Oracle8*i*

Administrator’s Reference

Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)

January 2001

Part No. A87438_01

Oracle8*i* Administrator's Reference,
Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)

Part No. A87438_01

Contents

1 Administering Oracle8*i*

2 Tuning Oracle8*i*

3 Administering SQL*Plus

4 Using Oracle Precompilers and the Oracle Call Interface

5 Configuring Net8

6 Running Oracle Data Option Demonstrations

A Optimal Flexible Architecture

Index

Send Us Your Comments

Oracle 8i Administrator's Reference, Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)

A87438_01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Email - osdwrite@us.oracle.com
- FAX - 650.506.7303 Attn: Corinne Arne
- Postal service:
  Corinne Arne
  Oracle Corporation
  500 Oracle Parkway, Mailstop 1op4
  Redwood Shores, CA 94065
  USA

If you would like a reply, please provide your name, address, and telephone number.

If you have problems with the software, please contact your local Oracle Support Services Center.

Preface

Purpose of this Guide

This guide, with the *Oracle8i Release Notes Release 3 (8.1.7) for Intel UNIX (DG/UX Intel and SCO UnixWare)*, the *Oracle8i Release Notes Release 3 (8.1.7) for Solaris Intel UNIX*, and the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*, describes how to install and configure Oracle8*i* release 3 (8.1.7) on Intel UNIX. For additional product information, see the *Oracle8i Generic Documentation Se*t.

Audience

This guide is intended for anyone responsible for installing Oracle8*i* release 3 (8.1.7) on Intel UNIX systems.

Oracle8*i* and Oracle8*i* Enterprise Edition

Unless otherwise noted, the information in this guide is common to both Oracle8*i* and Oracle8*i* Enterprise Edition.

Typographic Conventions

The following typographic conventions are used in this guide:

`monospace`	Monospace type indicates UNIX commands, directory names, user names, path names, file names, and text quoted from screen output.

brackets [ ]	Words enclosed in brackets indicate button and key names (for example, Press [Return]). Note that brackets have a different meaning when used in command syntax.
italics	Italic type indicates variables, including variable portions of file names. It is also used for emphasis.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment variables.

Command Syntax

All examples that show commands or file contents appear in monospace font and assume the use of the Bourne shell. The dollar prompt ($) at the beginning of the UNIX command examples is the default UNIX command prompt. Do not enter it in the examples.

backslash \	A backslash indicates a command that is too long to fit on a single line. Enter the line as printed (with a backslash) or enter it as a single line without a backslash:
	`dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \`
	`count=10000`

braces { }	Braces indicate required items: `.DEFINE {macro1}`
brackets [ ]	Brackets indicate optional items: `cvtcrt` *`termname`* [*`outfile`*] Note that brackets have a different meaning when used in regular text.
ellipses ...	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname` *`value1 value2 ... valueN`*
italics	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line |	A vertical line indicates a choice within braces or brackets: *`SIZE filesize [K|M]`*

Accessing Installed Documentation

Oracle8*i* release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel) documentation includes this guide and the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*. You can install documentation in HTML and PDF (Adobe Portable Document Format, which requires Acrobat Reader) formats. Platform-specific documentation files are installed from the Oracle8*i* CD-ROM. Generic documentation files are installed from the Oracle8*i* On-Line Generic Documentation CD-ROM. The location of the installed documentation files is determined according to the following rules:

- If the ORACLE_DOC environment variable is defined, the files are installed in that directory.
- If the ORACLE_DOC environment variable is not defined but the ORACLE_BASE environment variable is defined, the files are installed in the `$ORACLE_BASE/doc` directory.
- If neither the ORACLE_DOC nor the ORACLE_BASE environment variables are defined, the files are installed in the `$ORACLE_HOME/doc` directory.

To access the installed documentation, use a web browser to open either the `index.htm` or `products.htm` files (the latter file does not a require a frames-enabled browser). If you prefer paper documentation, you can print the PDF files.

Oracle Product Documentation

Oracle8*i* product documentation is available on the Oracle8*i* On-Line Generic Documentation CD-ROM. Instructions for accessing and installing the documents on the CD-ROM are found in the README file in the top level directory of the CD-ROM.

Oracle Information Navigator

Oracle Information Navigator is a Java-based search and navigation utility provided with Oracle online documentation. If you are using a Java-enabled browser, Information Navigator is launched automatically when you use a web browser to open the `index.htm` file at the top level of the CD-ROM. You can use the Information Navigator with Oracle documentation, whether you are reading from the CD-ROM or from installed files.

Related Documentation

If you are unfamiliar with the concepts or terminology associated with relational database management systems, read Chapter 1 in *Oracle8i Concepts* before beginning your installation.

Information on administering your system and tuning a production database is provided in these documents:

- *Oracle8i Installation Guide for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*
- *Oracle8i Release Notes Release 3 (8.1.7) for Intel UNIX (DG/UX Intel and SCO UnixWare)*
- *Oracle8i Release Notes Release 3 (8.1.7) for Solaris Intel UNIX*
- *Net8 Administrator's Guide*
- *Oracle8i Designing and Tuning for Performance*

Information on migrating or upgrading from a previous release of the Oracle Server is provided in *Oracle8i Migration*.

Information on installing Oracle Internet Directory is provided in the *Oracle Internet Directory Installation Guide*.

Oracle Services and Support

Oracle Corporation offers a wide range of support services, including Oracle Education courses, Oracle Consulting services, and Oracle Worldwide Customer Support. Information on Oracle products and global services is available on the Internet, from:

`http://www.oracle.com`

The following sections provide URLs for selected services.

Oracle Support Services

Technical Support contact information worldwide is listed at:

`http://www.oracle.com/support`

This site provides templates to help you prepare information on your problem before you call. You also need your CSI number (if applicable) or complete contact details, including any special project information relating to the problem.

Products and Documentation

Customers in the U.S.A. can contact the Oracle Store at:

`http://store.oracle.com`

Links to stores in other countries are provided from this site.

Product documentation is available at:

`http://docs.oracle.com`

Customer Service

Global Customer Service contacts are listed at:

`http://www.oracle.com/support/`

Education and Training

Information on training and worldwide training schedules is available at:

`http://education.oracle.com`

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at:

`http://technet.oracle.com`

OTN provides technical papers, code samples, product documentation, self-service developer support, and Oracle key developer products to enable rapid development and deployment of applications built on Oracle technology.

Support for the Hearing-Impaired

To avail of TTY access to Oracle Support Services within the United States of America, phone 1-800-446-2398.

1

Administering Oracle8*i*

This chapter describes tasks specific to administering Oracle8*i* on Intel UNIX. It contains the following sections:

- Overview
- Environment Variables
- Product Executables
- System Global Area
- Calculating the Size of the SGA
- Oracle8i Memory Requirements and Usage
- Server Resource Limits
- Database Limits
- Special Accounts and Groups
- Security
- Embedded PL/SQL Gateway
- Oracle HTTP Server (Solaris Intel Only)
- Demonstrations Files

Overview

You must set Oracle8*i* environment variables, parameters, memory, and user settings for Oracle8*i* to work. This chapter describes the various settings for Oracle8*i* on Intel UNIX.

In Oracle8*i* files and programs, a question mark (?) represents the value of the ORACLE_HOME environment variable. For example, Oracle8*i* expands the question mark in the following SQL statement to the full pathname of the Oracle home directory:

```
ALTER TABLESPACE TEMP ADD DATAFILE '?/DBS/DBS2.DBF' SIZE 2M
```

The @ sign represents the ORACLE_SID environment variable. For example, to indicate a file belonging to the current instance, enter:

```
ALTER TABLESPACE TABLESPACE_NAME ADD DATAFILE DBS2.DBF
```

Environment Variables

This section describes the most commonly-used Oracle8*i* and UNIX environment variables.

To display the current value of an environment variable, use the `env` command. For example, to display the value of the ORACLE_SID environment variable, enter:

```
$ echo $ORACLE_SID
```

Note: Use the `env` command to show the value of environment variables that have been exported to the environment. Bourne shell and Korn shell can set values without exporting them.

You must define some of these variables before installing Oracle8*i*. The required variable settings are listed in the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*.

Oracle8*i* Environment Variables

Table 1–1 provides the syntax for, and examples of, environment variables used by Oracle8*i*.

Table 1–1 Oracle8i Environment Variables on UNIX

Variable	Detail	Definition
EPC_DISABLED	Function	Disables Oracle Trace.
	Syntax	`[true |false]`
NLS_LANG	Function	Specifies the language, territory and character set of the client environment. The character set specified in NLS_LANG must match the character set of the terminal or terminal emulator. The character set specified in NLS_LANG can be different from the database character set, in which case Oracle automatically converts the character set. See the *Oracle8i National Language Support Guide* for a list of values.
	Syntax	*`language_territory.characterset`*
	Example	`french_france.we8dec`
ORA_NLS33	Function	Specifies the directory where language, territory, character set, and linguistic definition files are stored.
	Syntax	*`directory_path_same_as_oracle_base`*
	Example	`$ORACLE_HOME/ocommon/nls/admin/data`
ORACLE_BASE	Function	Specifies the base of the Oracle directory structure for Optimal Flexible Architecture (OFA) compliant databases.
	Syntax	*`directory_paths`*
	Example	`/u01/app/oracle`
ORACLE_HOME	Function	Specifies the directory containing the Oracle software.
	Syntax	*`directory_paths`*
	Example	`$ORACLE_BASE/product/8.1.7`

Table 1–1 Oracle8i Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
ORACLE_PATH	Function	Specifies the search path for files used by Oracle applications, such as `*.sql` (SQL*Plus), `*.frm` (Oracle Forms), and `*.rpt` (Oracle Reports). If the full path to the file is not specified, or is not in the current directory, the Oracle application uses ORACLE_PATH to locate the file.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/u01/oracle/adhoc/8.1.7/bin:.` **Note:** The period adds the current working directory to the search path.
ORACLE_SID	Function	Specifies the Oracle system identifier.
	Syntax	A string of numbers and characters that must begin with a letter. Oracle Corporation recommends a maximum of eight characters. For more information, see the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel).*
	Example	`SAL1`
ORACLE_TRACE	Function	Enables tracing of Bourne shell scripts during an installation. If set to T, many Oracle shell scripts run with the `set -x` flag on.
	Syntax	T or not T.
ORAENV_ASK	Function	Controls whether the `coraenv` or `oraenv` script prompts for ORACLE_SID or ORACLE_HOME. If the value is `NO`, the scripts do not prompt; otherwise they do.
	Syntax	*`string`*
	Example	NO or not NO.
SQLPATH	Function	Specifies the directory or list of directories that SQL*Plus searches for a `login.sql` file.
	Syntax	colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/home:/home/oracle:/u01/oracle`

Table 1–1 Oracle8i Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
TNS_ADMIN	Function	Specifies the directory containing the Net8 configuration files.
	Syntax	*`directory_paths`*
	Range of Values	Any directory; for more information, see the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel).*
	Example	`$ORACLE_HOME/network/admin`
TWO_TASK	Function	Specifies the default Net8 connect string descriptor alias defined in the `tnsnames.ora` file.
	Syntax	Any available network alias.
	Range of Values	Any valid Net8 alias defined in the `tnsnames.ora` file.
	Example	PRODDB_TCP

Note: Do not define environment variables with names that are identical to names of Oracle Server processes, for example: arch, pmon, and dbwr.

UNIX Environment Variables

Table 1–2 provides the syntax for, and examples of, UNIX environment variables used with Oracle8*i*.

Table 1–2 UNIX Environment Variables Used with Oracle8i

Variable	Detail	Definition
CLASSPATH	Function	Used with Java applications. This variable differs with the Java application. See the product documentation for your Java application for more information.
	Syntax	*`directory_paths`*
	Example	There is no default setting. CLASSPATH must include the *`JRE_Location`* and `$ORACLE_HOME/product/jlib` directories In the preceding examples, *`JRE_Location`* is defined as `$ORACLE_HOME/JRE.`

Table 1–2 UNIX Environment Variables Used with Oracle8i (Cont.)

Variable	**Detail**	**Definition**
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. See the X Windows documentation of the vendor for details.
	Syntax	`hostname:display` and the *hostname* is your system name (either IP address or alias); *display* is the monitor number. If you have a single monitor, the number is 0.

	Examples	`135.287.222.12:0`
		`bambi:0`

HOME	Function	The user's home directory.
	Syntax	*`directory_path`*
	Example	`/oracle/home`
JAVA_HOME (SCO UnixWare only)	Function	`/usr/java`
	Syntax	*`directory_path`*
	Example	`/usr/java`
LANG or LANGUAGE	Function	Specifies the language and character set used by the operating system for messages and other output. See the operating system documentation and the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel).*
	Syntax	`LANG=value`
	Example	`LANG=C`
LPDEST	Function	Specifies the user's default printer.
	Syntax	printer_name
	Example	`docprinter`
LD_LIBRARY_PATH	Function	Used by the shared library loader (`ld.so`) at runtime to find shared object libraries. See `man` pages on `ld.so` for details. On SCO UnixWare, must include `/usr/java/lib`.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib`
PATH	Function	Used by the shell to locate executable programs; must include `$ORACLE_HOME/bin`. On SCO UnixWare, must include `/usr/java/bin.`

Table 1–2 UNIX Environment Variables Used with Oracle8i (Cont.)

Variable	Detail	Definition
	Syntax	Colon-separated list of directories: `directory1:directory2:directory3`

	Example	`/bin:/usr/bin:/usr/local/bin:`
		`/usr/bin/X11:$ORACLE_HOME/bin:$HOME/bin:.`

		Note: The period adds the current working directory to the search path.
PRINTER	Function	Selects the default printer.
	Syntax	*`printer_name`*
	Example	`docprinter`
SHELL	Function	Specifies the command interpreter used during a `HOST` command.
	Syntax	*`shell`*
	Range of Values	`/bin/sh` or `/bin/csh` or `/bin/ksh` or any other command interpreter supplied with Intel UNIX
	Example	`/bin/sh`
TERM	Function	Used by Oracle Toolkit II character mode tools and other UNIX tools to determine terminal types.
	Example	`vt100`
	Syntax	String
	Example	`$THREADS_FLAG= native`
TMPDIR	Function	Specifies the default directory for temporary files; if set, temporary files are stored in this directory.
	Syntax	*`directory_path`*
	Example	`/u02/oracle/tmp`
XENVIRONMENT	Function	Specifies a file containing X-Windows system resource definitions. See your X-Windows documentation for more information.

Setting a Common Environment

The following section describes how to set a common UNIX environment.

The oraenv Command File

The `oraenv` (`coraenv` for the C shell) command file is created during installation. It contains values for Oracle environment variables and provides:

- A central means of updating all user accounts with database changes
- A mechanism for switching between Oracle8*i* databases

You may find yourself frequently adding and removing databases from your development system or your users may be switching between several different Oracle databases installed on the same system. With `oraenv` each user shell startup file profile calls the `oraenv` command file.

Local bin Directory

Place `oraenv` (or `coraenv`) and `dbhome` scripts in a local `bin` directory, separate from the Oracle software home directory, to ensure that these files are accessible to all users. Doing this also ensures that the `oraenv` script continues to work even if you change the path to specify a different Oracle home directory. The local `bin` directory is specified by the `root.sh` script, which you run after you install Oracle8*i*. The default location for the local `bin` directory on Intel UNIX is `/usr/local/bin`.

Switching Between Databases

To switch from one database or database instance to another, call the `oraenv` routine. Reply to the prompt with the value of the ORACLE_SID environment variable of the database to which you are switching. Always provide the full path of the `oraenv` command file. For example:

```
$ . /usr/local/bin/oraenv
ORACLE_SID= [default]? sid
```

Setting and Exporting the Value of a Variable in a Current Session

Use the `env` command to show the environment variable values that have been exported to the environment. The Bourne shell and Korn shell can set values without exporting them.

For the Bourne or Korn shell, enter:

```
$ ORACLE_SID=test
$ export ORACLE_SID
```

For the C shell, enter:

```
% setenv ORACLE_SID test
```

In the preceding example, *`test`* is the value of the ORACLE_SID environment variable.

Setting the System Time

The TZ environment variable sets the time zone. It enables you to adjust the clock for daylight saving time changes or different time zones. The adjusted time is used to time-stamp files, produce the output of the `date` command, and obtain the current SYSDATE.

Oracle Corporation recommends that you do not change your personal TZ value. Using different values of TZ such as GMT+24 might change the day a transaction is recorded. This affects Oracle applications that use SYSDATE, such as Oracle Financials. To avoid this problem, use sequence numbers to order a table instead of date columns.

Product Executables

You can manually relink your product executables using a relink shell script located in the `$ORACLE_HOME/bin` directory. Relinking is necessary after applying any operating system patches or after an operating system upgrade.

The relink script manually relinks Oracle product executables depending on the products that have been installed in the Oracle home directory.

To relink product executables, enter the following command, where *`parameter`* is one of the values listed in Table 1–3:

```
$ relink parameter
```

Table 1–3 Relink Script Parameters

Value	**Description**
`all`	Every product executable that has been installed
`oracle`	Oracle database executable only
`network`	`net_client`, `net_server`, `nau`, `cman`, `cnames`
`client`	`net_client`, `otrace`, `plsql`, `client_sharedlib`
`interMedia`	`ctx`, `ordimg`, `ordaud`, `ordvir`, `md`
`precomp`	All precompilers that have been installed
`utilities`	Utilities
`oemagent`	`oemagent`, `odg`

> **Note:** Shut down Oracle Intelligent Agent and other Oracle programs in this Oracle home directory before relinking the database executable.

System Global Area

The System Global Area (SGA) is the Oracle structure that is stored in shared memory. It contains static data structures, locks, and data buffers. Sufficient shared memory must be available to each Oracle process to address the entire SGA.

The maximum size of a single shared memory segment is specified by the `SHMMAX` parameter. For example, if SHMMAX is 132 MB and the SGA is 528 MB, the SGA requires four shared memory segments.

On Solaris Intel, this setting is located in the `/etc/system` directory.

On SCO UnixWare, this setting is located in the `/etc/conf/cf.d/stune` directory.

On DG/UX Intel, this setting is located in the `/var/build/system.`*`hostname`* directory.

The recommended value for `SHMMAX` is 4,294,967,296 regardless of the actual memory installed on the system.

If the size of the SGA exceeds the maximum size of a shared memory segment (SHMMAX), The value of the SHMSEG parameter is the maximum number of segments that can be attached by a process.

Note: Intimate Shared Memory (ISM) may cause problems when the value of the SHMMAX parameter is smaller than the database SGA size (Solaris Intel only).

Set the following initialization file parameters to control the size of the SGA:

- DB_BLOCK_BUFFERS
- DB_BLOCK_SIZE
- SORT_AREA_SIZE
- SHARED_POOL_SIZE
- JAVA_POOL_SIZE
- LARGE_POOL_SIZE

Use caution when setting values for these parameters. When values are set too high, too much of the computer's physical memory is devoted to shared memory, resulting in poor performance.

Note: On DG/UX Intel systems, set the value of the DB_BLOCK_BUFFERS parameter to at least 1024.

Calculating the Size of the SGA

You can determine the SGA size in one of the following ways:

- Calculate the approximate size of an instance's SGA using the following formula:

  (DB_BLOCK_BUFFERS × DB_BLOCK_SIZE)
  + SORT_AREA_SIZE
  + SHARED_POOL_SIZE
  + LOG_BUFFER
  + JAVA_POOL_SIZE

- Display the size of the SGA for a running database using the SQL*Plus SHOW SGA command. The result is shown in bytes.
- Determine the size of the SGA when you start your database system. The SGA size is displayed next to the heading Total System Global Area.

The address at which the SGA is attached affects the amount of virtual address space available for database buffers in the SGA and cursors in the user's application data area.

To relocate the SGA, perform the following steps:

1. Enter the following command:

   ```
   $ tstshm
   ```

   In the output from `tstshm`, the lines "Lowest shared memory address" and "Highest shared memory address" indicate the valid address range.

2. Check the "segment boundaries" output of the `tstshm` command to determine the valid virtual address boundaries at which a shared memory segment can be attached.

