespace is a default temporary tablespace when you are greeted with the following exception:

```
SQL> DROP TABLESPACE temp;
drop tablespace temp
*
ERROR at line 1:
ORA-12906: cannot drop default temporary tablespace
```

In cases where the temporary tablespace you want to resize (using the drop/recreate method) is the default temporary tablespace for the database, you have several more steps to perform, all documented below. The first step you need to perform is create another temporary tablespace (lets call it TEMP2). The next step would be to remove the temporary

tablespace you want to resize from being the default temporary tablespace (in our example, this will be a tablespace named TEMP) by making TEMP2 the default. Drop / recreate the TEMP tablespace to the size you want. Finally, make the newly created TEMP tablespace your default temporary tablespace for the database and drop the TEMP2 tablespace. A full example session is provided below:

```
SQL> CREATE TEMPORARY TABLESPACE temp2
    TEMPFILE '/u02/oradata/TESTDB/temp2_01.dbf' SIZE 5M REUSE
    AUTOEXTEND ON NEXT 1M MAXSIZE unlimited
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp2;
SQL> DROP TABLESPACE temp INCLUDING CONTENTS AND DATAFILES;
SQL> CREATE TEMPORARY TABLESPACE temp
  TEMPFILE '/u02/oradata/TESTDB/temp01.dbf' SIZE 500M REUSE
  AUTOEXTEND ON NEXT 100M MAXSIZE unlimited
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
SQL> DROP TABLESPACE temp2 INCLUDING CONTENTS AND DATAFILES;
```

Transportable Tablespaces

1- Make the tablespace Read-Only = alter tablespace xxxx read only;
2- Export it connecting as sys as sysdba = exp file=tt.dmp log=tt.log tablespaces=xxxx transportable_tablespaces=y
3- Copy the .dmp file and the data_files to the destination
4- Put the tablespace back in write mode = alter tablespace xxxx read write;
5- In the destination offline and drop the tablespace if exists
6- Import = imp file=tt.dmp log=tt.log tablespaces=test transportable_tablespace=y datafiles=(....., ...)

Orakill Utility

The *orakill* utility is provided with Oracle databases on Windows platforms. The executable (*orakill.exe*) is available to DBA's to kill Oracle sessions directly from the DOS command line without requiring any connection to the database.
In the Unix world, a DBA can kill a shadow process by issuing the kill -9 commands from the Unix prompt. Unix is able to provide this capability given that the Unix operating system is based on processes that fork other processes. All processes can be listed by using the ps Unix command. The Oracle background processes will be listed separately from all of the Oracle sessions since they have their own process.
Unlike the Unix operating system, Windows systems are thread-based. The background processes and sessions are all contained within the ORACLE.EXE executable and are not listed in the "Processes" tab of Windows Task Manager.
The *orakill* utility serves the same purpose as *kill -9* in Unix. The command requires the instance and the SPID of the thread to kill. The following query will return the SPID for each user connected to the database:

```
select a.username, a.osuser, b.spid
  from v$session a, v$process b
  where a.paddr = b.addr
    and a.username is not null;

USERNAME                       OSUSER                   SPID
------------------------------ ------------------------ -----
SCOTT                          Scott                    3116
AMOORE                         Alex                     4760
DMOORE                         Dave                     768
```

Given the SPID for each user listed above, the session for any user can be killed with the orakill command.
C:\oracle9i\bin>orakill ORCL92 4760

Why does Oracle provide a utility to kill sessions from the DOS prompt when a DBA could kill a user session from within Oracle? The following command will also kill the user session:
alter system kill session(sid, serial#);

The sid and serial# used in the command above can be obtained from the v$session view. There are a few reasons a DBA could use orakill instead of the alter system kill session command.
1. The alter system statement will not clear the locks if any exist. Instead, the session will remain connected until it times out, then the session is killed and the locks are released. The orakill command will kill the thread and the locks instantly.
2. A DBA may not be able to gain access to a SQL prompt due to a runaway query consuming all system resources. In this case, the session can be killed without ever logging in to the database.
The *orakill* utility should be used as a last resort only. If the session cannot be killed more gracefully (via *alter system kill session*), or the instance is inaccessible via SQL, then *orakill* can be used to terminate the offending session. Background processes should not be terminated, only user sessions. Killing a background process can cause serious Oracle errors and can bring the database down.

