

Oracle® Database

Database Administrator's Reference



19c for Linux and UNIX System-Based Operating Systems

E96356-01

January 2019

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Database Database Administrator's Reference, 19c for Linux and UNIX System-Based Operating Systems

E96356-01

Copyright © 2006, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Bharathi Jayathirtha

Contributing Authors: Subhash Chandra, Prakash Jashnani, Tanaya Bhattacharjee

Contributors: Kevin Flood, Pat Huey, Clara Jaeckel, Emily Murphy, Terri Winters, Subhranshu Banerjee, Mark Bauer, Robert Chang, Jonathan Creighton, Sudip Datta, Thirumaleshwara Hasandka, Joel Kallman, George Kotsovolos, Richard Long, Rolly Lv, Padmanabhan Manavazhi, Matthew Mckerley, Krishna Mohan, Rajendra Pingte, Hanlin Qian, Janelle Simmons, Roy Swonger, Michael Coulter, Robert Achacoso, Malai Stalin, Ramesh Chakravarthula, David Price, Douglas Williams, Joseph Therrartil Koonen, Binoy Sukumaran, and Sumanta Chatterjee.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	ix
Documentation Accessibility	ix
Set Up Java Access Bridge to Implement Java Accessibility	ix
Related Documentation	x
Conventions	x
Command Syntax	x
Terminology	xi
Accessing Documentation	xi

1 Administering Oracle Database

1.1 Overview	1-1
1.2 Environment Variables	1-1
1.2.1 Oracle Database Environment Variables	1-2
1.2.2 Linux and UNIX Environment Variables	1-5
1.2.3 Setting a Common Environment	1-6
1.2.4 Setting the System Time Zone	1-7
1.3 Initialization Parameters	1-8
1.3.1 ASM_DISKSTRING Initialization Parameter	1-8
1.3.2 DISK_ASYNCH_IO Initialization Parameter (HP-UX)	1-8
1.3.3 PROCESSOR_GROUP_NAME Initialization Parameter	1-9
1.3.4 Managing Diagnostic Data	1-9

2 Stopping and Starting Oracle Software

2.1 Stopping and Starting Oracle Processes	2-1
2.1.1 Stopping and Starting Oracle Database and Oracle Automatic Storage Management Instances	2-1
2.1.1.1 Stopping an Oracle Database or Oracle Automatic Storage Management Instance	2-2
2.1.1.2 Restarting an Oracle Database or Oracle Automatic Storage Management Instance	2-3
2.1.2 Stopping and Starting Oracle Restart	2-3

2.2	About Automating Database Shutdown and Startup	2-3
2.2.1	Automating Database Startup and Shutdown	2-4

3 Configuring Oracle Database

3.1	Using Configuration Assistants	3-1
3.1.1	Using Oracle Net Configuration Assistant	3-1
3.1.2	Using Oracle Database Upgrade Assistant	3-1
3.1.3	Using Oracle Database Configuration Assistant	3-2
3.2	Relinking Executables	3-2

4 Administering SQL*Plus

4.1	Administering Command-Line SQL*Plus	4-1
4.1.1	Using Setup Files	4-1
4.1.2	Using the PRODUCT_USER_PROFILE Table	4-2
4.1.3	Using Oracle Database Sample Schemas	4-2
4.1.4	Installing and Removing SQL*Plus Command-Line Help	4-2
4.1.4.1	Installing SQL*Plus Command-Line Help	4-3
4.1.4.2	Removing SQL*Plus Command-Line Help	4-3
4.2	Using Command-Line SQL*Plus	4-3
4.2.1	Using a System Editor from SQL*Plus	4-4
4.2.2	Running Operating System Commands from SQL*Plus	4-4
4.2.3	Interrupting SQL*Plus	4-5
4.2.4	Using the SPOOL Command	4-5
4.3	SQL*Plus Restrictions	4-5
4.3.1	Resizing Windows	4-5
4.3.2	Return Codes	4-5
4.3.3	Hiding the Password	4-5

5 Configuring Oracle Net Services

5.1	Locating Oracle Net Services Configuration Files	5-1
5.2	Using Adapters Utility	5-2
5.3	Using Oracle Protocol Support	5-2
5.3.1	IPC Protocol Support	5-3
5.3.2	TCP/IP Protocol Support	5-3
5.3.3	TCP/IP with Secure Sockets Layer Protocol Support	5-3
5.4	Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer	5-4