3. Change directory to the `$ORACLE_HOME/rdbms/lib` directory and enter the following command to generate the `ksms.s:`

   ```
   $ ORACLE_HOME/bin/genksms -b sgabeg > ksms.s
   ```

   In this example, *`sgabeg`* is the starting address of the SGA (which defaults to 0x20000000). It should fall within the range determined in step 2.

   Never set the starting address below 0x01000000. On most systems, this leaves approximately 7 MB for data segments. This amount must allow enough memory for the SORT_AREA_SIZE parameter and similar items.

   You might receive the following error messages if you reduced the value of the starting address:

   - `ORA-4030: out of process memory when trying to allocate %s bytes (%s,%s)`
   - `ORA-7324`: `smpall`: `malloc error while allocating sga`

   If you receive one of these messages, then you probably lowered the starting address into an area which the SGA needs. Increase the starting address, and try again.

4. Shut down the existing Oracle instance.

5. Enter the following command to rebuild the Oracle executable in the `$ORACLE_HOME/rdbms/lib` directory:

   ```
   $ make -f ins_rdbms.mk ksms.o
   $ make -f ins_rdbms.mk ioracle
   ```

   Using the `ioracle` target:

   - Backs up the old executable (`oracle0`).
   - Assigns the correct privileges to the new Oracle executable.
   - Moves the new executable into the `$ORACLE_HOME/bin` directory.

   The result is a new Oracle kernel that loads the SGA at the address specified when generating the ksms.s file in step 3.

   > **See Also:** For more information on how the use of Java in the database affects SGA calculations, see the README file in the `$ORACLE_HOME/javavm/doc` directory.

Oracle8*i* Memory Requirements and Usage

Calculate memory usage requirements to determine the number of users that the system can support. This also helps to determine the physical memory and swap space requirements.

1. To calculate the memory requirements, use the following formula:

   size of the Oracle executable text
   + *size of the SGA*
   + *n* * (*size of tool executables private data section*
   + + *size of the Oracle executables uninitialized data section*
   + + 8192 bytes for the stack
   + + 2048 bytes for the processes user area)

   In the preceding example, *n* represents the number of background processes.

2. For each client-server connection, use the following formula to estimate virtual memory requirements:

   size of the Oracle executable data section
   + *size of the Oracle executables uninitialized data section*
   + 8192 bytes for the stack
   + 2048 bytes for processes user area
   + *cursor area required for the application*

3. Use the `size` command to estimate an executable's text size, private data section size, and uninitialized data section size (or DSS).

   Program text is only counted once, no matter how many times the program is run, because the Oracle executable text is shared.

   The `ps` command returns process size in pages; your system page size is architecture-dependent.

4. Use the `PAGESIZE` command to determine whether the size is 4096 or 8192 bytes. For each process, multiply the SZ value by the page size.

5. Finally, add the text size for the Oracle executable and every other Oracle tool executable running on the system to that subtotal. Remember to count executable sizes only once, regardless of how many times the executable is run.

For Solaris Intel:

To calculate the Oracle physical memory (background and shadow processes) usage:

1. Run the `pmap` command while the database is running and users are connected to it.

2. Add the shared sections (indicated by `read/write/exec/shared` and `read/exec`) for the `pmon` process.

3. Add the private section (indicated by `read/write/exec`) for each shadow and background process, including `pmon`.

   Background process names begin with `ora_`, and end with the `SID`. Shadow process names begin with `oracleSID`.

4. Use the `ps` command to determine process size in pages.

   System page size is architecture-dependent. Use the `pagesize` command to determine whether the size is 4096 or 8192 bytes.

Note: Do not use the `ps -elf` command because the SZ column repeats the shared portion of memory for each process shown, and makes it appear that Oracle is using much more memory than it actually is.

See Also: Refer to your Solaris Intel documentation for a list of available switches for the `ps` command.

5. For each process, multiply the SZ value by the page size.
6. Add the text size for the Oracle executable and every other Oracle tool executable running on the system to that subtotal.

   Remember to count executable sizes only once, regardless of how many times the executable is run.

Server Resource Limits

Intel UNIX inherits resource limits from the parent process (see `getrlimit(2)` in your operating system documentation). These limits apply to the Oracle8*i* shadow process that executes for user processes. The Intel UNIX default resource limits are high enough for any Oracle8*i* shadow or background process. However, if these limits are lowered, the Oracle8*i* system could be affected. Discuss this with your Intel UNIX system manager.

Disk quotas established for the `oracle` user can hinder the operation of the Oracle8*i* system. Confer with the Intel UNIX system manager before establishing disk quotas.

Database Limits

Table 1–4 lists the maximum and default values for parameters in a CREATE DATABASE or CREATE CONTROLFILE statement.

Note: Interdependencies between these parameters can affect allowable values.

Table 1–4 Determining the Size of Control Files

Parameter	Default Value	Maximum Value
MAXDATAFILES	30	65534
MAXINSTANCES	1	63
MAXLOGFILES	16	255
MAXLOGHISTORY	100	65534
MAXLOGMEMBERS	2	5

Table 1–5 lists the file size limits specific to Oracle8*i*.

Table 1–5 Specific Oracle8i File Size Limits

File Type	Maximum Size
datafiles db_block_size	8,589,932,544 if the DB_BLOCK_SIZE is set to 2048
datafiles db_block_size	17,179,865,088 if the DB_BLOCK_SIZE is set to 4096
datafiles db_block_size	34,359,730,176 if the DB_BLOCK_SIZE is set to 8192
datafiles db_block_size	68,719,460,352 if the DB_BLOCK_SIZE is set to 16384

Special Accounts and Groups

Table 1–6 describes the special UNIX accounts required by the Oracle server.

Table 1–6 Special UNIX Accounts

Account	Description
oracle	The oracle software owner represents the account that owns the Oracle8*i* software. This maintenance account requires DBA privileges to CREATE, STARTUP, SHUTDOWN, and CONNECT to the database as the INTERNAL user. The oracle software owner must never be the `root` user.
`root`	The `root` user is a special UNIX account with maximum privileges (superuser). This account is used to configure the UNIX kernel, configure and install networking software, and create user accounts and groups.

Table 1–7 describes the special Oracle Server accounts required by the Oracle server.

Table 1–7 Oracle Server Accounts

Account	Description
SYS	This is a standard Oracle8*i* account with DBA privileges automatically created during installation. The SYS account owns all the base tables for the data dictionary. This account is used by the DBA.
SYSTEM	This is a standard Oracle8*i* account with DBA privileges automatically created during installation. Additional tables or views can be created by the SYSTEM user. DBAs may log in as SYSTEM to monitor or maintain databases.

Table 1–8 describes the special group accounts required by the Oracle server.

Table 1–8 Special Group Accounts

Group	Description
`dba`	The `oracle` software owner is the only required member of the dba group. You can add any other UNIX user to the `dba` group. Members of this group have access to SQL*Plus specially privileged functions. If your account is not a member of the `dba` group, you must enter a password to connect as INTERNAL or gain access to the other administrative functions of SQL*Plus. The default OSDBA group is `dba`.

Table 1–8 Special Group Accounts (Cont.)

Group	Description
`oinstall`	All users installing Oracle8*i* in any Oracle home directory must belong to the same UNIX group. The Oracle Universal Installer inventory is shared by all Oracle home directories on a system and is group writable. Oracle recommends installing with `oinstall` as the primary group.
`oper`	This is an optional UNIX group. Members have database OPERATOR privileges. OPERATOR privileges are a restricted set of DBA privileges.
`root`	Only the `root` user should be a member of the `root` group.

Security

Oracle8*i* uses features of the UNIX operating system to provide a secure environment for users. These features include file ownership, group accounts, and the ability of a program to change its user ID upon execution.

The two-task architecture of Oracle8*i* improves security by dividing work (and address space) between the user program and the Oracle program. All database access is achieved through the shadow process and special authorizations in the Oracle program.

> **See Also:** For more information on security issues, see the *Oracle8i Administrator's Guide*.

Groups and Security

Oracle programs are divided into two sets for security purposes: those executable by all users (*other*, in UNIX terms), and those executable by DBAs only. To improve security, Oracle Corporation recommends:

- The primary group for the `oracle` account should be the `oinstall` group.
- The `oracle` account must have the `dba` group as a secondary group.
- Only the `oracle` account should belong to the `oinstall` group.

Security for Server Manager Commands

Oracle Corporation recommends that you restrict access to Server Manager. Only the Oracle software owner and `dba` group members should have access to the system privileges for STARTUP, SHUTDOWN, and CONNECT AS INTERNAL.

> **Caution:** System-privileged statements can damage your database if used incorrectly. Note that users who are not in the `dba` group can connect as INTERNAL if they have the password.

Security for Database Files

The user ID used to install Oracle8*i* owns the database files. The default user ID is `oracle`. Set the authorizations on these files to read / write by owner, and read-only for group or other users.

The `oracle` software owner should own the directories containing the database files. For added security, revoke read permission from group and other users.

To access the protected database files, the Oracle program must have its set user ID, `setuid`, bit on.

The Oracle Universal Installer automatically sets the permissions of the Oracle executable to:

```
-rwsr-s--x 1 oracle dba
```

The `s` in the user execute field means that when you execute the Oracle program, it has an effective user ID of `oracle`, regardless of the actual user ID of the person starting it.

If you must manually set the permissions on the Oracle executable, enter:

```
$ chmod 6751 $ORACLE_HOME/bin/oracle
```

Remote Passwords

You can administer a database from a remote computer such as a PC without having an operating system account. In this case, users are validated by using an Oracle8*i* password file, created and managed by the `orapwd` utility. You can also use password file validation on systems that support operating system accounts.

Local password files are located in the `$ORACLE_HOME/dbs` directory and contain the user name and password information for a single database. If there are multiple Oracle home directories on a system, each has a separate password file. To enable the database to use the password file, set the initialization file parameter REMOTE_LOGIN_PASSWORDFILE to EXCLUSIVE.

Access to a Database from a Remote PC

When there is an Oracle8*i* password file, networked PC users with DBA privileges can access the database as INTERNAL. Privileged users, who want to perform DBA functions on the database, can enter the appropriate SQL*Plus command from their computer, appending the `dba` user password to the command. For example:

```
SQL> CONNECT INTERNAL/DBA_PASSWORD@ALIAS AS {SYSDBA|SYSOPER}
```

Remote Authentication

The initialization file parameters shown in Table 1–9 control the behavior of remote connections through non-secure protocols:

Table 1–9 Parameters for Controlling Remote Connections

Parameter	Description
REMOTE_OS_AUTHENT	Enables or disables `ops$` connection
OS_AUTHENT_PREFIX	Used by `ops$` accounts
REMOTE_OS_ROLES	Enables or disables roles through remote connections

See Also: For information on resource limits, see `getrlimit(2)` in your operating system documentation.

Running the orapwd Utility

The `orapwd` utility, in the `$ORACLE_HOME/bin` directory, must be run by the `oracle` software owner. The syntax of this command is:

```
$ orapwd file=filename password=password entries=max_users
```

Table 1–10 describes this syntax:

Table 1–10 Syntax for Executing orapwd

Variable	Description
`filename`	Name of the file where password information is written. The name of the file must be `orapw`*`sid`*, and you must supply the full pathname. Its contents are encrypted and not user-readable. This parameter is mandatory.
`password`	Initial password you selected for INTERNAL and SYS. Change this password after you create the database using an ALTER USER statement. This parameter is mandatory.
`max_users`	Maximum number of users allowed to connect to the database as `SYSDBA` or `SYSOPER`. This parameter is mandatory only if you want this password file to be EXCLUSIVE. Set *`max_users`* to a higher number than you expect to require because if you must exceed this value, you must create a new password file.

Example of the orapwd Utility

The following is an example of the orapwd utility:

```
$ orapwd file=/u01/app/oracle/product/8.1.7/dbs/orapwV817
password=V817pw entries=30
```

Customizing the init*sid*.ora File

The default `init`*`sid`*`.ora` file is provided with the Oracle8*i* software. The Oracle Universal Installer creates it in the `$ORACLE_BASE/admin/`*`sid`*`/pfile` directory. You can modify or customize the Oracle8*i* installation. A sample of the `init`*`sid`*`.ora` file is located in the `$ORACLE_HOME/dbs` directory.

Table 1–11 lists default initialization parameter values on Intel UNIX. All Oracle8*i* instances assume these values if you do not specify different values for them in the `init`*`sid`*`.ora` file. Oracle Corporation recommends that you include in the `init`*`sid`*`.ora` file only those parameters that differ from the default initialization parameter values.

Use the SQL*Plus command SHOW PARAMETERS to display the current values of these parameters on the system.

Table 1–11 Initialization Parameters

Parameter	Default Value	Range of Values
BACKGROUND_DUMP_DEST	`?/rdbms/log`	Valid directory name
BITMAP_MERGE_AREA_SIZE	1048576	65536 to unlimited
COMMIT_POINT_STRENGTH	1	0 to 255
CONTROL_FILES	`?/dbs/cntrloracle_`*`sid`*`.dbf`	Valid file names
CREATE_BITMAP_AREA_SIZE	8388608	65536 to unlimited
DB_BLOCK_BUFFERS	100 MB of buffers	4 MB to unlimited
DB_BLOCK_SIZE	2048	2 KB to 16 KB
DB_FILES	200	1 to 2000000
DB_FILE_DIRECT_IO_COUNT	64	0 to 1048576 / block size
DB_FILE_MULTIBLOCK_READ_COUNT	8	1 to min(DB_BLOCK_BUFFERS / 4, 1048576 / DB_BLOCK_SIZE)
DISTRIBUTED_TRANSACTIONS	1 / 4 TRANSACTIONS	0 to unlimited
HASH_AREA_SIZE	2*SORT_AREA_SIZE	0 to unlimited
HASH_MULTIBLOCK_IO_COUNT	0 (self-tuned)	0 to min(127, DB_BLOCK_BUFFERS / 4, 1048576 / DB_BLOCK_SIZE)
JAVA_POOL_SIZE	20000000	1000000 to 1000000000
LOCK_SGA	FALSE	TRUE, FALSE
LOG_ARCHIVE_DEST	`null`	Valid directory names
LOG_ARCHIVE_FORMAT	"%t_%s.dbf"	Valid file names
LOG_BUFFER	512KB or 128KB multiplied by the number of systems, whichever is larger	66560 to unlimited
LOG_CHECKPOINT_INTERVAL	0	0 to unlimited
MTS_MAX_DISPATCHERS	5	The value of MTS_DISPATCHERS to the value of PROCESSES
MTS_MAX_SERVERS	2*MTS_SERVERS, if MTS_SERVERS > 20, else 20	The value of MTS_SERVERS to the value of PROCESSES
MTS_SERVERS	1, if MTS_DISPATCHERS is specified, else 0	1 to the value of PROCESSES
MTS_LISTENER_ADDRESS	ADDRESS=*address*	

Table 1–11 Initialization Parameters (Cont.)

Parameter	Default Value	Range of Values
NLS_LANGUAGE	AMERICAN	Valid language names
NLS_TERRITORY	AMERICA	Valid territory names
OBJECT_CACHE_MAX_SIZE_PERCENT	10	0 to unlimited
OBJECT_CACHE_OPTIMAL_SIZE	102400 100K (Solaris)	10 KB to unlimited
OPEN_CURSORS	50	1 to unlimited
OS_AUTHENT_PREFIX	ops$	Arbitrary string
PROCESSES	30, if not PARALLEL_AUTOMATIC_TUNING	6 to unlimited
SHARED_POOL_SIZE	64 MB on 64-bit systems, 8 MB on 32-bit systems	300000 to unlimited
SORT_AREA_SIZE	65536	0 to unlimited

See Also: For more information on initialization parameters see *Oracle8i Server Reference*, *Oracle8i Administrator's Guide* and *Oracle8i Tuning*.

Embedded PL/SQL Gateway

The embedded PL/SQL gateway is a gateway embedded in the Oracle8*i* server to provide native support for deploying database applications written is PL/SQL on the web. The embedded PL/SQL gateway is implemented as an Oracle Servlet Engine (OSE) servlet, and relies upon the existence and configuration of both the OSE and mod_ose, the Apache module which supports the OSE. The following instructions provide information on how to install and configure the gateway.

Overview

Two Apache modules, mod_ose and mod_plsql, support web applications developed with PL/SQL.

The mod_ose module acts as a request router for an OSE running within an Oracle8*i* instance. mod_ose enables stateful OSE applications by routing stateful requests through the middle tier and back to a specified OSE/Oracle8*i* instance. Because the embedded PL/SQL gateway is implemented as an OSE servlet running in the Oracle8*i* server, it is able to host stateful, as well as stateless, PL/SQL web applications. A stateful PL/SQL web application is one in which all database session states (for example, package and transaction) is preserved between requests.

The mod_plsql module is the name given to the PL/SQL gateway running within an Apache module in the middle tier server and executing PL/SQL procedures in a backend Oracle server using OCI. mod_plsql currently supports only stateless PL/SQL web applications.

> **See Also:** For information on developing web applications with PL/SQL, refer to *Using mod_plsql* which is generic PL/SQL gateway documentation.

Installing the Embedded PL/SQL Gateway

As with all OSE servlets, the embedded PL/SQL gateway must be loaded and published. To load and publish the embedded PL/SQL gateway servlet:

1. Enter the following command to load the servlet, connect to SQL*Plus as `sys`, and run the `initplgs.sql` SQL script:

   ```
   SQL> @RDBMS/ADMIN/INITPLGS.SQL
   ```

2. The name of the embedded PL/SQL gateway servlet is `oracle.plsql.web.PLSQLGatewayServlet`. To publish the servlet, enter the following command:

   ```
   % $ORACLE_HOME/jis/bin/unix/sess_sh -s http://OSE machine name:OSE port \
   number -u sys/change_on_install -c "publishservlet -virtualpath \
   pls/*/webdomains/contexts/default plsGatewway \
   SYS:oracle.plsql.web.PLSQLGatewayServlet"
   ```

   This command publishes the gateway servlet as `plsGateway` with a default context. The servlet can be accessed using the virtual path `/pls`. The following example shows a URL that might access a gateway servlet:

   ```
   http://dlsun240/pls/dadname/hello_world
   ```

> **See Also:** For more information on using and publishing servlets, see the *Oracle Servlet Engine User's Guide*.

Configuring Oracle PL/SQL Embedded Gateways

Configuration procedures for configuring the Apache server / mod_ose and the embedded PL / SQL gateway are beyond the scope of this reference.

> **See Also:** For information on configuring the Apache server / mod_ose, see *Oracle Servlet Engine User's Guide*.

Oracle HTTP Server (Solaris Intel Only)

The Oracle HTTP Server is Apache based. Administration tasks for the server require access to the local system on which the server is running, and in some cases, requires `root` access.

The Oracle HTTP Server is started automatically upon installation on port 7777. The name of the server binary is `httpd`. Verify that the server is running using the following command:

```
ps -edaf | grep httpd
```

Starting/Stopping the Oracle HTTP Server

If you modify the configuration, you must restart the server. You must be logged in as the `root` user to start the server with SSL enabled. To restart the server:

```
cd $ORACLE_HOME/Apache/Apache/bin
su root
./apachectl {start|startssl|graceful|stop}
```

> **Note:** If the server is restarted with SSL enabled, the default ports are 80 and 443.

Accessing the Default Initial Static Page

The default initial static page contains links to online documentation for Apache as well as demonstrations for each of the components. To access the initial static page, load one of the following URLs in your internet browser:

- For servers without SSL enabled, enter:

  `http://<ServerName>:7777/`

- For servers with SSL enable, enter:

  `http://<ServerName>/`

In the preceding example, *`ServerName`* is configured in the Apache server configuration file `httpd.conf`. To locate the configuration file, enter:

```
grep ServerName $ORACLE_HOME/Apache/Apache/conf/httpd.conf
```

Oracle HTTP Server Status

Several status pages are available. For security reasons, server status is disabled in the default server configuration files. To enable them, edit the appropriate configuration file and restart the server.