The PL/SQL WRAP Utility
The WRAP encrypts PL/SQL code displaying it in hexadecimal format. Use the following command:

wrap iname=script.sql

The output file will be called script.plb. To rename the output file, use the oname option of Wrap (i.e., oname=output.sql).

COPY Command
It allows data to be copied between databases (or within the same database) via SQL*PLUS. The greatest ability of this command is to COMMIT alfter each array of data. Though the copy of an extremely large table can tax the rollback segments, it is possible to break the transaction into smaller entries. The syntax for this command is:

COPY FROM
remote username/remote password@connect string
TO
username/password@connect string
{APPEND, CREATE, INSERT, REPLACE]
table name using subquery

To set the transaction entry size, use the SQL*PLUS SET command to set a value for the ARRAYSIZE parameter. This determines the number of records that will be retrieved in each "batch". The COPY COMMIT parameter tells SQL*PLUS how many batches should be committed at one time. In the following example, the data is committed after every 1,000 records. This reduces the transaction's rollback segment entry size.
SET COPYCOMMIT 1;
SET ARRAYSIZE 1000
COPY FROM batman/androbin @t:gotham:city CREATE batmobile USING SELECT * FROM bat_mobile;
NOTE: The feedback from this command is not acqurate. After the final commit is complete, the database reports the number of records that were committed in the last batch. It does not report the total number of records committed.

Authid Current_User vs Authid Definer
A stored procedure is either run with the rights of the caller (*authid current_user*) or with the rights of the procedure's owner (*authid definer*).
This authid clause immediatly follows the create procedure or create function statement.
It can be ommited, in which case the default authid definer is taken. Example:

```
create procedure pu
 authid definer
```

```
as
  v_d t.d%type;
begin
  select d into v_d from u;
  dbms_output.put_line(v_d);
end;
/

or

CREATE OR REPLACE
Package fg_mc_web_insert_update AUTHID CURRENT_USER
  IS
.....
```

- How to find what is locking a table.

1. Query from DBA_OBJECTS to find the object_name of the table getting locked.
2. Query from V$LOCK where id1 = 'table_name', get sid.
3. Query from v$PROCESS where pid = sid. THis has info on what is locking the table.

Select Randomly from Table

Since Oracle 8i, there is a new feature that allows you to randomly "sample" from a table. This feature has many great uses.
The syntax is as follows:

```
SELECT COUNT(*) * 100
FROM EMP SAMPLE (1);
```

This will randomly sample 1% of the rows, multiple the count of them x 100 to get a rough estimate of the amount of rows in the table.
You can also randomly sample by blocks for better performance but possibly less random:

```
SELECT *
FROM EMP SAMPLE BLOCK (1);
```

Again, this samples roughly 1% of the table by blocks, which may not be 1% of the actual rows. But this will cause fewer blocks to be visited and decrease the elapsed time, but if the data is grouped in the table, it may not be very random.
This tool can be used to get a rough idea of the data in the table, or give good estimates when using group functions. For example, a great use of this would be on a 40 million row table:

```
SELECT AVG(number_of children) * 20
FROM dependants sample (5);
```

This will give you an average of the number of dependants by only sampling 5% of the table by only visiting 2 million rows and not 40 million.

Disk Space Needed for the Archive Files

The output of the above script tells you how many log switches are occurring on your system on a daily basis. The

actual disk space that is required to serve the archiving is given as well in MB. All you need to determine if the amount of log switches are stable or difference a lot from day to day. First obtain the size in Mb of your online redo log files, you can execute the following query:

SELECT distinct(to_char((bytes*0.000001),'9990.999')) size_mb FROM v$log;

Example output: SIZE_MB

1.024

Then run:

```
column ord     noprint
column date_   heading 'Date'         format A15
column no      heading '#Arch files'  format 9999999
column no_size heading 'Size Mb'      format 9999999
compute avg of no      on report
compute avg of no_size on report
break on report

select MAX(first_time) ord, to_char(first_time,'DD-MON-YYYY') date_,
      count(recid) no, count(recid) * &logfile_size no_size
from v$log_history
group by to_char(first_time,'DD-MON-YYYY')
order by ord
/
clear breaks
clear computes
clear columns
```

Unlocking Users

alter user &username account unlock;