6 Using Oracle Precompilers and the Oracle Call Interface

6.1	Overview of Oracle Precompilers	6-1
6.1.1	Precompiler Configuration Files	6-2
6.1.2	Relinking Precompiler Executables	6-2
6.1.3	Issues Common to All Precompilers	6-3
6.1.4	Static and Dynamic Linking	6-3
6.1.5	Client Shared and Static Libraries	6-4
6.1.6	Generating the Client Static Library	6-5
6.2	Bit-Length Support for Client Applications	6-5
6.3	Pro*C/C++ Precompiler	6-6
6.3.1	Pro*C/C++ Demonstration Programs	6-6
6.3.2	Pro*C/C++ User Programs	6-8
6.4	Pro*COBOL Precompiler	6-9
6.4.1	Pro*COBOL Environment Variables	6-9
6.4.1.1	Micro Focus Server Express COBOL Compiler	6-10
6.4.1.2	Acucorp ACUCOBOL-GT COBOL Compiler	6-11
6.4.2	Pro*COBOL Oracle Runtime System	6-12
6.4.3	Pro*COBOL Demonstration Programs	6-12
6.4.4	Pro*COBOL User Programs	6-13
6.4.5	FORMAT Precompiler Option	6-14
6.5	Pro*FORTRAN Precompiler	6-14
6.5.1	Pro*FORTRAN Demonstration Programs	6-14
6.5.2	Pro*FORTRAN User Programs	6-15
6.6	SQL*Module for ADA	6-16
6.6.1	SQL*Module for Ada Demonstration Programs	6-16
6.6.2	SQL*Module for Ada User Programs	6-17
6.7	OCI and OCCI	6-18
6.7.1	OCI and OCCI Demonstration Programs	6-18
6.7.2	OCI and OCCI User Programs	6-19
6.8	Running Oracle JDBC/OCI Programs with a 64-Bit Driver	6-19
6.9	Custom Make Files	6-20
6.10	Correcting Undefined Symbols	6-20
6.11	Multithreaded Applications	6-21
6.12	Using Signal Handlers	6-21
6.13	XA Functionality	6-23

7 SQL*Loader and PL/SQL Demonstrations

7.1	SQL*Loader Demonstrations	7-1
7.2	PL/SQL Demonstrations	7-1

8 Tuning Oracle Database

8.1	Importance of Tuning	8-1
8.2	Operating System Tools	8-1
8.2.1	vmstat	8-2
8.2.2	sar	8-3
8.2.3	iostat	8-4
8.2.4	swap, swapinfo, swapon, or lps	8-4
8.2.5	Oracle Solaris Tools	8-4
8.2.6	Linux Tools	8-5
8.2.7	IBM AIX on POWER Systems (64-Bit) Tools	8-5
8.2.7.1	Base Operation System Tools	8-5
8.2.7.2	Performance Toolbox	8-6
8.2.7.3	System Management Interface Tool	8-7
8.2.8	HP-UX Tools	8-7
8.3	Tuning Memory Management	8-8
8.3.1	Allocating Sufficient Swap Space	8-8
8.3.2	Monitoring Paging	8-10
8.3.3	Adjusting Oracle Block Size	8-11
8.3.4	Allocating Memory Resource	8-11
8.4	Tuning Disk Input-Output	8-12
8.4.1	Using Automatic Storage Management	8-12
8.4.2	Choosing the Appropriate File System Type	8-12
8.5	Monitoring Disk Performance	8-13
8.5.1	Monitoring Disk Performance on Operating Systems	8-13
8.5.2	Using Disk Resync to Monitor Automatic Storage Management Disk Group	8-13
8.6	System Global Area	8-14
8.6.1	Determining the Size of the SGA	8-15
8.6.2	System Resource Verifier Utility	8-15
8.6.2.1	Purpose of the sysresv Utility	8-15
8.6.2.2	Preconditions for Using sysresv	8-15
8.6.2.3	Syntax for sysresv	8-15
8.6.2.4	Examples of Using sysresv	8-16
8.6.3	Guidelines for Setting Semaphore Parameters	8-16
8.6.4	Shared Memory on IBM AIX on POWER Systems (64-Bit)	8-17
8.7	Tuning the Operating System Buffer Cache	8-19

A Administering Oracle Database on Linux

A.1	Supporting Asynchronous Input-Output	A-1
A.2	Asynchronous Input-Output Support	A-2
A.3	Enabling Direct Input-Output Support	A-2
A.4	Enabling Simultaneous Multithreading	A-2
A.5	Allocating Shared Resources	A-3
A.6	About Creating Cgroups on Linux Systems	A-4
A.7	Overview of HugePages	A-4
A.7.1	Reviewing HugePages Memory Allocation	A-4
A.7.2	Using HugePages on Linux	A-5
A.7.3	Tuning SGA With HugePages	A-5
A.7.4	Configuring HugePages on Linux	A-6
A.7.5	Restrictions for HugePages Configurations	A-8
A.7.6	Disabling Transparent HugePages	A-8