For the configuration file `$ORACLE_HOME/Apache/Apache/conf/httpd.conf`, use:

```
http://<ServerName>/server-status
http://<ServerName>/server-info
http://<ServerName>/perl-status
```

For the configuration file `$ORACLE_HOME/Apache/Jserv/etc/conf/jserv.conf`, use:

```
http://<ServerName>/jserv
```

Oracle HTTP Server Log Files

The following log files are generated by the server:

```
$ORACLE_HOME/Apache/Apache/logs/error_log
$ORACLE_HOME/Apache/Apache/logs/ssl_engine_log
$ORACLE_HOME/Apache/Jserv/logs/jserv.log
$ORACLE_HOME/Apache/Jserv/logs/mod_jserv.log
```

It is important to check them periodically to make sure that the server is working correctly. By default, the error log level is set to `warn` in the configuration files. You can change the default error level by editing the appropriate configuration file and restarting the server.

Demonstrations Files

This section describes how to build and run the SQL*Loader and PL/SQL demonstration programs installed with Oracle8*i*.

SQL*Loader Demonstrations

The following SQL*Loader demonstration files are included with Oracle8*i*. Run the demonstrations in numerical order:

`ulcase1` `ulcase3` `ulcase5` `ulcase7`

`ulcase2` `ulcase4` `ulcase6`

To create and run a demonstration

Run demonstrations while logged in as the user SCOTT/TIGER. Ensure that:

- The user SCOTT/TIGER has CONNECT and RESOURCE privileges
- The EMP and DEPT tables exist and are empty

In the following steps, *n* represents the demonstration number, listed in the previous section. To create and run a demonstration:

1. Run the `ulcasen.sql` script corresponding to the demonstration you want to run.

   ```
   $ sqlplus SCOTT/TIGER @ulcasen.sql
   ```

2. Load the demonstration data into the objects:

   ```
   $ sqlldr SCOTT/TIGER ulcasen.ctl
   ```

   - For the `ulcase2` demonstration, you do not have to run the `ulcase2.sql` script.
   - For the `ulcase6` demonstration, run the `ulcase6.sql` script, then enter the following at the command line:

     ```
     $ sqlldr SCOTT/TIGER ulcase6 DIRECT=true
     ```

   - For the ulcase7 demonstration, run the `ulcase7s.sql` script, then enter the following at the command line:

     ```
     $ sqlldr SCOTT/TIGER ulcase7
     ```

     After running the demonstration, run `ulcase7e.sql` to drop the insert trigger and global variable package.

Administering SQL*Loader

Oracle8*i* incorporates SQL*Loader functions. Demonstration and message files are in the `rdbms` directory.

The SQL*Loader control file includes the following additional file processing option strings, the default being `str`, which takes no argument:

```
[ "str" | "fix n" | "var n" ]
```

String	Description
`str`	Specifies a stream of records, each terminated by a newline character, which are read in one record at a time. This is the default.
`fix`	Indicates that the file consists of fixed-length records, each of which is *n* bytes long, where *n* is an integer value.
`var`	Indicates that the file consists of variable-length records, each of which is *n* bytes long, where *n* is an integer value specified in the first five characters of the record.

If you do not select the file processing option, the information is processed by default as a stream of records (STR). You might find that `fix` mode yields faster performance than the default STR mode because it does not scan for record terminators.

Newline Characters in Fixed Length Records

When using the FIX option to read a file containing fixed-length records, where each record is terminated by a newline character, include the length of the newline (one character) when specifying the record length to SQL *Loader.

For example, to read the following file specify FIX 4 instead of FIX 3 to account for the additional newline character:

```
AAA newline
BBB newline
CCC newline
```

If you do not terminate the last record in a file of fixed records with a newline character, do not terminate the other records with a newline character either. Similarly, if you terminate the last record with a newline character, terminate all records with a newline character.

Caution: Certain text editors, such as `vi`, automatically terminate the last record of a file with a newline character. This leads to inconsistencies if the other records in the file are not terminated with newline characters.

Removing Newline Characters

Use the `position(x:y)` function in the control file to discard the newline characters from fixed length records rather than loading them. For example, enter the following in your control file to discard newline characters from the fourth position:

```
load data
infile xyz.dat "fix 4"
into table abc
( dept position(01:03) char )
```

When this is done, newline characters are discarded because they are in the fourth position in each fixed-length record.

PL/SQL Demonstrations

PL/SQL includes a number of sample programs that you can load. Demonstration and message files are in the `rdbms` directory. The Oracle8*i* database must be open and mounted to work with the sample programs:

1. Run SQL*Plus and connect with the user/password SCOTT/TIGER:

   ```
   $ cd $ORACLE_HOME/plsql/demo
   $ sqlplus scott/tiger
   ```

2. To load the demonstrations, run `exampbld.sql` from SQL*Plus:

   ```
   SQL> @EXAMPBLD
   ```

Note: Build the demonstrations in any Oracle account with sufficient permissions. Run the demonstrations in the same account as they were built.

The following kernel demonstrations are available:

`examp1.sql`
`examp2.sql`
`examp3.sql`
`examp4.sql`
`extproc.sql`
`examp5.sql`
`examp6.sql`
`examp7.sql`
`examp8.sql`
`examp11.sql`
`examp12.sql`
`examp13.sql`
`examp14.sql`
`sample1.sql`
`sample2.sql`
`sample3.sql`
`sample4.sql`

To run the PL/SQL demonstrations, run SQL*Plus to connect to the database, using the same user/password used to create the demonstrations. Start the demonstration by typing an "at" sign (@) or the word START before the demonstration name. For example, to start the `examp1` demonstration, enter:

```
$ sqlplus SCOTT/TIGER
SQL> @EXAMPL
```

To build the precompiler PL/SQL demonstrations, enter:

```
$ cd $ORACLE_HOME/plsql/demo
$ make  -f demo_plsql.mk demos
```

The following precompiler demonstrations are available:

`examp9.pc`
`examp10.pc`
`sample5.pc`
`sample6.pc`

If you want to build a single demonstration, enter its name as the argument in the `make` command. For example, to build the `examp9.pc` executable, enter:

```
$ make  -f demo_plsql.mk examp9
```

To run the `examp9` demonstration from your current shell, enter:

```
$ ./examp9
```

To run the `extproc` demonstration:

1. Add the following line to the file, `tnsnames.ora`:

   ```
   (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=plsff))(CONNECT_DATA=(SID=extproc))
   )
   ```

2. Add the following line to the file, `listener.ora`:

   ```
   SC=(SID_NAME=extproc)(ORACLE_HOME=/u01/app/oracle/product/8.1.7)
   (PROGRAM=extproc))
   ```

3. From your SQL*Plus session, enter:

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> GRANT CREATE LIBRARY TO SCOTT;
Grant succeeded.
SQL> CONNECT SCOTT/TIGER
Connected.
SQL> CREATE LIBRARY DEMOLIB AS
'$ORACLE_HOME/plsql/demo/extproc.so';
Library created.
```

4. To run the demonstrations, enter:

```
SQL> CONNECT SCOTT/TIGER
Connected.
SQL> @EXTPROC
```

Database Examples

In the following examples, it is assumed that the local bin directory is `/usr/local/bin` and the production database is called PROD. In addition, ORAENV_ASK is reset to the default, Yes, after `oraenv` is executed. This ensures that the system prompts for a different ORACLE_SID environment variable the next time that the `oraenv` command is executed.

Note: Set the ORAENV_ASK environment variable to `no` to not prompt for the ORACLE_SID at startup.

If you have created a database manually instead of using Oracle Database Configuration Assistant, you must ensure that the system configuration is reflected in the `/var/opt/oracle/oratab` file.

For each server instance, add an entry in the following format:

```
ORACLE_SID:ORACLE_HOME:{Y|N}
```

The values `Y` or `N` indicate whether or not you want to activate the `dbstart` and `dbshut` scripts. The Oracle Database Configuration Assistant automatically adds an entry for each database it creates.

Example of Single Instance

For the Bourne or Korn shell, add or replace the following line in the `.profile` file:

```
. local_bin_directory/oraenv
```

with the following lines:

```
PATH=${PATH}:/usr/local/bin
ORACLE_SID=PROD
export PATH ORACLE_SID
ORAENV_ASK=NO
export ORAENV_ASK
. oraenv
unset ORAENV_ASK=
```

For the C shell, add or replace the following line in the `.cshrc` file:

```
source local_bin_directory/coraenv
```

with the following lines:

```
setenv PATH ${PATH}:/usr/local/bin
setenv ORACLE_SID PROD
setenv ORAENV_ASK NO
source /usr/local/bin/coraenv
unset ORAENV_ASK
```

Example of Multiple Instances

For multiple database instances, define the *`sid`* at startup.

For the Bourne or Korn shell**,** enter:

```
#!/usr/bin/sh
echo "The SIDs on this machine are:"
cat /var/opt/oracle/oratab | awk -F: ’{print $1}’ | grep -v "#"
ORAENV_ASK="YES"
.  /usr/local/bin/oraenv
```

For the C shell, enter:

```
#!/usr/bin/csh
echo "The SIDs on this machine are:"
cat /var/opt/oracle/oratab | awk -F: ’{print $1}’ | grep -v "#"
set ORAENV_ASK="YES"
source /usr/local/bin/coraenv
```

2

Tuning Oracle8*i*

The more your Oracle8*i* applications increase in complexity, the more you must tune the system to optimize performance and prevent data bottlenecks. This chapter describes how to configure your Oracle8*i* installation to optimize its performance. It contains the following sections:

- The Importance of Tuning
- Intel UNIX Tools
- SQL Scripts
- Tuning Memory Management
- Tuning Disk I/O
- Monitoring Disk Performance
- Tuning CPU Usage
- Tuning Oracle Resource Contention and UNIX Kernel Parameters
- Tuning and Specifying Block Size and File Size
- Tuning the Intel UNIX Buffer Cache Size
- Tuning the Intel UNIX Buffer Cache Size
- Using Raw Devices/Volumes

The Importance of Tuning

Oracle8*i* is a highly-optimizable software product. Frequent tuning optimizes system performance and prevents data bottlenecks. Although this chapter is written from the perspective of single-processor systems, most of the performance tuning tips provided here are also valid when using the parallel options and features available with Oracle8*i*.

Before tuning the system, observe its normal behavior using the Intel UNIX tools described in "Intel UNIX Tools" in the next section

Note: The Oracle Parallel Server (OPS) option is not supported on SCO UnixWare or Solaris Intel in this release.

See Also: For more information on tuning, see the *Oracle8i Parallel Server Concepts* and *Administration* and *Oracle8i Designing and Tuning for Performance* guides.

Intel UNIX Tools

Intel UNIX provides performance monitoring tools that you can use to assess database performance and determine database requirements. In addition to providing statistics for Oracle processes, these tools provide statistics for CPU usage, interrupts, swapping, paging, and context switching for the entire system.

See Also: For more information on UNIX tools, see *Oracle8i Designing and Tuning Performance*. Intel UNIX tools are described in the operating system documentation.

vmstat (Solaris Intel Only)

Use the `vmstat` utility to report process, virtual memory, disk, paging, and CPU activity on Solaris Intel, depending on the switches you supply with the command. The following command displays a summary of system activity, eight times, at five second intervals:

```
$ vmstat -S 5 8
```

The following example shows sample output from this command:

```
procs     memory            page            disk          faults      cpu
r b w   swap  free  si  so pi po fr de sr f0 s0 s1 s3   in   sy   cs us sy id
0 0 0   1892  5864   0   0  0  0  0  0  0  0  0  0  0   90   74   24  0  0 99
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   46   25   21  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   47   20   18  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  2   53   22   20  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   87   23   21  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   48   41   23  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   44   20   18  0  0 100
0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   51   71   24  0  0 100
```

The `w` column (under `procs`) shows the number of potential processes that have been swapped out (written to disk). If the value is not zero, swapping is occurring and your system has a memory shortage problem. The `si` and `so` columns indicate the number of swap-ins and swap-outs per second, respectively. Swap-outs should always be zero.

sar

Use the `sar` command to monitor swapping, paging, disk, and CPU activity, depending on the switches that you supply with the command. The following command displays a summary of paging activity ten times, at ten second intervals:

```
$ sar -p 10 10
```

The following example shows sample output from the command:

```
14:14:55  atch/s   pgin/s ppgin/s  pflt/s  vflt/s slock/s
14:15:25    0.00    0.00    0.00    0.00    0.00    0.00
14:15:35    0.00    0.00    0.00    0.00    0.00    0.00
14:15:45    0.00    0.00    0.00    0.00    0.00    0.00
14:15:55    0.00    0.00    0.00    0.00    0.00    0.00
14:16:05    0.00    0.00    0.00    0.00    0.00    0.00
14:16:15    0.00    0.00    0.00    0.00    0.00    0.00
14:16:25    0.00    0.00    0.00    0.00    0.00    0.00
14:16:35    0.00    0.00    0.00    0.00    0.00    0.00

Average     0.00    0.00    0.00    0.07    0.16    0.00
i
```

The following command displays a summary of disk activity every five seconds:

```
$ sar -d 5 5
```

iostat (Solaris Intel Only)

Use the `iostat` command on Solaris Intel systems to report terminal and disk activity depending on the switches you supply with the command. The report from `iostat` does not include disk request queues, but it shows which disks are busy. This information is useful when you must balance I/O loads.

The following command displays terminal and disk activity five times, at five second intervals:

```
$ iostat 5 5
```

The following example shows sample output from this command:

```
      tty         fd0           sd0           sd1           sd3          cpu
 tin tout Kps tps serv  Kps tps serv  Kps tps serv  Kps tps serv  us sy wt id
   0    1   0   0    0    0   0   31    0   0   18    3   0   42   0  0  0 99
   0   16   0   0    0    0   0    0    0   0    0    1   0   14   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    2   0   14   12   2   47   0  0  1 98
```

swap (Solaris Intel and SCO UnixWare Only)

Use the following command on Solaris Intel and SCO UnixWare systems to report information on swap space usage. A shortage of swap space can result in the system hanging and slow response time.

```
$ swap -l utility
```

The following example shows sample output from this command:

```
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t3d0s1   32,25      8 197592 162136
```

admswap (DG/UX Intel Only)

Use the following command on DG/UX Intel systems to report information on swap space usage:

```
$ admswap -o list -bu
```

The following example shows sample output from this command:

```
swapfile            blocks     in use
/dev/dsk/swap       200000      y
/dev/dsk/swap2      200000      y
```

mpstat (Solaris Intel and DG/UX Intel Only)

Use the following command on Solaris Intel and DG/UX Intel systems to report statistics for each processor :

```
$ mpstat
```

The following example shows sample output from this command:

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0    1    71   21   23    0    0    0    0    55    0   0   0  99
  2    0   0    1    71   21   22    0    0    0    0    54    0   0   0  99
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0    0    61   16   25    0    0    0    0    57    0   0   0 100
  2    1   0    0    72   16   24    0    0    0    0    59    0   0   0 100
```

Each row of the table represents the activity of one processor. The first row summarizes all activity since the last system re-boot; each subsequent row summarizes activity for the preceding interval. All values are events/per second unless otherwise noted. The arguments are for time interval between statistics and number of iterations.

truss (Solaris Intel and SCO UnixWare Only)

Use the following command on Solaris Intel and SCO UnixWare systems to trace the system calls and signals in the running process, if a process is hung on your system:

```
$ truss
```

Assuming the process id of the Oracle shadow process is 26986, sample output from the `truss` command is shown in the following example:

```
sigprocmask (SIG_BLOCK, 0x08046354, 0x0000000)              =0
sigprocmask (SIG_UNBLOCK, 0x08046354, 0x0000000)            =0
setcontext (0x08E6D554)
getcontext (0x08E6D554)
sigprocmask (SIG_BLOCK, 0x080465B4, 0x00000000              =0
hrtalarm (0x080464F8, 1)
sigprocmask (SIG_UNBLOCK, 0x080465B4, 0x00000000)           =1
semop (32328, 0x080465C8, 1) (sleeping...)
```

SQL Scripts

Oracle8*i* Release 3 includes a set of packages for database tuning called STATPACKS. For more information on STATPACKS, see *Oracle8i Designing and Tuning Performance*.

The `utlbstat.sql` and `utlestat.sql` scripts are used to monitor Oracle database performance and tune the System Global Area (SGA) data structures. For more information on these scripts, see *Oracle8i Designing and Tuning for Performance*. On Intel UNIX, the scripts are located in the `$ORACLE_HOME/rdbms/admin/` directory.

Tuning Memory Management

Start the memory tuning process by measuring paging and swapping space to determine how much memory is available.

The Oracle buffer manager ensures that the more frequently accessed data is cached longer. Monitoring the buffer manager and tuning the buffer cache can have a significant influence on Oracle performance. The optimal Oracle buffer size for your system depends on the overall system load and the relative priority of Oracle over other applications.

Allocate Sufficient Swap Space

Try to minimize swapping because it causes significant UNIX overhead. Use the `sar -w` command on Intel UNIX to check for swapping.

If your system is swapping and you must conserve memory:

- Avoid running unnecessary system daemon processes or application processes
- Decrease the number of database buffers to free some memory
- Decrease the number of UNIX file buffers, especially if you are using raw devices

---

Note: Commands for adding swap space are different on different platforms. On DG/UX Intel use `admswap -o list -vu`. On SCO UnixWare and Solaris Intel, use `swap -l`.

Control Paging

Paging might not present as serious a problem as swapping, because an entire program does not have to be stored in memory to run. A small number of page-outs might not noticeably affect the performance of your system.

To detect excessive paging, run measurements during periods of fast response or idle time to compare against measurements from periods of slow response.

Use the `sar -p` command to monitor paging. The following columns from the output of this command are important:

`vflt/s`	Indicates the number of address translation page faults. Address translation faults occur when a process references a valid page not in memory.
`rclm/s`	Indicates the number of valid pages that have been reclaimed and added to the free list by page-out activity. This value should be zero.

If your system consistently has excessive page-out activity, consider the following solutions:

- Install more memory
- Move some of the work to another system
- Configure your kernel to use less memory

Hold the SGA in a Single Shared Memory Segment

You cannot start the database without sufficient shared memory. If necessary, reconfigure the UNIX kernel to increase shared memory.

> **See Also:** For more information on the SGA, see "System Global Area" on page 1-10 and the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*.

Lock the SGA in Physical Memory

The primary function of the SGA is to cache database information. If the SGA begins paging to disk, caching becomes an overhead rather than a benefit. Some platform vendors provide techniques to lock the SGA into physical memory.

Although locking the SGA into physical memory can improve Oracle performance up to 20 percent, it can also reduce the performance of other applications on the same system due to less physical memory being available to those applications.

On DG/UX Intel, the DG/UX Intel kernel parameter, PERCENTLOCKABLE, specifies the number of physical RAM pages that can be locked in the system. The default setting for PERCENTLOCKABLE may not be sufficient for locking the entire SGA. Ensure that your DG/UX Intel kernel is configured with a high enough value for the PERCENTLOCKABLE parameter.

On DG/UX Intel and Solaris Intel systems, the owner of the database must be `root` to lock the shared memory.

To enable SGA locking in Oracle, set the value of the LOCK_SGA parameter to TRUE in the `init`*`sid`*`.ora` file.

Optimize Number of Database Buffers

The DB_BLOCK_SIZE parameter determines the size of the database buffers in the SGA. The DB_BLOCK_BUFFERS parameter is the memory parameter with the most direct effect on system performance. Optimize the number of database buffers to increase performance up to 200 percent.

Use the System Statistics Monitor in Server Manager to check the *hit ratio*. The hit ratio for the buffer cache is defined as:

$$\text{Hit Ratio} = \frac{\textit{Logical Reads} - \textit{Physical Reads}}{\textit{Logical Reads}}$$

In the preceding example, the *Logical Reads* represents *db block gets* + *consistent gets.*

If the hit ratio is less than 60 or 70 percent, increase the number of buffers in the cache by raising the value of the DB_BLOCK_BUFFERS parameter.

See Also: Alternatively, use the X$KCBRBH table to estimate the number of buffers based on statistics gathered from a running system. The X$KCBRBH table is described in *Oracle8 Server Tuning*. The System Statistics Monitor is described in the *Oracle Server Manager User's Guide*.