DBMS_Utility.get_time

The get_time function in the dbms_utility package is intended to be used to get the difference in time between two calls to it. Its advantage is that the time is in hundredths of seconds granularity instead of seconds.

```
DECLARE
   v_time     number;
BEGIN
   v_time := dbms_utility.get_time;
   --your code
  dbms_output.put_line('Time Used: ' || (dbms_utility.get_time - v_time) / 100
|| ' secs');
END;
```

Using orapwd to Connect Remotely as SYSDBA

The Oracle *orapwd* utility assists the DBA with granting SYSDBA and SYSOPER privileges to other users. By

default, the user SYS is the only user that has these privileges. Creating a password file via orapwd enables remote users to connect with administrative privileges through SQL*Net. The SYSOPER privilege allows instance startup, shutdown, mount, and dismount. It allows the DBA to perform general database maintenance without viewing user data. The SYSDBA privilege is the same as connect internal was in prior versions. It provides the ability to do everything, unrestricted. If orapwd has not yet been executed, attempting to grant SYSDBA or SYSOPER privileges will result in the following error:
SQL> grant sysdba to scott;
ORA-01994: GRANT failed: cannot add users to public password file

The following steps can be performed to grant other users these privileges:
1. Create the password file. This is done by executing the following command:
$ orapwd file=filename password=password entries=max_users
The filename is the name of the file that will hold the password information. The file location will default to the current directory unless the full path is specified. The contents are encrypted and are unreadable. The password required is the one for the SYS user of the database. The max_usersis the number of database users that can be granted SYSDBA or SYSOPER. This parameter should be set to a higher value than the number of anticipated users to prevent having to delete and recreate the password file.

2. Edit the init.ora parameter remote_login_passwordfile. This parameter must be set to either SHARED or EXCLUSIVE.When set to SHARED, the password file can be used by multiple databases, yet only the SYS user is recognized. When set to EXCLUSIVE, the file can be used by only one database, yet multiple users can exist in the file. The parameter setting can be confirmed by:
SQL> show parameter password

```
NAME                           TYPE        VALUE
------------------------------ ----------- ----------
remote_login_passwordfile      string      EXCLUSIVE
```

3. Grant SYSDBA or SYSOPER to users. When SYSDBA or SYSOPER privileges are granted to a user, that user's name and privilege information are added to the password file.
SQL> grant sysdba to scott;
Grant succeeded.

4. Confirm that the user is listed in the password file.
*SQL> select * from v$pwfile_users;*

```
USERNAME                       SYSDBA SYSOPER
------------------------------ ------ -------
SYS                            TRUE   TRUE
SCOTT                          TRUE   FALSE
```

Now the user SCOTT can connect as SYSDBA. Administrative users can be connected and authenticated to a local or remote database by using the SQL*Plus connect command. They must connect using their username and password, and with the AS SYSDBA or AS SYSOPER clause:
SQL> connect scott/tiger as sysdba;

The DBA utilizes the orapwd utility to grant SYSDBA and SYSOPER privileges to other database users. The SYS password should never be shared and should be highly classified.

Change DB Name and File paths

A common way to rename all the database and all data files is to rebuild the control file. This can be very labor intensive, however, and possibly dangerous if you have a large number of data files. You could make a mistake and lose a data file or two in the process because the database does not check the validity of the data files when you do a control file rebuild. Here is a low-stress, easy solution:

Step 1

Run the scripts below (change directories for your server) with the target database open. Spool out a file for each of your available mount points.
Yes, we will be trying to move the data files more then once, however, Oracle checks whether the data file exists before accepting the command and returns errors if it does not exist. This uses the Oracle Server to verify the validity of our paths and is completely goof proof!

Data File Script
select 'alter database rename file '''||file_name||''' to ', '''/ora01/oradata/sid/'|| substr(file_name, instr(file_name,'/',-1)+1) ||''';' from dba_data_Files

Redo Log Script
select 'alter database rename file '''||member||''' to ', '''/ora01/oradata/sid/'|| substr(member, instr(member,'/',-1)+1) ||''';' from v$logfile;

Step 2
SHUTDOWN the target database and STARTUP MOUNT.
Run each of the scripts your have created.

SQL> alter database open;

Step 3
Check to make sure all your data files are pointing to the new mount points.

SQL> select file_name from dba_data_files;

If you just needed to rename the paths of your data files, you are done. If you also need to rename the database, you have one more step.

Step 4
After you have verified that all the datafiles and redo logs point to the correct path, back up the control file to trace.