B Administering Oracle Database on Oracle Solaris

B.1	Oracle Solaris Shared Memory Environment	B-1
B.1.1	About Optimized Shared Memory	B-1
B.1.2	Checking for Optimized Shared Memory	B-2
B.1.3	About ISM and DISM	B-2
B.1.4	Checking for ISM or DISM	B-2
B.1.5	About the oradism Utility	B-3
B.1.6	How Oracle Database Decides Between OSM, ISM and DISM	B-3
B.2	About Creating Solaris Resource Pools	B-4
B.3	About Multi-CPU Binding Functionality	B-4

C Using Oracle ODBC Driver

C.1	Oracle ODBC Features Not Supported	C-1
C.2	Implementation of Data Types	C-2
C.3	Limitations on Data Types	C-2
C.4	Format of the Connection String for the SQLDriverConnect Function	C-3
C.5	Reducing Lock Timeout in a Program	C-5
C.6	Linking ODBC Applications	C-5
C.7	Obtaining Information About ROWIDs	C-5
C.8	ROWIDs in a WHERE Clause	C-5
C.9	Enabling Result Sets	C-6
C.10	Enabling EXEC Syntax	C-12
C.11	Supported Functionality	C-12
C.11.1	API Conformance	C-13

C.11.2	Implementation of ODBC API Functions	C-13
C.11.3	Implementation of the ODBC SQL Syntax	C-14
C.11.4	Implementation of Data Types	C-14
C.12	Unicode Support	C-14
C.12.1	Unicode Support Within the ODBC Environment	C-14
C.12.2	Unicode Support in ODBC API	C-15
C.12.3	SQLGetData Performance	C-15
C.12.4	Unicode Samples	C-16
C.13	Performance and Tuning	C-21
C.13.1	General ODBC Programming Guidelines	C-22
C.13.2	Data Source Configuration Options	C-22
C.13.3	DATE and TIMESTAMP Data Types	C-24
C.14	Error Messages	C-25

D Database Limits

D.1	Database Limits	D-1
-----	-----------------	-----

Index

Preface

This guide provides platform-specific information about administering and configuring Oracle Database 19c on the following platforms:

- Oracle Solaris
- Linux
- IBM AIX on POWER Systems (64-Bit)
- HP-UX Itanium

This guide supplements the *Oracle Database Administrator's Guide*.

Audience

This guide is intended for anyone responsible for administering and configuring Oracle Database 19c. If you are configuring Oracle RAC, then refer to *Oracle Real Application Clusters Administration and Deployment Guide*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Set Up Java Access Bridge to Implement Java Accessibility

Install Java Access Bridge so that assistive technologies on Microsoft Windows systems can use the Java Accessibility API.

Java Access Bridge is a technology that enables Java applications and applets that implement the Java Accessibility API to be visible to assistive technologies on Microsoft Windows systems.

Refer to *Java Platform, Standard Edition Accessibility Guide* for information about the minimum supported versions of assistive technologies required to use Java Access Bridge. Also refer to this guide to obtain installation and testing instructions, and instructions for how to use Java Access Bridge.

Related Topics

- *Java Platform, Standard Edition Java Accessibility Guide*

Related Documentation

For important information, refer to your platform-specific Release Notes, Installation Guides, and Examples Installation Guide in the *Oracle Database Documentation Library*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Command Syntax

UNIX command syntax appears in monospace font. The dollar character (\$), number sign (#), or percent character (%) are UNIX command prompts. Do not enter them as part of the command. The following command syntax conventions are used in this guide:

Convention	Description
backslash \	A backslash is the UNIX command continuation character. It is used in command examples that are too long to fit on a single line. Enter the command as displayed (with a backslash) or enter it on a single line without a backslash: <pre>dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \ count=10000</pre>
braces { }	Braces indicate required items: <pre>.DEFINE {macro1}</pre>
brackets []	Brackets indicate optional items: <pre>cvtcrt termname [outfile]</pre>
ellipses ...	Ellipses indicate an arbitrary number of similar items: <pre>CHKVAL fieldname value1 value2 ... valueN</pre>
<i>italic</i>	Italic type indicates a variable. Substitute a value for the variable: <pre>library_name</pre>

Convention	Description
vertical line	A vertical line indicates a choice within braces or brackets: <code>FILE <i>filesize</i> [K M]</code>

Terminology

The names of some UNIX operating systems have been shortened in this guide. These are:

Operating System	Abbreviated Name
Oracle Solaris on SPARC (64-Bit)	Oracle Solaris
Oracle Solaris on x86-64 (64-Bit)	Note: Where the information for Oracle Solaris is different on a particular architecture, this is noted in the text.
Linux x86-64	Linux
	Note: Where the information for Linux is different on a particular architecture, this is noted in the text.