Indirect Database Buffers on DG/UX Intel and SCO UnixWare Systems

The maximum SGA size supported is 2.5 GB if direct database buffers are used, but you can have a much larger SGA by using indirect database buffers. Enabling indirect database buffers involves a slight overhead of mapping and unmapping buffers as they are required. Oracle Corporation recommends that you enable this feature on systems with more than 4 GB of physical memory to improve the buffer cache hit ratio. For optimal utilization of memory resources on the system, the DB_BLOCK_SIZE parameter should be a multiple of 4 K.

To enable indirect database buffers, set the USE_INDIRECT_DATA_BUFFERS initialization parameter buffers to TRUE.

The following restrictions apply:

1. Indirect database buffers are not supported with Oracle Parallel Server option.
2. Do not enable this feature if you plan to have LOBS (large objects) in your database.
3. Oracle SGA must be locked in physical memory. Refer to "Lock the SGA in Physical Memory" on page 2-8 for more details.
4. This feature is supported with Oracle Enterprise Edition only.

Tuning Disk I/O

I/O bottlenecks are the easiest performance problems to identify. Balance I/O evenly across all available disks to reduce disk access times. For smaller databases and those not using the Parallel Query option, ensure that different datafiles and tablespaces are distributed across the available disks.

Tune the Database Writer to Increase Write Bandwidth

Oracle offers asynchronous I/O, multiple DBWR processes, and I/O slaves as solutions to prevent database writer (DBWR) activity from becoming a bottleneck.

Asynchronous I/O

Asynchronous I/O allows processes to proceed with the next operation without having to wait after issuing a write. It therefore improves system performance by minimizing idle time. Intel UNIX supports asynchronous I/O on raw devices only.

> **Note:** For faster asynchronous I/O performance, Oracle SGA should be locked in the memory. Refer to "Lock the SGA in Physical Memory" on page 2-8 for more information.

Multiple DBWR Processes

The DB_WRITER_PROCESSES parameter controls the number of database writers in the `init`*`sid`*`.ora` file. For write-intensive applications, the DBWR bandwidth can be an overwhelming bottleneck. Measure the performance with different numbers of database writers to find the optimal number of database writers for your database. If you use DBWR_IO_SLAVES, only one database writer process is used, regardless of the setting for DB_WRITER_PROCESSES.

On DG/UX Intel systems, Oracle Corporation recommends setting the DB_WRITER_PROCESSES to the number of blocks in the system or higher.

I/O Slaves

I/O slaves are specialized processes whose only function is to perform I/O. They replace the Oracle7 feature, Multiple DBWRs. In fact, they are a generalization of Multiple DBWRs and can be deployed by other processes as well. They can operate whether or not asynchronous I/O is available. They are allocated memory from LARGE_POOL_SIZE, if set, otherwise they are allocated memory from shared memory buffers. I/O slaves include a set of initialization parameters that allow a degree of control over the way they operate.

Table 2–1 lists the initialization parameters that control the operation of asynchronous I/O and I/O slaves.

Table 2–1 Initialization Parameters for Asynchronous I/O and I/O Slaves

Parameter	Range of Values	Default Value
DISK_ASYNCH_IO	TRUE/FALSE	TRUE
TAPE_ASYNCH_IO	TRUE/FALSE	TRUE
BACKUP_DISK_IO_SLAVES	TRUE/FALSE	FALSE
BACKUP_TAPE_IO_SLAVES	TRUE/FALSE	FALSE
DBWR_IO_SLAVES	0 - 999	0
ARCH_IO_SLAVES	0 - 999	0
DB_WRITER_PROCESSES	1-10	1

There might be times when the use of asynchronous I/O is not desirable or not possible. The first two parameters in Table 2–1, DISK_ASYNCH_IO and TAPE_ASYNCH_IO, allow asynchronous I/O to be switched off respectively for disk and tape devices. Because the number of I/O slaves for each process type defaults to zero, no I/O slaves are deployed unless set.

Set the DBWR_IO_SLAVES parameter to greater than 0 if DISK_ASYNCH_IO or TAPE_ASYNCH_IO is disabled, otherwise DBWR becomes a bottleneck. In this case, the optimal value on Intel UNIX for DBWR_IO_SLAVES is 4.

DB_WRITER_PROCESSES replaces the Oracle7 parameter DB_WRITERS and specifies the initial number of database writer processes for an instance. If you use DBWR_IO_SLAVES, only one database writer process is used, regardless of the setting for DB_WRITER_PROCESSES.

See "Customizing the initsid.ora File" on page 1-21 for information on other initialization parameters.

Look for Large Disk Request Queues Using iostat

A request queue shows how long the I/O requests on a particular disk device must wait to be serviced. Request queues are caused by a high volume of I/Os to that disk or by I/Os with long average seek times. Ideally, disk request queues should be at or near zero.

Choose the Appropriate File System Type

Intel UNIX allows a choice of file systems. File systems have different characteristics, and the techniques they use to access data can have a substantial affect on database performance. Typical file system choices are:

- `s5`: the UNIX System V File System
- `ufs`: the UNIX File System, derived from BSD UNIX
- `vxfs`: the Veritas File System
- raw device: no file system

The suitability of a file system to an application is usually undocumented. Even different `ufs` file systems are hard to compare because implementations differ. Although `ufs` is often the high-performance choice, performance differences vary from 0 to 20 percent, depending on the file system chosen.

Monitoring Disk Performance

To monitor disk performance, use the `sar -b` and `sar -u` commands.

Table 2–2 describes output of the `sar -b` commands.

Table 2–2 sar -b Output Fields

Field	Description
`bread/s, bwrit/s`	Blocks read and blocks written (important for file system databases)
`pread/s, pwrit/s`	Partition reads and partition writes (important for raw partition database systems)

An important `sar -u` column for disk performance is `%wio`, the percentage of CPU time waiting on blocked I/O.

Key indicators are:

- The sum of `bread`, `bwrit`, `pread`, and `pwrit` indicates the state of the disk I/O subsystem. The higher the sum, the greater the potential for disk I/O bottlenecks. The larger the number of physical drives, the higher the sum threshold number can be. A good default value is no more than 40 for two drives and no more than 60 for four to eight drives.

- The `%rcache` should be greater than 90 and `%wcache` should be greater than 60. Otherwise, the system may be disk I/O bound.
- If `%wio` is consistently greater than 20, the system is I/O bound.

Disk Performance Issues

Oracle block sizes should either match disk block sizes or be a multiple of disk block size.

If possible, do a file system check on the partition before using it for database files, then make a new file system to ensure that it is clean and unfragmented. Distribute disk I/O as evenly as possible, and separate log files from database files.

Tuning CPU Usage

The following sections describe how to tune CPU usage.

Keep All Oracle Users and Processes at the Same Priority

Oracle is designed to operate with all users and background processes operating at the same priority level. Changing priorities cause unexpected effects on contention and response times.

For example, if the log writer process (LGWR) gets a low priority, it is not executed frequently enough and LGWR becomes a bottleneck. On the other hand, if LGWR has a high priority, user processes can suffer poor response time.

Use Processor Affinity and Binding on Multi-Processor Systems

In a multi-processor environment, use processor affinity and binding if it is available on your system. Processor binding prevents a process from migrating from one CPU to another, allowing the information in the CPU cache to be better utilized. You can bind a server shadow process to make use of the cache as it is always active, and let background processes flow between CPUs.

Use Single-Task Linking for Large Exports/Imports and SQL*Loader Jobs

If you must transfer large amounts of data between the user and Oracle8*i* (for example, using the `export` and `import` utilities), it is efficient to a use single-task architecture. To make the single-task import (`impst`), export (`expst`), and SQL*Loader (`sqlldrst`) executables, use the `ins_rdbms.mk` makefile in the `$ORACLE_HOME/rdbms/lib` directory. The following example makes the `impst`, `expst`, and `sqlldrst` executables:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk singletask
```

Note: Linking Oracle executables as a single-task allows a user process to directly access the entire SGA. In addition, running single-task requires more memory because the Oracle executable text is no longer shared between the front-end and background processes.

Tuning Oracle Resource Contention and UNIX Kernel Parameters

You can improve performance by keeping the UNIX kernel as small as possible. The UNIX kernel typically pre-allocates physical RAM, leaving less memory available for other processes such as Oracle.

Traditionally, kernel parameters such as NBUF, NFILE, and NOFILES were used to adjust kernel size. However, most UNIX implementations dynamically adjust those parameters at run time, even though they are present in the UNIX configuration file.

Look for memory-mapped video drivers, networking drivers, and disk drivers that could be de-installed, freeing more memory for use by other processes.

Note: Remember to make a backup copy of your UNIX kernel. See your hardware vendor documentation for more information.

Tuning and Specifying Block Size and File Size

This section describes how you can improve the performance of Oracle8*i* by optimizing the size of Oracle blocks for the files in your database.

Note: To change block size, you must create a new database. To determine the most efficient configuration, experiment with block size before transferring your data to the new database.

On Intel UNIX, the default Oracle block size is 2KB and the maximum block size is 16 KB. You can set the block size to any multiple of 2KB up to a maximum of 16 KB, inclusive.

The optimal block size is typically the default size, however, it depends on the applications. To create a database with a different Oracle block size, add the following line to the `init`*`sid`*`.ora` file before creating the database:

```
db_block_size=new_block_size
```

Note: The value that you choose for the DB_BLOCK_SIZE parameter determines the maximum size of certain types of Oracle files. See Table 1–5, "Specific Oracle8i File Size Limits" on page 1-16 for more information on file size limits.

Tuning the Intel UNIX Buffer Cache Size

To take full advantage of raw devices, adjust the size of the Oracle8*i* buffer cache and, if memory is limited, the Intel UNIX buffer cache:

The Intel UNIX buffer cache holds blocks of data in memory while they are being transferred from memory to disk, or vice versa.

The Oracle8*i* buffer cache is the area in memory that stores the Oracle database buffers. Since Oracle8*i* can use raw devices, it does not need to use the Intel UNIX buffer cache.

If you use raw devices, you must increase the size of the Oracle8*i* buffer cache. If the amount of memory on the system is limited, make a corresponding decrease in the Intel UNIX buffer cache size. It is possible to increase or decrease the Oracle8*i* Buffer Cache by modifying the DB_BLOCK_BUFFERS parameter in the `init`*`sid`*`.ora` file and restarting the instance.

Use the `sar` command to determine which buffer caches you must increase or decrease. Table 2–3 shows the options of the `sar` command.

Table 2–3 sar Command options

Option	Description
`-b`	Reports the Intel UNIX buffer cache activity
`-w`	Reports the Intel UNIX swapping activity
`-u`	Reports CPU utilization
`-r`	Reports memory utilization
`-p`	Reports the Intel UNIX paging activity

Adjusting Cache Size

To adjust the cache size, perform one of the following:

- Increase the Oracle8*i* buffer cache size while the cache hit ratio continues to increase.
- Decrease cache sizes if the swapping / paging activity becomes high.

Using Trace and Alert Files

This section describes the trace (or dump) and alert files Oracle8*i* creates to diagnose and resolve operating problems.

Trace File Names

The file name format of a trace file is *`processname_sid_unixpid`*`.trc`, where:

`processname`	Is a three- or four-character process name showing which Oracle8*i* process the trace file is from (for example, `pmon`, `dbwr`, `ora`, or `reco`)
`sid`	Is the instance system identifier
`unixpid`	Is the UNIX process ID number
`.trc`	Is a filename extension appended to all trace file names

A sample trace file name is `lgwr_TEST_1237.trc`.

Alert Files

The `alert_`*`sid`*`.log` file is associated with a database and is located in the directory specified by the `init`*`sid`*`.ora` parameter BACKGROUND_DUMP_DEST. The default directory is `$ORACLE_HOME/rdbms/log`.

Using Raw Devices/Volumes

This section describes the use of raw devices on Oracle8*i*.

Disadvantages of Raw Devices/Volumes

Raw devices/volumes have the following disadvantages when used on Intel UNIX:

- Small client systems usually cannot use sufficiently large raw device and volume partitions. Disk partitions usually come in odd sizes that do not lend themselves to good database architecture.
- If a particular disk drive has intense I/O activity and performance would benefit from movement of an Oracle data file to another drive, it is likely that no acceptably sized section exists on a drive with less I/O activity. Moving data files around, a common advantage of UNIX, may not be possible with raw devices/volumes.
- Adding space to a tablespace can be a difficult process in a raw device/volume environment. Occasionally, all raw partitions are assigned data files at initial configuration time, leaving no raw storage to accommodate normal tablespace growth.

Guidelines for Using Raw Devices/Volumes

In addition to the disadvantages described in the previous section, you should consider the following issues when deciding whether to use raw devices/volumes:

- Oracle8i Parallel Server Installation (DG/UX Intel Only)
- Raw Disk Partition Availability
- Logical Volume Manager
- Dynamic Performance Tuning
- Mirroring and Online Disk Replacement

Oracle8*i* Parallel Server Installation (DG/UX Intel Only)

Each instance of OPS has individual log files. Therefore, in addition to the partitions required for the tablespaces and control files, each instance requires a minimum of three partitions for the log files. All the files must be on disks that can be shared by all nodes of a Intel UNIX cluster.

UNIX clusters do not provide access to a shared file system between all nodes of a cluster. As a result, all files associated with a database must be built on raw devices/volumes.

Raw Disk Partition Availability

Use raw devices/volumes for Oracle files only if your site has at least as many raw disk partitions as Oracle datafiles. If the raw disk partitions are already formatted, match datafile size to partition size as closely as possible to avoid wasting space.

Logical Volume Manager

With logical volumes, you can create logical disks based on raw partition availability. Because logical disks can be moved to more than one disk, the disk drives do not have to be reformatted to obtain logical disk sizes.

Dynamic Performance Tuning

You can optimize disk performance when the database is online by moving hot spots to cooler drives. Most hardware vendors who provide the logical disk facility also provide a graphical user interface that can be used for tuning.

Mirroring and Online Disk Replacement

You can mirror logical volumes to protect against loss of data. If one copy of a mirror fails, dynamic re-synchronization is possible. Some vendors also provide the ability to replace drives online in conjunction with the mirroring facility.

For OPS, you can use logical volumes for drives associated with a single UNIX system, as well as those that can be shared with more than one computer of a UNIX cluster. The latter allows for all files associated with the OPS to be placed on these shared logical volumes.

Raw Device Setup

Consider the following items when creating raw devices:

- Ensure that the owner and group are `oracle` and `oinstall`, respectively.
- The size of an Oracle datafile created in a raw partition must be at least two Oracle block sizes smaller than the size of the raw partition.

Caution: Do not attempt to set up raw devices without the help of an experienced system administrator and specific knowledge about the system you are using.

To set up raw devices on your system:

1. (This step is for OPS only): Make sure the partitions you are adding are on a shared disk.
2. Determine the names of the free disk partitions.

   A free partition is one that is not used for an Intel UNIX file system. That means that the partition follows these restrictions:

   - It is not listed when you execute the `/etc/mount` command.
   - It is not in use as a swap device.
   - It does not overlap a swap partition.
   - It is not in use by other Intel UNIX applications (for example, other instances of the Oracle server).
   - It does not overlap the Intel UNIX file system.
   - It does not use a space already used by the file system.

   To find out whether a partition is free, obtain a complete map of the starting locations and sizes of the partitions on the device and check for free space. Note that some partitions may contain file systems that are currently not mounted and are not listed in the `/etc/mount` output.

   Note: Make sure that the partition does *not* start at Cylinder 0.

3. Begin setting up the raw device for use by the Oracle8*i* Server by verifying that the disk is partitioned. If not, on Solaris Intel and SCO UnixWare systems use the operating system `format` utility to partition it. On DG/UX Intel systems, use the `sysadm` utility.

4. Make sure that the partition is owned by the `oracle` software owner. If necessary, to use the `chown` command to change its ownership on the block and character files for the device, enter one of the following:

   - On DG/UX Intel, enter:

     ```
     $ chown oracle /dev/rdsk/file1
     ```

   - On SCO UnixWare, enter:

     ```
     $ chown oracle /dev/rdsk/c0b0t0d0s1
     ```

   - On Solaris Intel, enter:

     ```
     $ chown oracle /dev/dsk/c120t0d0s0
     $ chown oracle /dev/rdsk/c120t0d0s0
     ```

   To use the `chmod` command to make the partition accessible only by the `oracle` software owner:

   - On DG/UX Intel, enter:

     ```
     $ chmod 600 /dev/rdsk/file1
     ```

   - On SCO UnixWare, enter:

     ```
     $ chmod 600 /dev/rdsk/c0b0t0d0s1
     ```

   - On Solaris Intel, enter:

     ```
     $ chmod 600 /dev/dsk/c120t0d0s0
     $ chmod 600 /dev/rdsk/c120t0d0s0
     ```

5. Create a symbolic link to the raw devices you require by entering one of the following:

   - On DG/UX Intel, enter:

     ```
     $ ln -s /dev/rdsk/file1 /oracle_data/datafile.dbf
     ```

   - On SCO UnixWare, enter:

     ```
     $ ln -s /dev/rdsk/c0b0t0d0s1 /oracle_data/datafile.dbf
     ```

   - On Solaris Intel, enter:

     ```
     $ ln -s /dev/rdsk/c120t0d0s0 /oracle_data/datafile.dbf
     ```

6. Make sure that you use the character special device, not the block special device. Enter the following command:

   ```
   $ ls -Ll datafile
   ```

   The command should return:

   ```
   crw-------  oracle dba datafile
   ```

   In the preceding example, the flags used in the above command are `L` = show symbolic links, and `l` = long listing.

   > **Note:** This symbolic link must be set up on each node of the OPS (for DG/UX Intel only). Check that no two symbolic links specify the same raw device.

7. Create or add the new partition to a new database.

   From Server Manager, use the SQL statement CREATE DATABASE to create the database using the specified raw partition.

   > **Note:** The size of an Oracle datafile created in a raw partition must be at least two Oracle block sizes smaller than the size of the raw partition.

Example 2–1

```
$ SVRMGRL
SVRMGR> CREATE DATABASE SID
SVRMGR> LOGFILE '/ORACLE_DATA/LOG1.DBF' SIZE 100K/
ORACLE_DATA/LOG2.DBF' SIZE 100K
SVRMGR> DATAFILE '/ORACLE_DATA/DATAFILE.DBF' SIZE 1000K REUSE;
```

8. If you want to add the partition to a tablespace in an existing Oracle database instead, enter:

```
$ SVRMGRL
SVRMGR> ALTER TABLESPACE TABLESPACE_NAME ADD DATAFILE
//DEV/RDSK/C0TLD0S6' SIZE 1000K REUSE;
```

Use the same procedure to set up a raw device for the redo files.

3

Administering SQL*Plus

This chapter describes how to use and administer SQL*Plus on Oracle8*i*. It contains the following sections:

- Administering SQL*Plus
- Using SQL*Plus
- Restrictions of SQL*Plus

Administering SQL*Plus

This section describes how to administer SQL*Plus.

Using Setup Files

When you start SQL*Plus, it executes the `glogin.sql` site profile setup file and then executes the `login.sql` user profile setup file.

Using the Site Profile File

The global site profile file is the `$ORACLE_HOME/sqlplus/admin/glogin.sql` directory. The default site profile is placed in the `$ORACLE_HOME/sqlplus/admin` directory when SQL*Plus is installed. If a site profile already exists, it is overwritten. If SQL*Plus is de-installed, the site profile file is deleted.

Using the User Profile File

The user profile file is `login.sql`. SQL*Plus looks in the current directory, and then in the directories you specify until it finds the `login.sql` file. You can specify the directories to search by setting the SQLPATH environment variable to a colon-separated list of directories.

For example, if the current directory is `/u02/oracle` and SQLPATH is set to `/home:/home/oracle:/u01/oracle`, SQL*Plus looks for the `login.sql` file in the following order:

1. `/u02/oracle` (the current directory)
2. `/home`
3. `/home/oracle`
4. `/u01/oracle`

The options set in the `login.sql` file override those set in the `glogin.sql` file.

> **See Also:** For more information on profile files, see the *SQL*Plus User's Guide and Reference*.

Using the PRODUCT_USER_PROFILE Table

During a Typical installation, the PRODUCT_USER_PROFILE table is created automatically. This table is used to disable the SQL and SQL*Plus commands you specify. To recreate this table, run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` script in the SYSTEM schema.

For example, enter:

```
$ sqlplus SYSTEM/MANAGER
SQL> @?/SQLPLUS/ADMIN/PUPBLD.SQL
```

SQL*Plus uses the value of the ORACLE_HOME environment variable wherever a question mark (?) appears.

Using Demonstration Tables

SQL*Plus is shipped with demonstration tables that you can use for testing.

Performing a Typical Installation

During a Typical installation, the user SCOTT and the demonstration tables are created automatically.

Creating Demonstration Tables Manually

Use the `$ORACLE_HOME/sqlplus/demo/demobld.sql` SQL script to create the demonstration tables. In SQL*Plus, you can use any user name to run the `demobld.sql` file to create the demonstration tables in a schema. For example, enter:

```
$ sqlplus SCOTT/TIGER
SQL> @?/SQLPLUS/DEMO/DEMOBLD.SQL
```

You can also use the `$ORACLE_HOME/bin/demobld` shell script to run the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script by entering:

```
$ demobld scott tiger
```

Deleting Demonstration Tables

Use the `$ORACLE_HOME/sqlplus/demo/demodrop.sql` script to drop demonstration tables. In SQL*Plus, you can use any user name to drop the demonstration tables in the user's schema. For example, enter:

```
$ SQLPLUS SCOTT/TIGER
SQL> @?/SQLPLUS/DEMO/DEMODROP.SQL
```

You can also use the `$ORACLE_HOME/bin/demodrop` shell script to run the `$ORACLE_HOME/sqlplus/demo/demodrop.sql` script by entering:

```
$ demodrop scott tiger
```

Note: Both the `demobld.sql` and `demodrop.sql` scripts drop the EMP, DEPT, BONUS, SALGRADE, and DUMMY. tables. Prior to running either script, make sure that no table with one of these names exists in the desired schema, or the table data is lost.

Installing and Removing the Help Facility

This section describes how to install and remove the help facility.

Performing a Typical Installation

When you copy a starter database with pre-built datafiles as part of the Typical installation or as an option in Oracle Database Configuration Assistant, SQL*Plus automatically installs the Help Facility.

Using the Database Configuration Assistant

You can use the Oracle Database Configuration Assistant to create help tables when creating a database.

Installing the Help Facility Manually

You can use the `$ORACLE_HOME/bin/helpins` shell script to manually install the help facility. Before you run the script, set the SYSTEM_PASS environment variable to the SYSTEM schema name and password. For example, enter:

```
$ setenv SYSTEM_PASS SYSTEM/MANAGER
$ helpins
```

If the SYSTEM_PASS variable is not set, the `helpins` scripts prompt you for the SYSTEM password and load the help data into the `SYSTEM` schema. Run the `$ORACLE_HOME/sqlplus/help/helpbld.sql` script with the `helpus.sql` script to create the help facility tables. Enter the following:

```
$ORACLE_HOME/sqlplus/help/helpbld.sql
```

```
helpus.sql
```

For example, enter:

```
$ sqlplus SYSTEM/MANAGER
SQL> @?/SQLPLUS/ADMIN/HELP/HELPBLD.SQL HELPUS.SQL
```

Note: Both the `helpins` shell script and the `helpbld.sql` SQL*Plus script drop existing help facility tables before creating new tables.

You can also run `$ORACLE_HOME/sqlplus/help/helpdrop.sql` in SQL*Plus to manually drop the help facility tables in a schema. For example, enter:

```
$ sqlplus SYSTEM/MANAGER
SQL> @?/SQLPLUS/ADMIN/HELP/HELPDROP.SQL
```

See Also: For more information on the help facility, see the *SQL*Plus User's Guide and Reference*.

Using SQL*Plus

This section describes how to use SQL*Plus.

Using a System Editor from SQL*Plus

If you enter an ED or EDIT command at the SQL*Plus prompt, the system starts an operating system editor, such as `ed`, `emacs`, `ned`, or `vi`. Your PATH variable must include the directory of the editor.

When you start the editor, the current SQL buffer is placed in the editor. When you exit the editor, the changed SQL buffer is returned to SQL*Plus.

You can specify which editor starts by defining the SQL*Plus `_editor` variable. This variable can be set in `glogin.sql`, in `login.sql`, or entered during a SQL*Plus session. For example, to set the default editor to `vi`, enter:

```
SQL> DEFINE _EDITOR=VI
```

If you do not set the *`_editor`* variable, the value of either the EDITOR or VISUAL environment variable is used. If both are set, the EDITOR variable value is used. When *`_editor`*, EDITOR, and VISUAL are not specified, the default editor is `ed`.

If you start the editor, SQL*Plus uses the `afiedt.buf` temporary file to pass text to the editor. You can use the SET EDITFILE command to rename this file. For example, enter:

```
SQL> SET EDITFILE /TMP/MYFILE.SQL
```

SQL*Plus does not delete the temporary file.

Running Operating System Commands from SQL*Plus

Using the HOST command or an exclamation point (!) as the first character after the SQL*Plus prompt causes subsequent characters to be passed to a sub-shell. The SHELL environment variable sets the shell used to execute operating system commands. The default shell is `/bin/sh`. If the shell cannot be executed, an error message is displayed.

To return to SQL*Plus, enter exit or press Ctrl+D.

For example, to enter one command, enter:

```
SQL>! COMMAND
```

In the preceding example, *`command`* represents the operating system command you want to execute.

To enter multiple OS commands from SQL*Plus, enter the HOST or ! command then press return. SQL*Plus returns you to the operating system prompt.

Interrupting SQL*Plus

While running SQL*Plus, you can stop the scrolling record display and terminate a SQL statement by pressing [Ctrl]+[c].

Using the SPOOL Command

The default extension name of files generated by the SPOOL command is `.lst`. To change this extension, specify a spool file containing a period (.). For example, enter:

```
SQL> SPOOL QUERY.LIS
```

Restrictions of SQL*Plus

This section describes SQL*Plus restrictions.

Resizing Windows

The default values for SQL*Plus LINESIZE and PAGESIZE do not automatically adjust for window size.

Return Codes

UNIX return codes use only one byte, which is not enough space to return an Oracle error code. The range for a return code is 0 to 255.

4

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes Oracle Precompilers and the Oracle Call Interface. It contains the following sections:

- Overview of Oracle Precompilers
- Pro*C/C++ Precompiler
- Pro*COBOL Precompiler
- Pro*FORTRAN Precompiler (DG/UX Intel Only)
- Oracle Call Interface
- Custom Make Files
- Correcting Undefined Symbols
- Multi-threaded Applications
- XA Functionality

Overview of Oracle Precompilers

Oracle precompilers are application development tools used to combine SQL statements from an Oracle database with programs written in a high-level language. Oracle precompilers are compatible with ANSI SQL and are used to develop open, customized applications that run with Oracle8*i* or any other ANSI SQL database management system.

> **See Also:** For general information on Oracle precompilers and interface features, see the *Oracle8i Server Application Developer's Guide*.

Precompiler Configuration Files

System configuration files for the Oracle Precompilers are located in the `$ORACLE_HOME/precomp/admin/` directory. Table 4–1 lists the system configuration files.

Table 4–1 System Configuration Files

Product	Configuration File
Pro*C/C++ release 8.1.7	`pcscfg.cfg`
Pro*COBOL release 8.1.7	`pcbcfg.cfg`
Pro*COBOL release 1.8.52	`pcccob.cfg`
Object Type Translator release 8.1.7	`ottcfg.cfg`

Table 4–2 lists the location of precompiler `README` files. The `README` files describe changes since the last release.

Table 4–2 Location of README Files for Oracle Products

Product	README File
Pro*C/C++ release 8.1.7	`$ORACLE_HOME/precomp/doc/proc2/readme.doc`
Pro*COBOL release 8.1.7	`$ORACLE_HOME/precomp/doc/procob2/readme.doc`
Pro*COBOL release 1.8.52	`$ORACLE_HOME/precomp/doc/pro1x/readme.txt`

Issues Common to All Precompilers

The following issues are common to all precompilers.

Note: To run Oracle Precompiler demonstrations, Oracle8*i* must already be installed.

Uppercase to Lowercase Conversion

In languages other than C, the compiler converts an uppercase function or subprogram name to lowercase. This can cause a “No such user exists” error message. If you receive this error message, verify that the function or subprogram name in your option file matches the case used in the `iapxtb` table.

Vendor Debugger Programs

Precompilers and vendor-supplied debuggers can be incompatible. Oracle Corporation does not guarantee that a program with a debugger will run the same way on an operating system.

Value of ireclen and oreclen

The ireclen and oreclen parameters do not have maximum values.

Static and Dynamic Linking

You can statically or dynamically link precompiler and OCI applications with Oracle libraries. With static linking, the libraries and objects of the whole application are linked together into a single executable program. As a result, application executables can become very large.

With dynamic linking, the executing code is partly stored in the executable program and partly stored in libraries that are linked dynamically by the application at run time. Libraries that are linked at run time are called dynamic or shared libraries. Benefits of dynamic linking are:

- Smaller disk requirements: More than one application or invocation of the same application can use the same dynamic library.
- Smaller main memory requirements: The same dynamic library image is loaded into main memory only one time and an be shared by more than one application.

Client Shared Library

The client shared library, `libclntsh.so`, is located in the `$ORACLE_HOME/lib` directory. If you use the Oracle provided `demo_`*`product`*`.mk` make file to link an application, the client shared library is used by default.

You may receive the following error message when starting an executable:

```
% sample1
ld.so.1: sample1: fatal: libclntsh.so.1.0: can't open file: errno=2
Killed
```

If you receive this error message, set the LD_LIBRARY_PATH environment variable as follows:

```
$ LD_LIBRARY_PATH=$ORACLE_HOME/lib
$ export LD_LIBRARY_PATH
```

The client shared library is created automatically during installation. If you must re-create the client shared library:

1. Exit all client applications using the client shared library, including all Oracle client applications such as SQL*Plus and Recovery Manager.
2. Log in as the `oracle` user and enter:

   ```
   $ cd $ORACLE_HOME/rdbms/lib
   $ make -f ins_rdbms.mk libclntsh.so
   ```

Pro*C/C++ Precompiler

Before you use Pro*C/C++, verify that the correct version of the operating system compiler is properly installed.

Pro*C/C++ Demonstration Programs

Demonstration programs are provided to show the varied functionality of the Pro*C/C++ precompiler. There are three types of demonstration programs: C, C++, and Object programs. Object programs demonstrate the new Oracle8*i* Object features. All of the demonstration programs are located in the `$ORACLE_HOME/precomp/demo/proc` directory. The programs assume that the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script exist in the SCOTT schema with the password TIGER. By default, all programs are dynamically linked with the `$ORACLE_HOME/lib/libclntsh.so` client shared library.

See Also: For more information on using demonstration programs, see the *Pro*C/C++ Precompiler Programmer's Guide*.

Use the `demo_proc.mk` make file, located in the `$ORACLE_HOME/precomp/demo/proc/` directory, to create the demonstration programs. For example, to precompile, compile, and link the `sample1` demonstration program, enter the following command:

```
$ make -f demo_proc.mk sample1
```

To create the C demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk samples
```

To create the C++ demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk cppsamples
```

To create the Object demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk object_samples
```

Some demonstration programs require you to run a SQL script, located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message displays requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `calldemo` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/calldemo.sql` script, enter:

```
$ make -f demo_proc.mk calldemo RUNSQL=run
```

To create all Object demonstration programs and run all corresponding required SQL scripts, enter:

```
$ make -f demo_proc.mk object_samples RUNSQL=run
```

Note: The ORA_CLIENT_LIB environment variable is no longer used by the `demo_proc.mk` make file.

See Also: For information on using SQL*Plus to build demonstration programs, see "Using Demonstration Tables" on page 3-3.

Pro*C/C++ User Programs

You can use the `$ORACLE_HOME/precomp/demo/proc/demo_proc.mk` make file to create programs. The syntax for linking a program with the `demo_proc.mk` make file is:

```
$ make -f demo_proc.mk target OBJS="objfile1 objfile2 ..." EXE=exename
```

In the preceding example:

- *`target`* is the make file target that you want to use (example, `build`)
- *`objfilen`* is the object file to link the program
- *EXE* is the executable program

For example, to create the program `myprog` from the Pro*C/C++ source `myprog.pc`, enter one of the following commands, depending on the source and the type of executable you want to create.

For C source, dynamically linked with the client shared library, enter:

```
$ make -f demo_proc.mk build OBJS=myprog.o EXE=myprog
```

For C source, statically linked, enter:

```
$ make -f demo_proc.mk build_static OBJS=myprog.o EXE=myprog
```

For C++ source, dynamically linked with the client shared library, enter:

```
$ make -f demo_proc.mk cppbuild OBJS=myprog.o EXE=myprog
```

For C++ source, statically linked, enter:

```
$ make -f demo_proc.mk cppbuild_static OBJS=myprog.o EXE=myprog
```

Note: In the preceding examples, the file `myprog.o` is the object file generated by the compiler.

Pro*COBOL Precompiler

There are two versions of Pro*COBOL included with this release. Table 4–3 shows the naming conventions for these versions.

*Table 4–3 Pro*COBOL Naming Differences*

Feature	Pro*COBOL Release 8.1.7	Pro*COBOL Release 1.8.52
Executable	`procob`	`procob18`
Demonstration Directory	`procob2`	`procob`
make file for MicroFocus COBOL	`demo_procob.mk`	`demo_procob18.mk`

Pro*COBOL supports statically linked, dynamically linked, or dynamically loadable programs. Dynamically linked programs use the client shared library `$ORACLE_HOME/lib/libclntsh.so`. Dynamically loadable programs use the `rtsora` executable located in the `$ORACLE_HOME/bin` directory.

MicroFocus COBOL Compiler

The MicroFocus COBOL Compiler requires the COBDIR and LD_LIBRARY_PATH environment variables be set to run the compiler.

The COBDIR variable must be set to the directory where the compiler is installed. For example, enter:

```
$ set COBDIR /opt/cobol; export COBDIR
```

In the preceding example, `/opt/cobol` is the compiler directory.

The LD_LIBRARY_PATH environment variable must include the `$COBDIR/coblib` directory. For example, to append `$COBDIR/coblib` to LD_LIBRARY_PATH, enter:

```
$ set LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:$COBDIR/coblib
$ export LD_LIBRARY_PATH
```

If LD_LIBRARY_PATH does not contain the `$COBDIR/coblib` directory, the following error message appears when you compile a program:

```
ld.so.1: rts32: fatal: libfhutil.so.2.0: can't open file: errno=2
```

Pro*COBOL Demonstration Programs

This section describes the Pro*COBOL demonstration programs.

Oracle Run Time System

Oracle provides its own complete run time system, called `rtsora`, to run dynamically loadable Pro*COBOL programs. Use the `rtsora` run time system in place of the MicroFocus provided `cobrun` run time system to run dynamically loadable Pro*COBOL programs. If you attempt to run a Pro*COBOL program with `cobrun`, you receive the following error:

```
$ cobrun sample1.gnt
Load error : file 'SQLADR'
error code: 173, pc=0, call=1, seg=0
173     Called program file not found in drive/directory
```

Demonstration Programs

Demonstration programs are provided to show the varied functionality of the Pro*COBOL precompiler. All of the demonstration programs are located in either the `$ORACLE_HOME/precomp/demo/procob` directory or the `$ORACLE_HOME/precomp/demo/procob2` directory, depending on the Pro*COBOL version. The programs assume that the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script exist in the SCOTT schema with the password TIGER. By default, the demonstration programs are dynamically linked with the `$ORACLE_HOME/lib/libclntsh.so` client shared library.

Use the demonstration make file to create the demonstration programs. The demonstration make file for Pro*COBOL release 8.1.7 is `$ORACLE_HOME/precomp/demo/procob/demo_procob.mk`. The demonstration make file for Pro*COBOL release 1.8.52 is `$ORACLE_HOME/precomp/demo/procob/demo_procob18.mk`.

For example, to precompile, compile, and link the `sample1` demonstration program for Pro*COBOL release 8.1.7, enter:

```
$ cd $ORACLE_HOME/precomp/demo/procob2
$ make -f demo_procob.mk sample1
```

To create all Pro*COBOL demonstration programs, enter:

```
$ make -f demo_procob.mk samples
```

To create and run a dynamically loadable `sample1.gnt` program to be used with `rtsora`, enter:

```
$ make -f demo_procob.mk sample1.gnt
$ rtsora sample1.gnt
```

Some demonstration programs require you to run a SQL script, located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message displays requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `sample9` demonstration program and run the required `calldemo.sql` script located in the `$ORACLE_HOME/precomp/demo/sql` directory, enter:

```
$ make -f demo_procob.mk sample9 RUNSQL=run
```

> **See Also:** For information on using SQL*Plus to build demonstration programs, see "Using Demonstration Tables" on page 3-3.
>
> For more information on the demonstration programs see the *Programmer's Guide to the Pro*COBOL Precompiler.*

Pro*COBOL User Programs

You can use `demo_procob.mk` make file to create user programs. Be sure to use the appropriate make file depending on the Pro*COBOL version and COBOL compiler used. The syntax for linking a user program with the demonstration make file is:

```
$ make -f demo_procob.mk target COBS="cobfile1 cobfile2..." EXE=exename
```

In the preceding example:

- *`target`* is the make file target that you want to use (for example, `build`)
- *`cobfilen`* is the COBOL source for the program
- *`exename`* is the executable program

For example, to create the `myprog` program from the Pro*COBOL source `myprog.pco`, enter one of the following lines, depending on the type of executable and the use of shared library resources:

- For a dynamically linked executable with client shared library, enter:

  ```
  $ make -f demo_procob.mk build COBS=myprog.cob EXE=myprog
  ```

- For a statically linked executable without client shared library, enter:

```
$ make -f demo_procob.mk build_static COBS=myprog.cob EXE=myprog
```

Note: In the above examples, the file `myprog.cob` is the object file generated by the compiler.

- For a dynamically loadable module usable with `rtsora`, enter:

```
$ make -f demo_procob.mk myprog.gnt
```

FORMAT Precompiler Option

The `FORMAT` precompiler option specifies the format of input lines for COBOL. If you specify `FORMAT=ANSI` (the default), columns 1 to 6 contain an optional sequence number, column 7 indicates comments or continuation lines, paragraph names begin in columns 8 to 11, and statements begin in columns 12 to 72.

If you specify `FORMAT=TERMINAL`, columns 1 to 6 are dropped, making column 7 the leftmost column.

Pro*FORTRAN Precompiler (DG/UX Intel Only)

Before using Pro*FORTRAN, make sure that the correct version of the compiler is installed.

See Also: The required version for your operating system is specified in *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*.

Pro*FORTRAN Demonstration Programs

Demonstration programs are provided to show the varied functionality of the Pro*FORTRAN precompiler. All of the demonstration programs are located in the `$ORACLE_HOME/precomp/demo/profor` directory. The programs assume that the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script exist in the SCOTT schema with the password TIGER. By default, all programs are dynamically linked with the client shared library `$ORACLE_HOME/lib/libclntsh.so`.

Use the `demo_profor.mk` make file located in the `$ORACLE_HOME/precomp/demo/profor` directory to create the demonstration programs. For example, to precompile, compile, and link the `sample1` demonstration program, enter:

```
$ make -f demo_profor.mk sample1
```

or

```
$ make -f demo_profor.mk build FORS=sample1.pfo EXE=sample1
```

To create all Pro*FORTRAN demonstration programs, enter:

```
$ make -f demo_profor.mk samples
```

Some demonstration programs require you to run a SQL script, located the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message appears requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `sample11` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample11.sql` script, enter:

```
$ make -f demo_profor.mk sample11 RUNSQL=run
```

> **See Also:** For information on using SQL*Plus to build demonstration programs, see "Using Demonstration Tables" on page 3-3.
>
> For more information on Pro*FORTRAN precompilers, see *Pro*FORTRAN Supplement to Oracle Precompilers*.