SQL> alter database backup controlfile to trace;

Because your data files have already been renamed to use the new database name, all you have to do is focus on the renaming the database in the script and running the rebuild control file script. This process reduces the stress of rebuilding the control file and is an easy way to rename your data files, even if you have no idea where they reside.

Get IP and Name of Server

```
--Get Server Name and IP Address
declare
  v_host_name v$instance.host_name%type;
  v_ip_address varchar2(50);
begin
  select host_name into v_host_name from v$instance;
  dbms_output.put_line('the database server name is ' || v_host_name);
  SELECT UTL_INADDR.GET_HOST_ADDRESS(v_host_name) into v_ip_address FROM DUAL;
  dbms_output.put_line('the database server ip address is ' || v_ip_address);
end;
/

SELECT host_name, UTL_INADDR.GET_HOST_ADDRESS(host_name) ip FROM v$instance;
```

Creating Scott Example Schema
If it is not already present create the SCOTT schema:
conn sys/password as sysdba
@$ORACLE_HOME/rdbms/admin/utlsampl.sql

Create a PLAN_TABLE if it does not already exist:

conn sys/password as sysdba
@$ORACLE_HOME/rdbms/admin/utlxplan.sql
CREATE PUBLIC SYNONYM plan_table FOR sys.plan_table;
GRANT INSERT, UPDATE, DELETE, SELECT ON sys.plan_table TO public;

Install Oracle Java Virtual Machine

How does one install the Oracle JServer/JVM Option?
* Make sure your database is started with large java_pool_size (>20M) and shared_pool_size (>50M) INIT.ORA parameter values.
* Run the $ORACLE_HOME/javavm/install/initjvm.sql script from SYS AS SYSDBA to install the Oracle JServer Option on a database.
* Grant JAVAUSERPRIV to users that wants to use Java:
SQL> GRANT JAVAUSERPRIV TO FRAUDGUARD;

If you want to uninstall it just execute the following as SYS:
$ORACLE_HOME/rmjvm.sql

Pivot Table

```
create  table VOL_MONTHLY_REPORT
(
  COUNT_ITEMS_PROT                    VARCHAR2(16)    ,
  COUNT_ITEMS_OPEN                    NUMBER          ,
  DATECREATED                         DATE            ,
  APPLICATIONID                       NUMBER(2, 0)    ,
  ARCHIVEDDATE                        DATE            )
/

Select DATECREATED, APPLICATIONID, COUNT_ITEMS_OPEN
From VOL_MONTHLY_REPORT
order by 1,2;

DATECREATED        APPLICATIONID COUNT_ITEMS_OPEN
------------------ ------------- ----------------
01/JAN/07 00:00:00             3               10
01/JAN/07 00:00:00             4               20
02/JAN/07 00:00:00             3               33
02/JAN/07 00:00:00             4               44


SELECT DATECREATED,
DECODE (APPLICATIONID, 3, COUNT_ITEMS_OPEN, NULL) VRA,
DECODE (APPLICATIONID, 4, COUNT_ITEMS_OPEN, NULL) IRD
FROM (SELECT DATECREATED, APPLICATIONID, COUNT_ITEMS_OPEN FROM
VOL_MONTHLY_REPORT);
```

```
DATECREATED               VRA        IRD
----------------- ---------- ----------
01/JAN/07 00:00:00         10
01/JAN/07 00:00:00                    20
02/JAN/07 00:00:00         33
02/JAN/07 00:00:00                    44
```

```
select DATECREATED, VRA, IRD
from (select DATECREATED,
        max(case when APPLICATIONID=3 then COUNT_ITEMS_OPEN else null end) VRA,
        max(case when APPLICATIONID=4 then COUNT_ITEMS_OPEN else null end) IRD
      from VOL_MONTHLY_REPORT
      group by DATECREATED);
```

```
DATECREATED               VRA        IRD
----------------- ---------- ----------
01/JAN/07 00:00:00         10         20
02/JAN/07 00:00:00         33         44
```

What Code is executed by a Session?