Accessing Documentation

The documentation for this release includes platform-specific documentation and generic product documentation. Platform-specific documentation includes information about installing, configuring, and using Oracle products on a particular platform. The documentation is available in Adobe portable document format (PDF) and HTML format.

All Oracle documentation is available at the following URL:

<http://docs.oracle.com/en/>

Note:

Platform-specific documentation is current at the time of release. For the latest information, Oracle recommends you to go to Oracle Technology Network website.

1

Administering Oracle Database

This chapter provides information about administering Oracle Database on UNIX-based operating systems. It contains the following sections:

- [Overview](#)
- [Environment Variables](#)
- [Initialization Parameters](#)
- [Managing Diagnostic Data](#)

See Also:

- The appropriate appendix in this guide for platform-specific information about administering Oracle Database
- *Oracle Database Administrator's Guide* and *Database 2 Day DBA* for generic information about administering Oracle Database

1.1 Overview

You must set Oracle Database environment variables, parameters, and user settings for Oracle Database to work. This chapter describes the various settings for Oracle Database.

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

```
SQL> ALTER TABLESPACE TEMP ADD DATAFILE '?/dbs/temp02.dbf' SIZE 200M
```

Similarly, the at sign (@) represents the `ORACLE_SID` environment variable. For example, to indicate a file that belongs to the current instance, run the following command:

```
SQL> ALTER TABLESPACE tablespace_name ADD DATAFILE tempfile@.dbf
```

1.2 Environment Variables

This section describes the most commonly used Oracle Database and operating system environment variables. You must define some environment variables before installing Oracle Database. This section covers the following topics:

- [Oracle Database Environment Variables](#)
- [Linux and UNIX Environment Variables](#)

- [Setting a Common Environment](#)
- [Setting the System Time Zone](#)

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, run the following command:

```
$ env | grep ORACLE_SID
```

To display the current value of all environment variables, run the `env` command as follows:

```
$ env | more
```

1.2.1 Oracle Database Environment Variables

The following table describes some environment variables used with Oracle Database.

Table 1-1 Oracle Database Environment Variables

Variable	Definition
NLS_LANG	<p>Function: Specifies the language, territory, and character set of the client environment. The client character set specified by <code>NLS_LANG</code> must match the character set of the terminal or terminal emulator. If required, <code>NLS_LANG</code> can be temporarily reset to another character set before starting a non-interactive batch program to match the character set of files and scripts processed by this program. The character set specified by <code>NLS_LANG</code> can be different from the database character set, in which case the character set is automatically converted.</p> <p>Refer to <i>Oracle Database Globalization Support Guide</i> for a list of parameters for this variable.</p> <p>Syntax: <i>language_territory.characterset</i></p> <p>Example: <code>french_france.we8iso8859p15</code></p>
ORA_NLS10	<p>Function: Specifies the directory where the language, territory, character set, and linguistic definition files are stored.</p> <p>Syntax: <i>directory_path</i></p> <p>Example: <code>\$ORACLE_HOME/nls/data</code></p>
ORA_TZFILE	<p>Function: Specifies the full path and file name of the time zone file. The Oracle Database Server always uses the large time zone file (<code>\$ORACLE_HOME/oracore/zoneinfo/timezlrgr_number.dat</code>). If you want to use the small time zone file on the client side, you must set this environment variable to the full path of the small time zone file (<code>\$ORACLE_HOME/oracore/zoneinfo/timezone_number.dat</code>). If you use the small time zone file on the client side, you must ensure that the database you access contains data only in the time zone regions recognized by the small time zone file.</p> <p>Syntax: <i>directory_path</i></p> <p>Example: <code>\$ORACLE_HOME/oracore/zoneinfo/timezlrgr_11.dat</code></p>
ORACLE_BASE	<p>Function: Specifies the base of the Oracle directory structure for Optimal Flexible Architecture compliant installations.</p> <p>Syntax: <i>directory_path</i></p> <p>Example: <code>/u01/app/oracle</code></p>