Pro*FORTRAN User Programs

You can use the `$ORACLE_HOME/precomp/demo/profor/demo_profor.mk` make file to create user programs. The syntax for linking a user program with `demo_profor.mk` is:

```
$ make -f demo_profor.mk target FORS="forfile1 forfile2 ..." EXE=exename
```

In the preceding example:

- *`target`* is the make file target that you want to use (for example, `build`)
- *`objfilen`* is the object file to link the program
- *EXE* is the executable program

For example, to create the program `myprog`, from the Pro*FORTRAN source `myprog.pfo`, enter one of the following commands, depending on the type of executable that you want:

For a dynamically linked executable with the client shared library, enter:

```
$ make -f demo_profor.mk build FORS=myprog.f EXE=myprog
```

For a statically linked executable, enter:

```
$ make -f demo_profor.mk build_static FORS=myprog.f EXE=myprog
```

Oracle Call Interface

Before using the Oracle Call Interface (OCI), make sure that the correct version of Pro*C/C++ is installed.

> **See Also:** The required version for your operating system is specified in the *Oracle8i Installation Guide for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)*.

OCI Demonstration Programs

Demonstration programs are provided to show the varied functionality of the OCI. There are two types of demonstration programs: C and C++. All of the demonstration programs are located in the `$ORACLE_HOME/rdbms/demo` directory. By default, all programs are dynamically linked with the client shared library `$ORACLE_HOME/lib/libclntsh.so`.

Many of the demonstration programs assume that the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script exist in the SCOTT schema with the password TIGER.

Use the `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk` make file to create the demonstration programs. For example, to compile and link the `cdemo1` demonstration program, enter:

```
$ make -f demo_rdbms.mk cdemo1
```

To create the OCI C demonstration programs, enter:

```
$ make -f demo_rdbms.mk demos
```

To create all of the OCI C++ demonstration programs, enter:

```
$ make -f demo_rdbms.mk cppdemos
```

Some demonstration programs require you to run a SQL script found in the `$ORACLE_HOME/rbdms/demo` directory. If you do not run the script, a message appears requesting you to run the script. In most cases, the SQL script name is the same as the program name with a `.sql` extension.

> **See Also:** For information on using SQL*Plus to build demonstration programs, see "Using Demonstration Tables" on page 3-3.
>
> For more information on the demonstration programs see the *Programmer's Guide to the Oracle Call Interface* and the program source for details of each program.

OCI User Programs

You can use the `demo_rdbms.mk` make file, located in the `$ORACLE_HOME/rdbms/demo` directory, to create programs. The syntax for the linking a user program with `demo_rdbms.mk` is:

```
$ make -f demo_rdbms.mk target OBJS="objfile1 objfile2 ..." EXE=exename
```

In the preceding example:

- *`target`* is the make file target that is used (example, `build`)
- *`objfilen`* is the object file to link the program
- *EXE* is the executable program

For example, to create the `myprog` program from the C/C++ source `myprog.c`, enter one of the following, depending on the type of executable you want:

To create the `myprog` program from the C source, dynamically linked with the client shared library, enter:

```
$ make -f demo_rdbms.mk build OBJS=myprog.o EXE=myprog
```

To create the `myprog` program from the C source, statically linked, enter:

```
$ make -f demo_rdbms.mk build_static OBJS=myprog.o EXE=myprog
```

To create the `myprog` program from the C++ source, dynamically linked with client shared library, enter:

```
$ make -f demo_rdbms.mk buildc++ OBJS=myprog.o EXE=myprog
```

To create the `myprog` program from the C++ source, statically linked, enter:

```
$ make -f demo_rdbms.mk buildc++_static OBJS=myprog.o EXE=myprog
```

Note: In the above examples, the file `myprog.o` is the object file generated by the compiler.

Custom Make Files

Oracle Corporation recommends that you use the provided `demo_`*`product`*`.mk` make files to link user programs as described in the specific product sections of this chapter. If you modify the provided make file, or if you choose to use a custom-written make file, the following restrictions apply:

- Do not modify the order of the Oracle libraries. Oracle libraries are included on the link line more than once so that all the symbols are resolved during linking. The order of the Oracle libraries is essential for the following reasons:
    - Oracle libraries are mutually referential. Functions in library A call functions in library B, and functions in library B call functions in library A.
    - The Intel UNIX linker is a one-pass linker. The linker searches a library exactly once at the point it is encountered in the link line.
- If you add your own library to the link line, add it to the beginning or to the end of the link line. Do not place user libraries between the Oracle libraries.
- If you choose to use a make utility such as `nmake` or GNU `make`, you should be aware of how macro and suffix processing differs from the make utility provided with Intel UNIX. Oracle make files are tested and are supported with the Intel UNIX `make` utility.
- Oracle library names and the contents of Oracle libraries are subject to change between releases. Always use the `demo_`*`product`*`.mk` make file that ships with the current release as a guide to determine the required libraries.

Correcting Undefined Symbols

Oracle provides the `symfind` utility to assist in locating a library or object file where a symbol is defined. A common error when linking a program is undefined symbols, which produces an error message similar to the following:

```
$ make -f demo_proc.mk sample1
Undefined                       first referenced
 symbol                             in file
sqlcex                              sample1.o
sqlglm                              sample1.o
ld: fatal: Symbol referencing errors. No output written to sample1
```

The error occurs when the linker cannot find a definition for a referenced symbol. If this error message occurs, ensure that the library or object file containing the definition exists on the link line and that the linker is searching the correct directories for the file.

The following example shows the output from the `symfind` utility, used to locate the `sqlcex` symbol:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
------------------------------------------------------
Command:         /u01/app/oracle/product/8.1.7/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/8.1.7
Output File:     (none)
Note:            I do not traverse symbolic links
                 Use '-v' option to show any symbolic links

Locating Archive and Object files ...
[11645] |    467572|      44|FUNC |GLOB |0    |8      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libclntsh.so
[35]    |         0|      44|FUNC |GLOB |0    |5      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libsql.a
```

Multi-threaded Applications

The Oracle libraries provided with this release are thread safe, allowing support for multi-threaded applications.

Using Signal Handlers

Signals can be used by Oracle8*i* for two-task communication. Signals are installed in a user process when you connect to the database and are de-installed when you disconnect.

Table 4–4 describes the signals that Oracle8*i* uses for two-task communications.

Table 4–4 Signals for Two-Task Communications

Signal	Description
SIGCLD	The pipe driver uses SIGCLD, also referred to as SIGCHLD, when an Oracle process dies. The UNIX kernel sends a SIGCLD signal to the user process. The signal handler uses the wait() routine to see if a server process died. The Oracle process does not catch SIGCLD; the user process catches it.
SIGCONT	The pipe two-task driver uses SIGCONT to send out-of-band breaks from the user process to the Oracle process.
SIGINT	Two-task drivers use SIGINT to detect user interrupt requests. The Oracle process does not catch SIGINT; the user process catches it.
SIGIO	Net8 protocols use SIGIO to indicate incoming networking events.
SIGPIPE	The pipe driver uses SIGPIPE to detect end-of-file on the communications channel. When writing to the pipe, if no reading process exists, a SIGPIPE signal is sent to the writing process. Both the Oracle process and the user process catch SIGPIPE. SIGCLD is similar to SIGPIPE, but only applies to user processes, not to Oracle processes.
SIGTERM	The pipe driver uses SIGTERM to signal interrupts from the user to the Oracle process. This occurs when the user presses the interrupt key [Ctrl]+[c]. The user process does not catch SIGTERM; the Oracle process catches it.
SIGURG	Net8 TCP/IP drivers use SIGURG to send out-of-band breaks from the user process to the Oracle process.

The listed signals affect Pro*C or other precompiler applications. You can install one signal handler for `SIGCLD` (or `SIGCHLD`) and `SIGPIPE` when connected to the Oracle process. If you call the `osnsui()` routine to set it up, you can have more than one signal handle for `SIGINT`. For `SIGINT`, use `osnsui()` and `osncui()` to register and delete signal-catching routines. You can also install as many signal handlers as you want for other signals. If you are not connected to the Oracle process, you can have multiple signal handlers.

The following example shows how you can set up a signal routine and a catching routine:

```
/* user side interrupt set */
word osnsui( /*_ word *handlp, void (*astp), char * ctx, _*/)
/*
** osnsui: Operating System dependent Network Set User-side Interrupt. Add an
** interrupt handling procedure astp. Whenever a user interrupt(such as a ^C)
** occurs, call astp with argument ctx. Put in *handlp handle for this
** handler so that it may be cleared with osncui. Note that there may be many
** handlers; each should be cleared using osncui. An error code is returned if
** an error occurs.
*/

/* user side interrupt clear */
word osncui( /*_ word handle _*/ );
/*
** osncui: Operating System dependent Clear User-side Interrupt. Clear the
** specified handler. The argument is the handle obtained from osnsui. An error
** code is returned if an error occurs.
*/
```

The following example is a template for using the `osnsui()` and the `osncui()` routines in an application program:

```
/*
** User interrupt handler template.
*/
void sig_handler()
{
...
}

main(argc, argv)
int arc;
char **argv;
    {

    int handle, err;
    ...

    /* set up my user interrupt handler */
    if (err = osnsui(&handle, sig_handler, (char *) 0))
{
/* if the return value is non-zero, an error has occurred
Take appropriate action for the error. */
...
}
...
/* clear my interrupt handler */
if (err = osncui(handle))
{
/* if the return value is non-zero, an error has occurred
Take appropriate action for the error. */
...
}
...
}
```

XA Functionality

Oracle XA is the Oracle implementation of the X/Open Distributed Transaction Processing (DTP) XA interface. The XA standard specifies a bi-directional interface between resource managers (for example, Oracle) that provide access to shared resources within transactions, and between a transaction service that monitors and resolves transactions.

Oracle Call Interface has XA functionality. When building a TP-monitor XA application, ensure that the TP-monitor libraries (that define the symbols ax_reg and ax_unreg) are placed in the link line before the Oracle client shared library. This link restriction is required only when using the XA dynamic registration (Oracle XA switch `xaoswd`).

Oracle8*i* does not support Oracle7 release 7.1.6 XA calls (although it does support release Oracle7 release 7.3 XA calls). Therefore , TP-monitor XA applications using Oracle7 release 7.1.6 XA calls must be relinked with the Oracle8*i* XA library. The Oracle8*i* XA calls are defined in both the `$ORACLE_HOME/lib/libclntsh.so` shared library and the `$ORACLE_HOME/lib/libclient8.a` static library.

5

Configuring Net8

This chapter describes Net8 features on Oracle8i Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel). It includes the following sections:

- Core Net8 Products and Features
- Net8 Protocol Support
- BEQ Protocol
- IPC Protocol
- RAW Protocol
- TCP/IP Protocol Adapter
- Oracle Enterprise Manager (Solaris Intel Only)
- Configuring Oracle Intelligent Agent for Oracle SNMP
- Oracle Advanced Security

Core Net8 Products and Features

This section describes core Net8 products and features.

Net8 Files and Utilities

This section describes the files and utilities that you can use to configure Oracle Net8 products.

Location of Net8 Configuration Files

The default directory for Net8 configuration files on Intel UNIX systems is `/var/opt/oracle`.

Oracle Net8 searches the following locations for global files in the following order:

1. The directory specified by the TNS_ADMIN environment variable, if set
2. The `/var/opt/oracle` directory
3. The `$ORACLE_HOME/network/admin` directory

To set the TNS_ADMIN environment variable in the startup files of all network users, enter:

For the Bourne or Korn shell:

```
$ TNS_ADMIN=directory_path
$ export TNS_ADMIN
```

For the C shell, enter:

```
% setenv TNS_ADMIN directory_name
```

For each system level configuration file, users may have a corresponding local private configuration file (stored in the user's home directory). The settings in the local file override the settings in the system level file. The local configuration file for `sqlnet.ora` is `$HOME/.sqlnet.ora`. The local configuration file for `tnsnames.ora` is `$HOME/.tnsnames.ora.` Syntax for these files is identical to that of the corresponding system files.

Sample Configuration Files

Examples of the `cman.ora`, `listner.ora`, `names.ora`, `sqlnet.ora`, and `tnsnames.ora` configuration files are located in the `$ORACLE_HOME/network/admin/samples` directory.

The Adapters Utility

Net8 provides support for various network protocol adapters and naming methods. These adapters and naming methods are linked to particular executables and provide the interface between network protocols and Net8. To display installed Net8 protocol adapters, enter:

```
$ adapters
```

To display adapters linked with a specific executable, enter:

```
$ adapters executable
```

For example, to display the Net8 protocol adapters linked with the Oracle executable, enter:

```
$ adapters oracle
```

The `adapters` executable displays output similar to the following:

```
Net8 Protocol Adapters linked with oracle are:
BEQ Protocol Adapter
IPC Protocol Adapter
TCP/IP Protocol Adapter
RAW Protocol Adapter
Net8 Naming Adapters linked with oracle are:
Oracle TNS Naming Adapter
Oracle Naming Adapter
Oracle Advanced Security/Networking Security products linked with oracle are:
```

Oracle Connection Manager

Oracle Connection Manager (OCM) is a router through which a client connection request may be sent either to its next hop or directly to the database server. Clients who route their connection requests through the OCM can then take advantage of the connection concentration, Net8 access control, or multi-protocol support features configured on that OCM.

OCM listens for incoming requests from clients and initiates connection requests to destination services. OCM performs these tasks with the help of Oracle Connection Gateway Process and Oracle Connection Manager Administrative Process.

OCM also includes a feature which you can use to control client access to designated servers in a TCP/IP environment. By specifying certain filtering rules you may allow or restrict access from specific clients to a server based on the following criteria:

- Source host names or IP addresses for clients
- Destination host names or IP addresses for servers
- Destination database service name

Net8 Firewall Proxy

Some firewall vendors also offer Net8 Firewall Proxy, which is installed on firewalls requiring an application proxy. Net8 Firewall Proxy has the same access controls as Oracle Connection Manager.

> **See Also:** For more information on the Oracle Connection Manager, see the *Net8 Administrator's Guide*.

Server Models

Net8 connects clients and servers using two different models:

- Multi-threaded Server Model—The listener passes the connection to the dispatcher or redirects the clients to connect through a dispatcher.

  In a Multi-threaded Server (MTS) configuration, client user processes connect to a dispatcher. A dispatcher can support multiple client connections concurrently. Each client connection is bound to a virtual circuit. A virtual circuit is a piece of shared memory used by the dispatcher for client database connection requests and replies. This approach enables a small pool of server processes to serve a large number of clients. A significant advantage of the MTS model over the dedicated server model is the reduction in the use of system resources, enabling the support of an increased number of users.

- Dedicated Server Model—The listener starts a dedicated server and passes the connection to the dedicated server or redirects the clients to the dedicated server.

  In the dedicated server model, there is one server process for each client. For clients to connect to dedicated servers, the listener and the instance must be running on the same system. Dedicated server processes require more memory than MTS.

See Also: For more information on the Multi-threaded Server, see the *Net8 Administrator's Guide*.

Oracle Names

Oracle Names uses Oracle proprietary software to store the names and addresses of all database services on a network. Clients that want to connect to a server direct their connection requests to an Oracle Names server. Oracle Names servers resolve the name to a network address and return that information to the client.

See Also: For information on Oracle Names, see the *Net8 Administrator's Guide*.

Net8 Protocol Support

Net8 release 8.1.7 on Intel UNIX supports the following protocols:

- BEQ
- IPC
- RAW
- TCP/IP

Before installing the TCP/IP protocol, you must install the appropriate operating system software. The BEQ, IPC, and RAW Net8 protocols do not have any specific operating system requirements.

See Also: Refer to the *Oracle8i Installation Guide Release 3 (8.1.7) for Intel UNIX (DG/UX Intel, SCO UnixWare, Solaris Intel)* for more information on Net8.

ADDRESS Specification

The IPC and TCP/IP Net8 protocols each have a protocol-specific ADDRESS specification that is used for Net8 configuration files and for the MTS_LISTENER_ADDRESS database initialization parameter in the `init`*`sid`*`.ora` file. See the ADDRESS specification heading under each protocol section in this chapter for details.

Table 5–1 shows a summary of ADDRESS specifications for each protocol.

Table 5–1 ADDRESS Specification Summary

Supported Protocol | **ADDRESS Specification**

BEQ

```
(ADDRESS =
    (PROTOCOL = BEQ)
    (PROGRAM = ORACLE_HOME/bin/oracle)
    (ARGV0 = oracleORACLE_SID)
    (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    (ENVS = 'ORACLE_HOME=ORACLE_HOME,ORACLE_SID=ORACLE_SID')
)
```

IPC

```
(ADDRESS =
    (PROTOCOL=IPC)
    (KEY=key)
)
```

RAW

N / A

TCP / IP

```
(ADDRESS =
    (PROTOCOL=TCP)
    (HOST=hostname)
    (PORT=port_id)
)
```

BEQ Protocol

The BEQ protocol adapter is both a communications mechanism and a process-spawning mechanism. To use the BEQ protocol adapter, the client and server must be on the same system. If a network service name is not specified, either directly by the user on the command line or Login screen or indirectly through an environment variable such as TWO_TASK, then the BEQ protocol is used. In this case, the BEQ protocol always uses a dedicated server and the multi-threaded server is never used. This dedicated server is started automatically by the BEQ protocol, which waits for the server process to start and attach to an existing System Global Area (SGA). If the startup of the server process is successful, the BEQ protocol then provides inter-process communication through UNIX pipes.

An important feature of the BEQ protocol is that no listener is required for its operation. The protocol is linked into the client tools and directly starts its own server process without outside interaction. However, the BEQ protocol can only be used when the client program and Oracle8*i* are installed on the same system. The

BEQ protocol is always installed and always linked to all client tools and to the Oracle8*i* server.

The BEQ protocol connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the following parameters in any order:

```
(ADDRESS =
    (PROTOCOL = BEQ)
    (PROGRAM = ORACLE_HOME/bin/oracle)
    (ARGV0 = oracleORACLE_SID)
    (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    (ENVS = 'ORACLE_HOME=ORACLE_HOME,ORACLE_SID=ORACLE_SID')
)
```

Table 5–2 describes the syntax for the BEQ protocol connection parameters.

Table 5–2 Syntax for BEQ Protocol Connection Parameters

Parameter	Description
PROTOCOL	The protocol adapter to be used. The value is BEQ. It is not case sensitive.
PROGRAM	The full path to the Oracle executable.
ARGV0	The name of the process as it appears in a `ps` listing. The recommended value is oracle*ORACLE_SID*.
ARGS	`'(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))'`
ENVS	Environment specification that includes the ORACLE_HOME and ORACLE_SID variables, where ORACLE_HOME represents the full path to the Oracle home directory of the database, and ORACLE_SID represents the system identifier of the database.

Example 5–1 shows a sample BEQ ADDRESS:

Example 5–1 BEQ ADDRESS Specifying a Client

```
(ADDRESS =
    (PROTOCOL = BEQ)
    (PROGRAM = /u01/app/oracle/product/8.1.7/bin/oracle)
    (ARGV0 = oracleV817)
    (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    (ENVS = 'ORACLE_HOME=/u01/app/oracle/product/8.1.7,ORACLE_SID=V817')
```

IPC Protocol

The IPC protocol is similar to the BEQ protocol in that it can only be used when the client program and the Oracle8*i* server are installed on the same system. The IPC protocol differs from the BEQ protocol in that it can be used both with a dedicated server and Multi-threaded Server configurations. The IPC protocol requires a listener for its operation. The IPC protocol is always installed and always linked to all client tools and to Oracle8*i*.

For the IPC protocol, the location of the UNIX Domain Socket (IPC) file on UNIX systems changed after Oracle7 release 7.1. If you have Oracle7 release 7.1 installed on the same computer as Oracle8*i* and you attempt to make an IPC connection between the two instances, the connection might fail. The solution to this problem is to make a symbolic link between the previous location of the IPC file directory (`/var/tmp/o`) to its current directory location (`/var/tmp/.oracle`).

The IPC protocol connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the following parameters in any order:

```
(ADDRESS=
    (PROTOCOL=IPC)
    (KEY=key)
)
```

Table 5–3 describes the syntax for IPC protocol connections parameters.

Table 5–3 Syntax for IPC Protocol Connection Parameters

Parameter	**Description**
PROTOCOL	IPC protocol adapter to be used. The value is IPC. It is not case sensitive.
KEY	Service name of database or database identifier (SID).

Example 5–2 shows a sample IPC ADDRESS.

Example 5–2 IPC Address Specifying a Client