Use the following code to see the ACTIVE Sessions:

```
set echo off;
set termout on;
set linesize 80;
set pagesize 60;
select substr(a.spid,1,9) pid, substr(b.sid,1,5) sid, substr(b.serial#,1,5)
ser#,
       substr(b.machine,1,6) box, substr(b.username,1,10) username,
--       b.server,
       substr(b.osuser,1,8) os_user, substr(b.program,1,30) program
from v$session b, v$process a
 where b.paddr = a.addr
   and type='USER'
   and status = 'ACTIVE'
order by spid;
```

Use the following script to retrieve the SQL statement for a particular user:

```
select SQL_TEXT from V$SQLAREA
  where (address, hash_value)
     IN (select SQL_ADDRESS, SQL_HASH_VALUE
              from V$SESSION
              where SID = &SID);
```

Export Data to Excel

set feed off markup html on spool on
set pagesize 0

```
alter session set nls_date_format='YYYY-MM-DD';
spool emp.xls
select * from emp;
spool off
set markup html off spool off
```

ORA-00257 archiver error. Connect internal, until freed

There are two options to resolve this.
Either step below should free the archiver process and archive logging should start:
1. Clean (delete) archive log files to make some space.
Connect as SYS and issue the following command to archive the remaining logs:
ALTER SYSTEM ARCHIVE LOG ALL,
This command archives all unarchived log files. This should cause the archiver to start.

OR
1. a) Query v$log and the operating system file system and determine which archive log has the latest date. This file might have a filesize of 0.
b) If the latest log has a filesize of 0, rename the archive log file. If Oracle is not able to create the file successfully (filesize of 0), then it is recommended that a full backup be taken, to ensure that the backups are valid.

Managing Redo Log Files and Groups

--First Review the groups

```
select group#,status from v$log;
    GROUP# STATUS
---------- ----------------
         1 INACTIVE
         2 CURRENT
         3 INACTIVE
```

-- Then Review the members per group

```
select group#, member from v$logfile;
    GROUP#  MEMBER
----------  ------------------------------------------------------------
         1  C:\ORACLE\PRODUCT\10.2.0\ORADATA\DEV10G2\REDO01.LOG
         2  C:\ORACLE\PRODUCT\10.2.0\ORADATA\DEV10G2\REDO02.LOG
         3  C:\ORACLE\PRODUCT\10.2.0\ORADATA\DEV10G2\REDO03.LOG
```

-- In order to remove a group to increase its size, that group must be inactive:

```
alter database drop logfile group 2;
alter database add logfile group 2 ('/opt/oracle/oradata/XGUARD/redo02.log')
size 600M reuse;
```

--You can force the switch by using:

```
alter system switch logfile;
```

Working With Sequences

Alter Sequence

Change Increment
ALTER SEQUENCE <sequence_name> INCREMENT BY <integer>;
ALTER SEQUENCE seq_inc_by_ten INCREMENT BY 20;

Change Max Value
ALTER SEQUENCE <sequence_name> MAX VALUE <integer>
ALTER SEQUENCE seq_maxval MAXVALUE 10;

Change Cycle
ALTER SEQUENCE <sequence_name> <CYCLE | NOCYCLE>
ALTER SEQUENCE seq_cycle NOCYCLE;

Change Cache
ALTER SEQUENCE <sequence_name> CACHE <integer> | NOCACHE
ALTER SEQUENCE seq_cache NOCACHE;

Change Order
ALTER SEQUENCE <sequence_name> <ORDER | NOORDER>
ALTER SEQUENCE seq_order NOORDER;

Drop Sequence
DROP SEQUENCE <sequence_name>;
DROP SEQUENCE seq_cache;

Sequence Resets
By finding out the current value of the sequence and altering the increment by to be negative that number and selecting the sequence once -- the sequence can be reset to 0.
If any session attempts to use the sequence while this is happening an ORA-08004 error will be generated.

```
CREATE SEQUENCE seq;
SELECT seq.NEXTVAL FROM dual;
SELECT seq.NEXTVAL FROM dual;
SELECT seq.NEXTVAL FROM dual;

COLUMN S new_val inc;
SELECT seq.NEXTVAL S FROM dual;

ALTER SEQUENCE seq INCREMENT BY -&inc MINVALUE 0;
SELECT seq.NEXTVAL S FROM dual;

ALTER SEQUENCE seq increment by 1;
SELECT seq.NEXTVAL FROM dual;
/
```

Sequence Related Queries
Last Number Selected From Sequence

```
SELECT sequence_name, last_number
   FROM user_sequences;
```

Next Number From Sequence

```
SELECT sequence_name, (last_number + increment_by) NEXT_VALUE
FROM user_sequences;
```