Table 1-1 (Cont.) Oracle Database Environment Variables

Variable	Definition
ORACLE_HOME	Function: Specifies the directory containing the Oracle software. Syntax: <i>directory_path</i> Example: \$ORACLE_BASE/product/19.0.0/dbhome_1
ORACLE_PATH	Function: Specifies the search path for files used by Oracle applications such as SQL*Plus. If the full path to the file is not specified, or if the file is not in the current directory, then the Oracle application uses ORACLE_PATH to locate the file. Syntax: Colon-separated list of directories: <i>directory1:directory2:directory3</i> Example: /u01/app/oracle/product/19.0.0/dbhome_1/bin:
ORACLE_SID	Function: Specifies the Oracle system identifier. Syntax: A string of numbers and letters that must begin with a letter. Oracle recommends a maximum of 8 characters for system identifiers. For more information about this environment variable, refer to <i>Oracle Database Installation Guide</i> . Example: SAL1
ORACLE_TRACE	Function: Enables the tracing of shell scripts during an installation. If it is set to T, then many Oracle shell scripts use the <code>set -x</code> command, which prints commands and their arguments as they are run. If it is set to any other value, or no value, then the scripts do not use the <code>set -x</code> command. Syntax: T or not T Example: T
ORAENV_ASK	Function: Controls whether the <code>oraenv</code> or <code>coraenv</code> script prompts or does not prompt for the value of the ORACLE_SID environment variable. If it is set to NO, then the scripts do not prompt for the value of the ORACLE_SID environment variable. If it is set to any other value, or no value, then the scripts prompt for a value for the ORACLE_SID environment variable. Syntax: NO or not NO Example: NO
SQLPATH	Function: Specifies the directory or list of directories that SQL*Plus searches for a <code>login.sql</code> file. Syntax: Colon-separated list of directories: <i>directory1:directory2:directory3</i> Example: /home:/home/oracle:/u01/oracle
TNS_ADMIN	Function: Specifies the directory containing the Oracle Net Services configuration files. Syntax: <i>directory_path</i> Example: \$ORACLE_HOME/network/admin

Table 1-1 (Cont.) Oracle Database Environment Variables

Variable	Definition
TWO_TASK	<p>Function: Specifies the default connect identifier to use in the connect string. If this environment variable is set, then do not specify the connect identifier in the connect string. For example, if the TWO_TASK environment variable is set to sales, then you can connect to a database by using the following command:</p> <pre>SQL> CONNECT username Enter password: password</pre> <p>Syntax: Any connect identifier.</p> <p>Range of Values: Any valid connect identifier that can be resolved by using a naming method, such as a tnsnames.ora file or a directory server.</p> <p>Example: PRODDB_TCP</p>
NLS_OS_CHARSET	<p>Function: Specifies the Oracle character set name corresponding to the UNIX locale character set in which the file names and user names are encoded by the operating system. You must set the environment variable NLS_OS_CHARSET, if the UNIX locale character set is different from the Oracle client character set. These two character sets may differ, for example, if NLS_LANG is set to a particular character set used to encode an SQL script, which is to be executed in an SQL*Plus session. Usually, the Oracle client character set and the operating system character set are the same and NLS_OS_CHARSET must not be set.</p> <p>Syntax: <i>character set</i></p> <p>Example: WE8ISO8859P1</p>

 **Note:**

- To prevent conflicts, do not define environment variables with names that are identical to the names of Oracle Database server processes. For example ARCH, PMON, and DBWR.
- If the Oracle installation owner (oracle user) environment variables are missing, or if they are set to incorrect paths, then you may encounter undefined errors during Oracle Database startup. Ensure that all the environment variables are correctly set before starting the database. Refer to My Oracle Support Note 373303.1 for more information:

<https://support.oracle.com/epmos/faces/DocumentDisplay?id=373303.1>

 **Caution:**

You must consult Oracle Support before you implement any changes as suggested in the My Oracle Support Note 373303.1.

1.2.2 Linux and UNIX Environment Variables

Oracle Database uses operating system environment variables to define system resources and software locations.

Table 1-2 Environment Variables Used with Oracle Database

Variable	Definition
ADA_PATH (IBM AIX on POWER Systems (64-Bit) only)	<p>Function: Specifies the directory containing the Ada compiler</p> <p>Syntax: <i>directory_path</i></p> <p>Example: <code>/usr/lpp/powerada</code></p>
CLASSPATH	<p>Function: Used with Java applications. The required setting for this variable depends on the Java application. Refer to the product documentation for Java application for more information.</p> <p>Syntax: Colon-separated list of directories or files: <i>directory1:directory2:file1:file1</i></p> <p>Example: There is no default setting. CLASSPATH must include the following directories: <code>\$ORACLE_HOME/jdk/jre/lib:\$ORACLE_HOME/jlib</code></p>
DISPLAY	<p>Function: Used by X-based tools. Specifies the display device used for input and output. Refer to the X Window System documentation for information.</p> <p>Syntax: <i>hostname:server[.screen]</i></p> <p>where <i>hostname</i> is the system name (either IP address or alias), <i>server</i> is the sequential code number for the server, and <i>screen</i> is the sequential code number for the screen. If you use a single monitor, then use the value 0 for both server and screen (0.0).</p> <p>Note: If you use a single monitor, then <i>screen</i> is optional.</p> <p>Example:</p> <pre>192.0.2.1:0.0 password:0</pre>
HOME	<p>Function: The home directory of the user.</p> <p>Syntax: <i>directory_path</i></p> <p>Example: <code>/home/example_home</code></p>
LANG or LC_ALL	<p>Function: Specifies the language and character set used by the operating system for messages and other output. Oracle tools that are programmed in Java, such as Oracle Universal Installer and Oracle Database Configuration Assistant, may also use this variable to determine the language of their user interface. Refer to the operating system documentation for more information.</p>
LD_OPTIONS	<p>Function: Specifies the default linker options. Refer to the <code>ld</code> man page for more information about this environment variable.</p>
LPDEST (Oracle Solaris only)	<p>Function: Specifies the name of the default printer.</p> <p>Syntax: <i>string</i></p> <p>Example: <code>docprinter</code></p>