```
(ADDRESS=
    (PROTOCOL=IPC)
    (KEY=PROD)
```

The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

RAW Protocol

When data is transferred between a client and a server, Net8 adds its own header information to every network packet. Through the Raw Transport feature, Net8 can now minimize header information on each packet going over the network.

After a connection is established, two types of information flow over the network: data and break handling. The connection packets require the Net8 header information to establish the connection correctly. However, after the connection is established, all data packets are stripped of their Net8 header information and passed directly to the operating system, bypassing the Net8 network and protocol layers. The performance of the connection is increased because of fewer protocol stack layers for the data to flow through and fewer bytes that are transmitted over the network.

This feature is transparently enabled when it is required. If no existing features require that header information be transmitted, the headers are stripped off. For example, Raw Transport would not be enabled when you use authentication, which requires certain information to be sent with each packet of information.

This feature requires no configuration. Net8 determines whether the conditions are met and then transparently switches to Raw Transport mode.

TCP/IP Protocol Adapter

Oracle Corporation recommends that you reserve a port for your listener in the `/etc/services` file of each Net8 node on the network. The default port is 1521. The entry lists the listener name and the port number, for example:

```
listener      1521/tcp
```

In this example *`listener`* is the name of the listener, as defined in the `listener.ora` file. You should reserve more than one port to start more than one listener.

Specifying a TCP/IP ADDRESS

The TCP/IP protocol connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the parameters in any order:

```
(ADDRESS=
    (PROTOCOL=TCP)
    (HOST=hostname)
    (PORT=port_id)
)
```

Table 5–4 describes the syntax for the TCP/IP protocol connection parameters.

Table 5–4 Syntax for TCP/IP Protocol Connection Parameters

Parameter	**Description**
PROTOCOL	The protocol adapter to be used. The value is TCP. It is not case sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP port. Either a number or the name specified in the `/etc/services` file. Oracle Corporation recommends a value of 1521.

Example 5–3 shows a sample TCP/IP ADDRESS.

Example 5–3 TCP/IP ADDRESS Specifying a Client

```
(ADDRESS=
    (PROTOCOL=TCP)
    (HOST=MADRID)
    (PORT=1521)
)
```

You can specify the last field by name, for example, (PORT=listener)**.**

Oracle Enterprise Manager (Solaris Intel Only)

The following section describes how to configure Oracle Enterprise Manager.

Agent Service Discovery and Auto-Configuration

You do not have to configure Oracle Intelligent Agent, unless you want to integrate it with a Simple Network Management Protocol (SNMP) system. See "Configuring Oracle Intelligent Agent for Oracle SNMP" on page 5-11 for more information.

> **See Also:** For information on Oracle Names and the Net8 Assistant, see the *Net8 Administrator's Guide*.

Debugging Tcl Scripts

The `oratclsh` executable is provided for debugging your Tcl scripts. Before executing `oratclsh`, set the TCL_LIBRARY environment variable to specify the `$ORACLE_HOME/network/agent/tcl` directory.

> **See Also:** See the *Oracle Enterprise Manager Application Developer's Guide* for more information on debugging Tcl scripts.

Configuring Oracle Intelligent Agent for Oracle SNMP

Although Oracle Intelligent Agent does not require Simple Network Management Protocol (SNMP) to work, you can configure Oracle SNMP support before starting the Intelligent Agent. Note that all of the configuration files for the following steps are located in the `$ORACLE_HOME/network/snmp/peer` directory.

Configure Master Agent

In the `CONFIG.master` file, make the following change:

1. Search for the line beginning with `MANAGER`.
2. Change the `ipaddr` field, coded as `130.35.10.210`, to the IP address or hostname of the system where you want SNMP trap messages sent.

You can also make other changes to the `CONFIG.master` file as documented within the file.

Configure the Encapsulator

To configure the encapsulator, perform the following steps:

1. Add the following line to the `snmpd.conf` file, where *hostname_or_IP_address* represents the local system IP address:

   ```
   trap hostname_or_IP_address
   ```

2. In the `CONFIG.encap` file, you can modify the port number which is set to `1161` in the default file. If you modify the port number, you must also modify the port number for NEW_SNMPD_PORT in the `start_peer` script.

   NEW_SNMPD_PORT is the port on which the `snmpd` agent (the native Intel UNIX SNMP agent) listens. Make sure that this is the same port as specified in the `CONFIG.encap` file. NEW_TRAPD_PORT is the PEER encapsulator port to which the `snmpd` agent sends traps.

   NEW_SNMPD_PORT and NEW_TRAPD_PORT in the `start_peer` script must have different port numbers. You can also modify the NEW_TRAPD_PORT port number.

Verify start_peer Script

The `start_peer` script contains a line similar to the following:

```
SNMPD = snmpd_executable_path
```

If the `snmpd` executable on your system is not in the location indicated by this line, specify the correct location of the `snmpd` executable.

Start the SNMP Components

To start the SNMP components, perform the following steps:

1. Verify that the SNMP components (`master_peer`, `encap_peer`, and `snmpd`) , are *not* running:

   ```
   $ ps -aef | grep peer
   $ ps -aef | grep snmp
   ```

   If any of the components are running, log in as the `root` user and use the `kill` command to terminate the processes before proceeding.

2. As the `root` user, run the `start_peer` script to start the PEER master agent, PEER encapsulator, and native Intel UNIX SNMP agent:

   ```
   # cd $ORACLE_HOME/network/snmp/peer
   # ./start_peer -a
   ```

   Note: If you do not have the native Intel UNIX SNMP agent on your system, you must *not* use the PEER encapsulator. To start the master agent only, run `start_peer -m`.

   ---

3. Verify that the SNMP components are running:

   ```
   # ps -aef | grep peer
   # ps -aef | grep snmp
   ```

Configure and Start the Database Subagent

The configuration and startup of the database subagent (the Oracle Intelligent Agent) is described in the *Oracle Enterprise Manager Configuration Guide*.

Oracle Advanced Security

The following section describes the Oracle Advanced Security Option.

.bak Files

When you install Oracle Advanced Security, three `.bak` files are created: `naeet.o.bak`, `naect.o.bak`, and `naedhs.o.bak`. They are located in the `$ORACLE_HOME/lib` directory. These files are required for relinking during Oracle Advanced Security de-installation and should not be deleted.

Security and Single Sign-On

For more information on configuring Security and Single Sign-On, see the *Oracle Advanced Security Administrator's Guide*.

6

Running Oracle Data Option Demonstrations

This chapter describes Oracle Data Option demonstrations. It contains the following sections:

- Oracle8i interMedia
- Oracle8i Time Series Demonstrations
- Oracle8i Visual Information Retrieval
- Oracle8i Spatial

Oracle8*i* *inter*Media

Oracle8*i* *inter*Media includes the following components:

- Text
- Audio, Video, and Image
- Locator
- Web Agent and Clipboard

Text

There are no demonstrations for *inter*Media Text in Oracle8*i*. However, *inter*Media Text now includes code samples. Use a web browser to open the following file:

```
$ORACLE_HOME/ctx/sample/api/index.html
```

> **See Also:** See the *Oracle8i interMedia Text Reference*, and *Oracle8i ConText interMedia Text Migration* for more information on *inter*Media text.

Audio, Video, and Image

Oracle8*i* *inter*Media includes audio, video, and image scripts and sample programs in the following directories:

```
$ORACLE_HOME/ord/aud/demo/
$ORACLE_HOME/ord/img/demo/
$ORACLE_HOME/ord/vid/demo/
```

Sample Audio Scripts

The *inter*Media audio scripts consist of the following files:

- `auddemo.sql` - demonstration that shows features of the audio object including:
    - Checking *inter*Media objects
    - Creating a sample table containing audio
    - Inserting NULL rows into the audio table
    - Checking the rows out
    - Checking all the audio attributes directly

  - Checking all the audio attributes by calling methods
  - Installing your own format plug-in using the `fplugins.sql` and `fpluginb.sql` files
- `fplugins.sql` - demonstration format plug-in specification that you can use as a guideline to write any format plug-in you want to support.
- `fpluginb.sql` - demonstration format plug-in body that you can use as a guideline to write any format plug-in you want to support.

See the `README.txt` file in the `$ORACLE_HOME/ord/aud/demo` directory for requirements and instructions on running this SQL demonstration.

Sample Program for Modifying Images or Testing the Image Installation

After you have installed Oracle8*i* *inter*Media Image, you can run the Oracle8*i* *inter*Media Image demonstration program. You can also use this program to confirm successful installation.

This section lists the steps required to build and run the *inter*Media Image demonstration.

The *inter*Media Image demonstration files are located in the `$ORACLE_HOME/ord/img/demo` directory, where `$ORACLE_HOME` is the Oracle home directory.

Demonstration Installation Steps

To install the Oracle8*i* *inter*Media Image demonstration:

1. Create the SCOTT/TIGER database user, if this user does not exist:

   ```
   $ svrmgrl
   SVRMGRL> CONNECT INTERNAL;
   SVRMGRL> CREATE USER SCOTT IDENTIFIED BY TIGER;
   SVRMGRL> GRANT CONNECT,RESOURCE TO SCOTT;
   ```

2. Create the image demonstration directory in the Oracle home directory:

   ```
   $ svrmgrl
   SVRMGRL> CONNECT INTERNAL;
   SVRMGRL> CREATE OR REPLACE DIRECTORY IMGDEMODIR AS '$ORACLE_
   HOME/ORD/IMG/DEMO';
   ```

3. Grant privileges on the directory to PUBLIC:

   ```
   SVRMGRL> grant read on directory imgdemodir to public with grant option;
   ```

4. If required, make the `imgdemo` program:

```
$ cd $ORACLE_HOME/ord/img/demo
$ make -f demo_ordimg.mk imgdemo
```

Running the Demonstration

The `imgdemo` file is a sample program that shows how Oracle8*i inter*Media Image can be used from within a program. The demonstration is written in C and uses the Oracle Call Interface (OCI) to access the database and exercise Oracle8*i inter*Media Image.

The program operates on the `imgdemo.dat` file, which is a bitmap (BMP) image in the demonstration directory. Optionally, you can supply the name of an image file on the command line, provided that the file is located in the same directory as the demonstration. In either way, after the image has been manipulated by Oracle8*i inter*Media Image, the resulting image is written to the `imgdemo.out` file. You can then view the image with an image viewer such as `xv`.

When you run the demonstration, it deletes and re-creates a table named IMGDEMOTAB in the SCOTT/TIGER schema of the default database. This table is used to hold the demonstration data. After the table is created, a reference to the image file is inserted into the table. The data is then loaded into the table and converted to the JFIF format using the processCopy( ) method of ORDImage.

The demonstration extracts the image properties from the database using the setProperties( ) method. After the demonstration runs setProperties( ) it issues an UPDATE command. This is required because setProperties( ) only updates a local copy of the type attributes.

Next, the demonstration uses the Oracle8*i inter*Media Image process( ) method to cut and scale the image within the database. The demonstration then runs an update that commits the change. The demonstration cuts a portion of the image 100 pixels wide by 100 pixels high starting from pixel location (100,100). This sub-image is scaled to twice its original size and the resulting image is written to a file named `imgdemo.out` in the current directory.

Executing the Demonstration from the Command Line

To execute the demonstration, enter:

```
$ imgdemo [image-file]
```

The optional image file must be located in the same directory as the demonstration program. The demonstration displays a number of messages describing its progress, along with any errors it encounters. Expect to see the following messages:

```
Dropping table IMGDEMOTAB...
Creating and populating table IMGDEMOTAB...
Loading data into cartridge...
Modifying image characteristics...
Writing image to file imgdemo.out...
Disconnecting from database...
Logged off and detached from server.
Demo completed successfully.
```

If the program encounters any errors, it is likely that either Oracle8*i* *inter*Media software has not been installed correctly or the database has not been started. If the program completes successfully, you can view the original image and the resultant image which has undergone the cutting and scaling described previously, with an image viewer such as `xv`.

Sample Video Scripts

The *inter*Media Video scripts consist of the following files:

- `viddemo.sql` - demonstration that shows features of the video object including:
  - Checking *inter*Media objects
  - Creating a sample table with video in it
  - Inserting NULL rows into the video table
  - Checking the rows out
  - Checking all the video attributes directly
  - Checking all the video attributes by calling methods
  - Installing your own format plug-in using the two files, `fplugins.sql` and `fpluginb.sql`
- `fplugins.sql` - demonstration format plug-in specification that you can use as a guideline to write any format plug-in you want to support.

- `fpluginb.sql` - demonstration format plug-in body that you can use as a guideline to write any format plug-in you want to support

> **See Also:** See the README.txt file in the `$ORACLE_HOME/ord/vid/demo` directory for requirements and instructions on running this SQL demonstration.

Java Demonstration

Oracle8*i inter*Media Locator includes a Java demonstration to help you learn to use both the audio and video client-side Java classes for building your own applications. These two demonstrations instantiate the audio and video object at the client side and start a number of accessor methods. The audio Java demonstration files are located in the `$ORACLE_HOME/ord/aud/demo` directory and the video Java demonstration files are located in the `$ORACLE_HOME/ord/vid/demo` directory.

> **See Also:** See the `README.txt` file in each directory for requirements and instructions on running each Java demonstration.

MediaAnnotator

The MediaAnnotator program is not contained on the Oracle8*i inter*Media CD. It (along with other free Oracle software) is available at the following URL:

`http://technet.oracle.com/software/products/intermedia/software_index.htm`

> **See Also:** See the *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* and *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference* for more information on MediaAnnotator.

Locator

Oracle8*i inter*Media Locator includes a number of scripts that you can modify and run.

> **See Also:** See the *Oracle8i interMedia Locator User's Guide and Reference* for more information on *inter*Media Locator.

Sample Scripts

Sample Oracle8*i inter*Media Locator scripts are available in the following directory after you install *inter*Media Locator:

`$ORACLE_HOME/md/demo/geocoder`

These scripts consist of the following files:

- `geohttp.sql`

  This file contains two parts that use different modes to run the geocode function. These modes are:

  - Interactive mode

    See the *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this part of the file.

  - Batch mode

    You must update the setup tables in the `nh_cs.sql` file before you run the `geohttp.sql` in batch mode. See the *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this part of the file.

- `geoindex.sql`

  This file contains:

  - A function named ESTIMATE_LEVEL to better estimate the index level for use with the spatial locator index for within-distance queries that use a radius distance greater than 100 miles. For a listing of this file, see the *Oracle8i interMedia Locator User's Guide and Reference*.

  - A procedure statement named SETUP_LOCATOR_INDEX that builds a setup spatial locator index on the location column that contains the spatial information within the CUST_TABLE table where the spatial information is stored. For a listing of this file, see the *Oracle8i interMedia Locator User's Guide and Reference*.

- `geolocate.sql`

  This file contains a routine that dynamically creates a geometry of interest and then queries against the NH_COMPUTER_STORES table to find out how many stores are within a 10-mile radius of the office. For a listing of this file, see the *Oracle8i interMedia Locator User's Guide and Reference*.

Web Agent and Clipboard

The Web Agent and Clipboard are not contained on the Oracle8*i* *inter*media CD. They (along with other free Oracle software) are available at the following URL:

`http://technet.oracle.com/software/products/intermedia/software_index.htm`

> **See Also:** See *Using Oracle8i interMedia with the Web* for more information on Web Agent.

Oracle8*i* Time Series Demonstrations

Table 6–1 shows the demonstrations included with Oracle8*i* Time Series. This table includes a description of each demonstration and the default directory in which its files are installed.

In the following table, all directories listed are sub directories of the `$ORACLE_HOME/ord/ts/demo` directory:

Table 6–1 Oracle8i Time Series Demonstrations

Description	Sub Directory
Quick-start demonstration: quick start using Oracle8*i* Time Series. (See the *Oracle8i Time Series User's Guide* for more information).	`tsquick`
Usage demonstration for end users and product developers who want to use existing Oracle8*i* Time Series features. (See the *Oracle8i Time Series User's Guide*).	`usage`
Electric utility application demonstrating how to compute peak and off-peak summaries of 15-minute data.	`usageutl`
Java-based retrieval of time series data, using the prototype Oracle8*i* Time Series Java API and designed to run in a Web browser. (See the *Oracle8i Time Series User's Guide* for more information).	`applet`
Simple Java code segments that perform time series operations and print the results. (See the *Oracle8i Time Series User's Guide* for more information).	`java`
Demonstration showing the use of administrative tools procedures to "retrofit" existing time series detail tables. Demonstration showing how to support time series queries for multiple qualifier columns in the time series detail table.	`retrofit`
Advanced-developer demonstration for those who want to extend Oracle8*i* Time Series features.	`extend`

Table 6–1 Oracle8i Time Series Demonstrations (Cont.)

Description	**Sub Directory**
OCI demonstration showing how to call Oracle8*i* Time Series functions using the Oracle Call Interface.	`oci`
PRO*C / C++ demonstration showing how to call Oracle8*i* Time Series functions in applications created using the Oracle Pro*C / C++ Precompiler.	`proc`
Oracle Developer demonstration showing how to call Oracle8*i* Time Series functions in an Oracle Forms application.	`dev2k`

The `README.txt` file in the demonstration directory introduces the demonstrations. In addition, the directory for each demonstration contains a `README.txt` file with a more detailed description of that demonstration.

> **See Also:** See the *Oracle8i Time Series User's Guide* for more information on Oracle Time Series demonstrations

Oracle8*i* Visual Information Retrieval

A sample program is included with Visual Information Retrieval. The sample program demonstrates how to load two images into the database, generate their signatures, and then compare their signatures using a weighted similarity function.

The sample program uses two data files, `virdemo1.dat` and `virdemo2.dat`, as its input. No other input or parameters are required.

Environment

The following assumptions are made about the environment:

- Visual Information Retrieval has been installed and PUBLIC has EXECUTE privilege on it.
- The installation script has been run. The VIRDEMODIR directory has been created and granted PUBLIC READ access in order that the image data file can be read into the database.
- The `virdemo1.dat` and `virdemo2.dat` files are valid image files that are located in the VIRDEMODIR directory and the user has read and write access to the directory.
- User SCOTT has the default TIGER password. You may need to increase the tablespace allocated to SCOTT to successfully run this sample program.

Running the Sample Program

There are two ways to run the sample program: using the included sample images, or using your own images.

Example 6–1 runs the sample program using the included image files. The images are compared using equal attribute weights:

- Globalcolor = 1.0
- Localcolor = 1.0
- Texture = 1.0
- Structure = 1.0

Example 6–1 Run the Sample Program with Included Images

```
% virdemo
Image 1 and 2 have a similarity score of 0.0
```

Example 6–2 shows how to specify your own images on the command line. The images be located in the `$ORACLE_HOME/ord/vir/demo` directory.

Example 6–2 Run the Sample Program with Your Own Images