Table 1-2 (Cont.) Environment Variables Used with Oracle Database

Variable	Definition
LD_LIBRARY_PATH	<p>Function: Environment variable to specify the path used to search for libraries on UNIX and Linux. The environment variable may have a different name on some operating systems, such as LIBPATH on IBM AIX on POWER Systems (64-Bit), and SHLIB_PATH on HP-UX.</p> <p>Syntax: Colon-separated list of directories: <i>directory1:directory2:directory3</i></p> <p>Example: /usr/dt/lib:\$ORACLE_HOME/lib</p>
PATH	<p>Function: Used by the shell to locate executable programs; must include the \$ORACLE_HOME/bin directory.</p> <p>Syntax: Colon-separated list of directories: <i>directory1:directory2:directory3</i></p> <p>Example: /bin:/usr/bin:/usr/local/bin:/usr/bin/X11:\$ORACLE_HOME/bin:\$HOME/bin:</p>
PRINTER	<p>Function: Specifies the name of the default printer.</p> <p>Syntax: <i>string</i></p> <p>Example: docprinter</p>
TEMP, TMP, and TMPDIR	<p>Function: Specifies the default directories for temporary files. If set, then the tools that create temporary files create them in one of these directories.</p> <p>Syntax: <i>directory_path</i></p> <p>Example: /u02/oracle/tmp</p>
USER (when using SSH to connect on HP-UX Itanium systems)	<p>Function: Specifies the name of the user logging in.</p> <p>Syntax: <i>string</i></p> <p>Example: oracle</p>

1.2.3 Setting a Common Environment

This section describes how to set a common operating system environment by using the `oraenv` or `coraenv` scripts, depending on the default shell:

- For the Bourne, Bash, or Korn shell, use the `oraenv` command.
- For the C shell, use the `coraenv` command.

oraenv and coraenv Script Files

The `oraenv` and `coraenv` scripts are created during installation. These scripts set environment variables based on the contents of the `oratab` file and provide:

- A central means of updating all user accounts with database changes
- A mechanism for switching between databases specified in the `oratab` file

You may find yourself frequently adding and removing databases from the development system or your users may be switching between several different Oracle Databases installed on the same system. You can use the `oraenv` or `coraenv` script to ensure that user accounts are updated and to switch between databases.

 **Note:**

Do not call the `oraenv` or `coraenv` script from the Oracle software owner (typically, `oracle`) user's shell startup script. Because these scripts prompt for values, they can prevent the `dbstart` script from starting a database automatically when the system starts.

The `oraenv` or `coraenv` script is usually called from the user's shell startup file (for example, `.profile` or `.login`). It sets the `ORACLE_SID` and `ORACLE_HOME` environment variables and includes the `$ORACLE_HOME/bin` directory in the `PATH` environment variable setting. When switching between databases, users can run the `oraenv` or `coraenv` script to set these environment variables.

 **Note:**

To run one of these scripts, use the appropriate command:

- `coraenv` script:

```
% source /usr/local/bin/coraenv
```
- `oraenv` script:

```
$ . /usr/local/bin/oraenv
```

Local bin Directory

The directory that contains the `oraenv`, `coraenv`, and `dbhome` scripts is called the local `bin` directory. All database users must have read access to this directory. Include the path of the local `bin` directory `PATH` environment variable setting for the users. When you run the `root.sh` script after installation, the script prompts you for the path of the local `bin` directory and automatically copies the `oraenv`, `coraenv`, and `dbhome` scripts to the directory that you specify. The default local `bin` directory is `/usr/local/bin`. If you do not run the `root.sh` script, then you can manually copy the `oraenv` or `coraenv` and `dbhome` scripts from the `$ORACLE_HOME/bin` directory to the local `bin` directory.