```
% virdemo image1 image2 global_color local_color texture structure
```

You must specify all six parameters; the two file names and the four attribute weights (ranging from 0.0 to 1.0) in this sample program. Note that when using the VIRScore( ) operator in your own applications, you only need to provide one attribute weight.

The Visual Information Retrieval demonstration directory provides several other sample image files to demonstrate the effects of emphasizing the different visual attributes. You can use an image viewer (such as `xv`) to display the images, and then compare them using the sample program, experimenting with different weights.

> **See Also:** See the *Oracle8i Visual Information Retrieval User's Guide and Reference* and *Oracle8i Visual Information Retrieval Java Client User's Guide and Reference* for more information on Visual Information Retrieval.

Oracle8*i* Spatial

For information on Oracle8*i* Spatial, see the `$ORACLE_HOME/md/demo/readme.txt` file and the *Oracle8i Spatial User's Guide and Reference*.

A

Optimal Flexible Architecture

This appendix contains information on Optimal Flexible Architecture. It contains the following sections:

- Optimal Flexible Architecture
- Optimal Flexible Architecture Implemented on UNIX

Optimal Flexible Architecture

Oracle Corporation recommends that the Optimal Flexible Architecture (OFA) standard be implemented when installing and configuring Oracle8*i*. The OFA standard is a set of configuration guidelines created to ensure fast, reliable Oracle databases that require little maintenance.

OFA is designed to:

- Organize large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- Facilitate routine administrative tasks such as software and data backup, which are often vulnerable to data corruption
- Facilitate switching between multiple Oracle databases
- Adequately manage and administer database growth
- Help eliminate fragmentation of free space in the data dictionary, isolate other fragmentation, and minimize resource contention

Characteristics of an OFA-Compliant Database

An OFA-compliant database provides the following benefits:

File System Organization

The file system is organized to enable easy administration for issues such as:

- Adding data into existing databases
- Adding users
- Creating databases
- Adding hardware

Distributed I/O Loads

I/O loads are distributed across enough disk drives to prevent performance bottlenecks.

Hardware Support

In most cases, investment in new hardware is not required to take advantage of the OFA standard.

Safeguards Against Drive Failures

By distributing applications across more than one drive, drive failures affect as few applications as possible.

Distribution of Home Directories

The following items can be distributed across more than one disk drive:

- The collection of home directories
- The contents of an individual home directory

Integrity of Login Home Directories

It is possible to add, move, or delete login home directories without having to revise programs that refer to them.

Independence of UNIX Directory Subtrees

Categories of files are separated into independent UNIX directory subtrees so that files in one category are minimally affected by operations on files in other categories.

Supports Concurrent Execution of Application Software

You can execute multiple versions of application software simultaneously, enabling the user to test and use a new release of an application before abandoning the previous version. Transferring to a new version after an upgrade is simple for the administrator and transparent for the user.

Distinguishes Administrative Information for Each Database

The ability to separate administrative information on one database from that of another ensures a reasonable structure for the organization and storage of administrative data.

Uses Consistent Database File Naming

Database files are named so that:

- Database files are easily distinguishable from all other files
- Files of one database are easily distinguishable from files of another database
- Control files, redo log files, and data files are identifiable as such
- The association of data file to tablespace is clearly indicated

Separation of Tablespace Contents

Tablespace contents are separated to:

- Minimize tablespace free space fragmentation
- Minimize I/O request contention
- Maximize administrative flexibility

I/O Loads Tuned Across All Drives

I/O loads are tuned across all drives, including drives storing Oracle data in raw devices.

Additional Benefits of OFA for Parallel Server (DG/UX Only)

For Oracle Parallel Server Installations:

- Administrative data is stored in a central place, accessible to all database administrators
- Administrative data for an instance is associated with the instance by the file name

Optimal Flexible Architecture Implemented on UNIX

A careful naming strategy for database files eliminates data administration problems. The OFA rules provided here correspond to the original OFA recommendations published in *The OFA Standard: Oracle8 for Open Systems.*

Mount Points

This section describes the naming conventions for mount points.

Create Mount Points

An installation of Oracle8*i* requires at least two mount points: one for the software and at least one for the database files. If implementing the recommended OFA, at least four mount points are required: one for the software and at least three for database files.

Mount Point Syntax

Name all mount points using the syntax `/`*`pm`*, where *`p`* is a string constant and *`m`* is a unique fixed-length key (typically a two-digit number) used to distinguish each mount point. For example: `/u01` and `/u02`, or `/disk01` and `/disk02`.

Naming Mount Points for Very Large Databases (VLDBs)

If each disk drive contains database files from one application and there are enough drives for each database to ensure no I/O bottleneck, then use the syntax `/`*`q`*`/`*`dm`* for naming mount points. Table A–1 describes the variables used in this syntax.

Table A–1 Syntax for Naming Mount Points

Variable	Description
q	A string denoting that Oracle data is stored here
dm	The value of the initialization parameter DB_NAME (synonymous with the instance *sid* for single-instance databases)

For example, mount points named `/u01/oradata/test` and `/u02/oradata/test` allocate two drives for the Oracle test database.

Naming Directories

This section describes the naming conventions for OFA compliant directories.

Home Directory Syntax

Name home directories using the syntax `/pm/h/u`. Table A–2 describes the variables used in this syntax.

Table A–2 Syntax for Naming Home Directories

Variable	Description
pm	A mount point name
h	A standard directory name
u	The name of the owner of the directory

For example, `/u01/app/oracle` is the Oracle Server software owner home directory (also referred to as the Oracle base directory and the default used by the Installer) and `/u01/app/applmgr` is an Oracle applications software owner home directory.

Placing home directories at the same level in the UNIX file system is advantageous because it allows the collection of applications owner login home directories on different mount points to be referred to with the single pattern matching string, `/*/app/*`.

Referring to Path Names

Refer to explicit path names only in files designed specifically to store them, such as `/etc/passwd` and the Oracle `oratab` file. Refer to group memberships only in the `/etc/group` file.

Software Directories

To help fulfill the OFA feature of simultaneously executing multiple versions of application software, store each version of the Oracle8*i* Server software in a directory matching the pattern `/pm/h/product/v`.

Table A–3 describes the variables used in this syntax.

Table A–3 Syntax for Naming Oracle8i Server Software Directories

Variable	Description
pm	A mount point name
h	A standard directory name
v	The version of the software

For example, `/u01/app/oracle/product/817` indicates the start of the directory structure where the Oracle8*i* Server files are located. Set the ORACLE_HOME environment variable to this directory.

Naming Subdirectories

To facilitate the organization of administrative data, Oracle Corporation recommends that you store database-specific administration files in subdirectories matching the pattern *h*`/admin/`*d*`/`*a*`/`, where *h* is the Oracle software owner's home directory, *d* is the database name (DB_NAME), and *a* is a subdirectory for each of the database administration files. Table A–4 describes the database administration file subdirectories:

Table A–4 Subdirectories for Database Administration Files

Sub Directory	Description
`adhoc`	Ad hoc SQL scripts for a particular database
`arch`	Archived redo log files
`adump`	Audit files (Set AUDIT_FILE_DEST in the `configdb_name.ora` file to the `/adump` directory. Clean out this subdirectory periodically).
`bdump`	Background process trace files
`cdump`	Core dump files
`create`	Programs used to create the database
`exp`	Database export files
`logbook`	Files recording the status and history of the database
`pfile`	Instance parameter files
`udump`	User SQL trace files

For example, the subdirectory `adhoc` would have the following path name, `/u01/app/oracle/admin/sab/adhoc/` if it were part of the database named sab.

Naming Database Files

The following naming convention for database files ensures that they are easily identifiable:

- For control files, use `/`*`pm`*`/`*`q`*`/`*`d`*`/control.ctl`
- For redo log files, use `/`*`pm`*`/`*`q`*`/`*`d`*`/redo`*`n`*`.log`
- For data files, use `/`*`pm`*`/`*`q`*`/`*`d`*`/`*`tn`*`.dbf`

Table A–5 describes this syntax:

Table A–5 Syntax for Naming Database Files

Variable	**Description**
pm	A mount point name described previously in this chapter
q	A string distinguishing Oracle data from all other files (usually named ORACLE or oradata)
d	The DB_NAME of the database
t	An Oracle tablespace name
n	A two-digit string

Note: Do not store files other than a control, redo log, or data file associated with database *d* in the path */pm/q/d*.

Following this convention, you could produce, for example, a data file with the name `/u03/oradata/sab/system01.dbf`, making it easy to see to which database the file belongs.

Separate Segments with Different Requirements

Separate groups of segments with different lifespans, I/O request demands, and backup frequencies across different tablespaces.

Table A–6 describes the special tablespaces that you must create for each Oracle database. These tablespaces are in addition to those required for application segments.

Table A–6 Special Tablespaces

Tablespace	Description
SYSTEM	Data dictionary segments
TEMP	Temporary segments
RBS	Rollback segments
USERS	Miscellaneous user segments
INDX	Index associated with data in USERS tablespace
OEM_REPOSITORY	repository for Oracle Enterprise Manager
DRSYS	Oracle *inter*Media segment

This method is effective because dictionary segments are never dropped, and no other segments that can be dropped are allowed in the SYSTEM tablespace. This ensures that the SYSTEM tablespace does not require a rebuild due to tablespace free space fragmentation.

Because rollback segments are not stored in tablespaces holding applications data, the administrator is not blocked from taking an application's tablespace offline for maintenance. The segments are partitioned physically by type, and the administrator can record and predict data growth rates without complicated tools.

Naming Tablespaces

Name tablespaces descriptively using a maximum of eight characters. Although Oracle8*i* tablespace names can be 30 characters long, portable UNIX file names are restricted to 14 characters. The recommended standard for a data file basename is `tn.dbf`, where *`t`* is a descriptive tablespace name and *`n`* is a two-digit string. Because the extension plus the two-digit string occupy a total of six characters, only eight characters remain for the tablespace name.

Descriptive names enable the data file to be associated with the tablespace that uses it. For example, the names `GLD` and `GLX` might be used for the tablespaces storing General Ledger data and indices, respectively.

Note: Do not embed reminders of the word "tablespace" in your tablespace names. Tablespaces are distinguishable by context, and names do not need to convey information on type.

Exploiting the OFA Structure for Oracle Files

Table A–7 describes the syntax used for identifying classes of files.

Table A–7 Directory Structure Syntax for Identifying Classes of Files

Directory Structure Syntax	Description
`/u[0-9][0-9]`	User data directories
`/*/home/*`	User home directories
`/*/app/*`	User application software directories
`/*/app/applmgr`	Oracle apps software subtrees
`/*/app/oracle/product`	Oracle Server software subtrees
`/*/app/oracle/product/8.1.7`	Oracle Server 8.1.7 distribution files
`/*/app/oracle/admin/sab`	`sab` database administrative subtrees
`/*/app/oracle/admin/sab/arch/*`	`sab` database archived log files
`/*/oradata`	Oracle data directories
`/*/oradata/sab/*`	`sab` database files
`/*/oradata/sab/*.log`	`sab` database redo log files

OFA File Mapping

Table A–8 shows a hierarchical file mapping of a sample OFA-compliant database, including each file mount point, application, database, and tablespace. The file names indicate the file type (control, log, or data).

Table A–8 Hierarchical File Mapping for OFA Installation

Directory	Description
`/`	Root mount point
`/u01/`	User data mount point 1
`/u01/app/`	Subtree for app software
`/u01/app/oracle/`	Home for `oracle` software user
`/u01/app/oracle/admin/`	Subtree for database admin files
`/u01/app/oracle/admin/TAR/`	Subtree for Support logs
`/u01/app/oracle/admin/db_name1/`	Admin subtree for *db_name1* database
`/u01/app/oracle/admin/db_name2/`	Admin subtree for *db_name2* database
`/u01/app/oracle/doc/`	Online documentation
`/u01/app/oracle/local/`	Subtree for local Oracle software
`/u01/app/oracle/local/aps6/`	An Oracle6 admin package
`/u01/app/oracle/local/aps7/`	An Oracle7 admin package
`/u01/app/oracle/product/`	Distribution files
`/u01/app/oracle/product/7.3.3/`	ORACLE_HOME for 7.3.3 instances
`/u01/app/oracle/product/8.0.4/`	ORACLE_HOME for 8.0.4 instances
`/u01/app/oracle/product/8.1.6/`	ORACLE_HOME for 8.1.6 instances
`/u01/app/ltb/`	Home directory for a user
`/u01/app/sbm/`	Home directory for a user
`/u01/oradata/`	Subtree for Oracle data
`/u01/oradata/`*db_name1/*	Subtree for *db_name1* database files
`/u01/oradata/`*db_name2/*	Subtree for *db_name2* database files
`/u02/`	User data mount point 2
`/u02/home/`	Subtree for login home directories
`/u02/home/cvm/`	Home directory for a user

Table A–8 Hierarchical File Mapping for OFA Installation (Cont.)

Directory	Description
`/u02/home/vrm/`	Home directory for a user
`/u02/oradata/`	Subtree for Oracle data
`/u02/oradata/db_name1/`	Subtree for *db_name1* database files
`/u02/oradata/db_name2/`	Subtree for *db_name2* database files
`/u03/`	User data mount point 3
`/u03/oradata/`	Subtree for Oracle data
`/u03/oradata/db_name1/`	Subtree for *db_name1* database files
`/u03/oradata/db_name2/`	Subtree for *db_name2* database files

Raw Device Sizes

Choose a small set of standard sizes for all raw devices that can be used to store Oracle database files. In general, standardizing on a single size is recommended. If a single size is used, raw files can be moved from one partition to another safely. The size should be small enough so that a fairly large number can be created but large enough to be convenient.

For example, a 2 GB drive could be divided into 10 partitions of 200 MB each. This is a good balance between size and number. Any tablespace using raw devices should stripe them across several drives. If possible, use a logical volume manager to do the stripping.

File Mapping for Multiple-Instance OFA Database (DG/UX Only)

When using the Oracle Parallel Server, select one node to act as the Oracle administrative home for the cluster. The administrative home contains the administrative subtree. Create subdirectories for each instance accessing the database within the `bdump`, `cdump`, `logbook`, `pfile`, and `udump` directories of `~/admin/`*`d`*`/`. Mount the `admin` directory for the administrative home as the `admin` directory for every instance. Table A–9 shows a sample directory structure.

Table A–9 Administrative Directory Structure for Dual-Instance Oracle Parallel Server

Directory Path	Description
`u01/app/oracle/admin/sab/`	Administrative directory for **sab** database
`u01/adhoc/`	Directory for miscellaneous scripts
`u01/arch/`	Log archive dest for all instances
`u01/arch/redo001.arc`	Archived redo log file
`u01/bdump/`	Directory for background dump files
`u01/bdump/inst1/`	Background dump dest for *inst1* instance
`u01/bdump/inst2/`	Background dump dest for *inst2* instance
`u01/cdump/`	Directory for core dump files
`u01/cdump/inst1/`	Core dump dest for *inst1* instance
`u01/cdump/inst2/`	Core dump dest for *inst2* instance
`u01/create/`	Directory for creation scripts
`u01/create/1-rdbms.sql`	SQL script to create *inst* database
`u01/exp/`	Directory for exports
`u01/exp/20000120full.dmp`	January 20, 2000 full export dump file
`u01/exp/export/`	Directory for export parfiles
`u01/exp/import/`	Directory for import parfiles
`u01/logbook/`	Directory for *inst* logbook entries
`u01/logbook/inst1/`	Directory for *inst1* instance reports
`u01/logbook/inst1/params.lst`	V$PARAMETER REPORT FOR *INST1* INSTANCES
`u01/logbook/inst2/`	Directory for *inst2* instance reports
`u01/logbook/inst2/params.lst`	v$PARAMETER REPORT FOR *INST2* INSTANCES
`u01/logbook/user.lst`	Dba_users report
`u01/pfile/`	Directory for instance parameter files
`u01/pfile/inst1/`	Directory for *inst1* instance parameters
`u01/pfile/inst1/init`	Instance parameters for *inst1* instance
`u01/pfile/inst2`	Directory for *inst2* instance parameters
`u01/pfile/inst2/init`	Instance parameters for *inst2* instance
`u01/udump/`	Directory for user dump files
`u01/udump/inst1/`	User dump dest for *inst1* instance
`u01/udump/inst2/`	User dump dest for *inst2* instance

Directory Structure

The following section describes the directory structure for OFA compliant installations.

ORACLE Base Directory

The Oracle base directory is the root of the Oracle directory structure. When installing an OFA-compliant database using the Oracle Universal Installer, the default Oracle base directory is set to *`/pm/app/oracle`*. Table A–10 describes an Oracle base directory structure and content.

Table A–10 Oracle Base Directory Structure and Content

Directory Structure Syntax	Description
`admin`	Administrative files
`doc`	Online documentation
`local`	Subtree for local Oracle software
`product`	Oracle software

Oracle Home Directory

If you install an OFA-compliant Oracle Server, the Oracle home directory is `/`*`pm`*`/app/oracle/product/`*`release_number`*. Table A–11 describes the Oracle home directory structure and content. Under UNIX, the Oracle home directory contains the subdirectories described in Table A–9, as well as a subdirectory for each Oracle product installed.

Table A–11 Oracle Home Directory Structure and Content

Directory	Description
`assistants`	Configuration Assistants
`bin`	Binaries for all products
`ctx`	*inter*Media Text options
`dbs`	`init`*`sid`*`.ora`, `lk`*`sid`*
`install`	Installation related files
`lib`	Oracle product libraries
`jlib`	Java classes
`md`	Spatial options
`mlx`	Xerox Stemmer (for *inter*Media Text options)

Table A–11 Oracle Home Directory Structure and Content (Cont.)

Directory	Description
`network`	Net8
`nlsrtl`	NLS runtime loadable data
`ocommon`	Common files for all products
`odg`	Data gatherer
`oracore`	Core libraries
`ord`	Data options
`otrace`	Oracle TRACE
`plsql`	PL/SQL
`precomp`	Precompilers
`rdbms`	Server files and libraries required for the database
`slax`	SLAX parser
`sqlplus`	SQL*Plus

Contents of Product Subdirectories

Table A–12 describes the subdirectories contained in each product subdirectory:

Table A–12 Contents of Product Subdirectories

Directory	Description
`admin`	Administrative SQL and shell scripts (for example, `catalog.sql`, `catexp.sql`, and `demo.sql`)
`admin/*`	Special directories for other products
`admin/resource`	Resource files
`admin/terminal`	Runtime terminal files
`demo`	Demonstration scripts and datafiles
`doc`	README files (for example, `readmeunix.doc`)
`install`	Product installation scripts
`jlib`	Product Java classes
`lib`	Product libraries and distributed makefiles
`log`	Trace files and log files (for example, `orasrv.log` and `*.trc` files)

Table A–12 Contents of Product Subdirectories (Cont.)

Directory	Description
`mesg`	U.S. message files and Multilingual Option (formerly National Language Support) message text and binary files (for example, `oraus.msg` and `oraus.msb`)

Examples of Product Subdirectories

Table A–13 shows examples of product subdirectories and their contents.

Table A–13 Examples of Product Subdirectories

Directory	Description
`rdbms`	`admin`, `doc`, `install`, `lib`, `log`, `mesg`
`sqlplus`	,`admin`, demo, `doc`, `install`, `lib`, `mesg`

File Naming Conventions in the admin Directory

Table A–14 shows the SQL scripts located in the `rdbms/admin` directory.

Table A–14 admin Directory, File Naming Conventions

File	Description
`cat*.sql`	Creates catalog and data dictionary tables and views. The following files are run automatically during installation: `catalog.sql` (for all installations) `catproc.sql` (for all installations) `catparr.sql` (for Parallel Server option installations) `catrep.sql` (for all installations) The `catproc.sql` file in turn runs the scripts for creating the standard PL/SQL packages, such as DBMS_SQL and DBMS_OUTPUT.
`d*.sql`	Downgrade scripts
`dbms*.sql`	Additional database packages
`u*.sql`	Upgrade scripts
`utl*.sql`	Creates tables and views for database utilities

Filename Extensions

Table A–15 describes filename extensions.

Table A–15 Filename Extensions

Extension	Description
`.a`	Object file libraries; Ada runtime libraries
`.aud`	Oracle audit file
`.bdf`	X11 font description file
`.bmp`	X11 bitmap file
`.c`	C source file
`.ctl`	SQL*Loader control file; Oracle Server control file
`.dat`	SQL*Loader datafile
`.dbf`	Oracle Server tablespace file
`.dmp`	Export file
`.doc`	ASCII text file
`.env`	Shell script file for setting environment
`.h`	C header file; also, `sr.h` is a SQL*Report Writer help file
`.jar`	Java class archive
`.l`	UNIX manual page
`.lis`	Output of SQL*Plus scripts
`.log`	Installation log files; Oracle Server redo log files
`.mk`	`make` files
`.msb`	NLS message file (binary)
`.msg`	NLS message file (text)
`.o`	Object module
`.ora`	Oracle configuration files
`.orc`	Installation prototype files
`.pc`	Pro*C source file
`.pco`	Pro*COBOL source file
`.ppd`	printer driver file
`.sh`	Bourne shell script file

Table A–15 Filename Extensions (Cont.)

Extension	Description
`.sql`	SQL* script files
`.sys`	Bourne shell script file
`.tab`	SQL* script file
`.trc`	Trace files
`.tut`	Bourne shell script file
`.utd`	Uniform Terminal Definitions
`.zip`	Zip file

Index