1.2.4 Setting the System Time Zone

The `TZ` environment variable sets the time zone. It enables you to adjust the clock for daylight saving time changes or different time zones.

 **See Also:**

- "ORA_TZFILE" in "Table 1-1"
- *Oracle Database Globalization Support Guide* and *Oracle Database Administrator's Guide* for more information about setting the database time zone

1.3 Initialization Parameters

These sections provide information about Oracle Database initialization parameters.

Topics:

- [ASM_DISKSTRING Initialization Parameter](#)
- [DISK_ASYNCH_IO Initialization Parameter \(HP-UX\)](#)
- [PROCESSOR_GROUP_NAME Initialization Parameter](#)
- [Managing Diagnostic Data](#)

1.3.1 ASM_DISKSTRING Initialization Parameter



Note:

Only Automatic Storage Management instances support the `ASM_DISKSTRING` initialization parameter.

The syntax for assigning a value to the `ASM_DISKSTRING` initialization parameter is as follows:

```
ASM_DISKSTRING = 'path1'[, 'path2', . . .]
```

In this syntax, *pathn* is the path to a raw device. You can use wildcard characters when specifying the path.

[Table 1-3](#) lists the platform-specific default values for the `ASM_DISKSTRING` initialization parameter.

Table 1-3 Default Values of the ASM_DISKSTRING Initialization Parameter

Platform	Default Search String
Oracle Solaris	/dev/rdisk/*
Linux	/dev/sd*
IBM AIX on POWER Systems (64-Bit)	/dev/rhdisk*
HP-UX	/dev/rdisk*

1.3.2 DISK_ASYNCH_IO Initialization Parameter (HP-UX)

The `DISK_ASYNCH_IO` initialization parameter determines whether the database files reside on raw disks or file systems. Asynchronous I/O is available only with Automatic Storage Management disk group which uses raw partitions as the storage option for database files. The `DISK_ASYNCH_IO` parameter can be set to `TRUE` or `FALSE` depending on where the files reside. By default, the value is set to `TRUE`.

 **Note:**

The `DISK_ASYNC_IO` parameter must be set to `FALSE` when the database files reside on file system. This parameter must be set to `TRUE` only when the database files reside on raw partitions.

1.3.3 PROCESSOR_GROUP_NAME Initialization Parameter

`PROCESSOR_GROUP_NAME` specifies the name of the processor group in which an instance is running. This parameter instructs Oracle databases to run only on processors which are a part of the specified operating system processor groups. For NUMA systems, all System Global Area (SGA) and Program Global Area (PGA) are allocated from the NUMA nodes associated with the CPUs in this processor group.

`PROCESSOR_GROUP_NAME` parameter is only supported on Linux x86-64 and Oracle Solaris 11 SRU 4 and later.

On Linux x86-64, the named subset of CPUs is created through a Linux feature called control groups (cgroups). Cgroups are introduced in Linux kernel version 2.6.24. It is created by specifying a name and a set of CPUs for the group. When a process is mapped to a cgroup, it uses only the CPUs associated with the cgroup.

On Oracle Solaris 11 SRU 4, the named subset of CPUs is created through a feature called resource pools. Each resource pool consists of a name and a set of CPUs. When a process is mapped to a resource pool, it uses the CPUs associated with the resource pool.

 **Note:**

Oracle recommends that the `PROCESSOR_GROUP_NAME` parameter is set only for databases using a dedicated connection broker. The `USE_DEDICATED_BROKER` initialization parameter is used to configure the dedicated connection brokers.

Related Topics

- *Oracle Database Reference*
- *Oracle Database Reference*

1.3.4 Managing Diagnostic Data

Diagnostic data includes the trace files, dumps, and core files to investigate, track, and resolve problems quickly and effectively.

Related Topics

- *Oracle Database Administrator's Guide*

2

Stopping and Starting Oracle Software

This chapter describes how to identify Oracle Database processes, and provides basic information about how to stop and restart them. It also describes how to set up automatic startup and shutdown of the Oracle Database. It contains the following sections:

- [Stopping and Starting Oracle Processes](#)
- [Automating Shutdown and Startup](#)

Note:

When using Oracle Restart, you can use Service Control Utility (SRVCTL), a command-line interface, to manage Oracle processes (database instance, listener, Oracle ASM instance). With SRVCTL, you can manage the Oracle Restart configuration, see the status of processes managed by Oracle Restart, and start or stop processes such as Oracle Database. SRVCTL is enhanced to support Oracle Clusterware, and single instance Oracle databases with Oracle Restart.

See Also:

Oracle Database Administrator's Guide and *Oracle Real Application Clusters Administration and Deployment Guide* for more information about SRVCTL commands.

2.1 Stopping and Starting Oracle Processes

This section describes how to stop and start Oracle processes. It contains the following topics:

- [Stopping and Starting Oracle Database and Oracle Automatic Storage Management Instances](#)
- [Stopping and Starting Oracle Restart](#)

2.1.1 Stopping and Starting Oracle Database and Oracle Automatic Storage Management Instances

This section describes how to stop and start Oracle Database and Oracle Automatic Storage Management instances and contains the following topics:

- [Stopping an Oracle Database or Oracle Automatic Storage Management Instance](#)

- [Restarting an Oracle Database or Oracle Automatic Storage Management Instance](#)

2.1.1.1 Stopping an Oracle Database or Oracle Automatic Storage Management Instance

Caution:

Do not stop an Oracle Automatic Storage Management instance until you have stopped all Oracle Database instances that use Oracle Automatic Storage Management instance to manage their storage.

To stop an Oracle Database or Oracle Automatic Storage Management instance:

1. Run the following commands to identify the SID and Oracle home directory for the instance that must be shut down:

On Oracle Solaris:

```
$ cat /var/opt/oracle/oratab
```

On other operating systems:

```
$ cat /etc/oratab
```

The `oratab` file contains lines similar to the following, which identify the `SID` and corresponding Oracle home directory for each database or Oracle Automatic Storage Management instance on the system:

```
$ORACLE_SID:$ORACLE_HOME:<N/Y>
```

Note:

Oracle recommends that you use the plus sign (+) as the first character in the `SID` of Oracle Automatic Storage Management instances.

2. Run the `oraenv` or `coraenv` script, depending on the default shell, to set the environment variables for the instance that must be shut down:

- Bourne, Bash, or Korn shell:

```
$ . /usr/local/bin/oraenv
```

- C shell:

```
% source /usr/local/bin/coraenv
```

When prompted, specify the `SID` for the instance.

3. Run the following commands to shut down the instance:

```
$ sqlplus
SQL> CONNECT SYS AS SYSDBA
Enter password: sys_password
SQL> SHUTDOWN NORMAL
```

After the instance shuts down, you can quit SQL*Plus.

2.1.1.2 Restarting an Oracle Database or Oracle Automatic Storage Management Instance

Caution:

If the database instance uses Oracle Automatic Storage Management for storage management, then you must start the Oracle Automatic Storage Management instance before you start the database instance.

To restart an Oracle Database or Oracle Automatic Storage Management instance:

1. Repeat steps 1 and 2, if required, to set the `ORACLE_SID` and `ORACLE_HOME` environment variables to identify the SID and Oracle home directory for the instance you want to start.
2. Run the following commands to start the instance:

```
$ sqlplus
SQL> CONNECT SYS AS SYSDBA
Enter password: sys_password
SQL> STARTUP
```

After the instance starts, you can exit from SQL*Plus.

2.1.2 Stopping and Starting Oracle Restart

To stop or start Oracle Restart, run the following command:

- **Start:** This option is used to start Oracle Restart

Syntax and Options:

```
crsctl start has
```

- **Stop:** This option is used to stop Oracle Restart

Syntax and Options:

```
crsctl stop has
```

See Also:

Oracle Database Administrator's Guide for more information about the `srvctl` commands

2.2 About Automating Database Shutdown and Startup

Oracle recommends that you configure the system to automatically start Oracle Database when the system starts, and to automatically shut it down when the system

shuts down. Automating database startup and shutdown guards against incorrect database shutdown.

To automate database startup and shutdown, use the `dbstart` and `dbshut` scripts, which are located in the `$ORACLE_HOME/bin` directory. The scripts refer to the same entries in the `oratab` file, which are applied on the same set of databases. You cannot, for example, have the `dbstart` script automatically start `sid1`, `sid2`, and `sid3`, and have the `dbshut` script shut down only `sid1`. However, you can specify that the `dbshut` script shuts down a set of databases while the `dbstart` script is not used at all. To do this, include a `dbshut` entry in the system shutdown file, but do not include the `dbstart` entry in the system startup files.



See Also:

The `init` command in the operating system documentation for more information about system startup and shutdown procedures

2.2.1 Automating Database Startup and Shutdown

To automate database startup and shutdown by using the `dbstart` and `dbshut` scripts:

1. Log in as the `root` user.
2. Edit the `oratab` file for the platform.

To open the file, use one of the following commands:

- On Oracle Solaris:

```
# vi /var/opt/oracle/oratab
```

- On IBM AIX on POWER Systems (64-Bit) and Linux:

```
# vi /etc/oratab
```

Database entries in the `oratab` file are displayed in the following format:

```
$ORACLE_SID:$ORACLE_HOME:<N|Y>
```

In this example, the values `Y` and `N` specify whether you want the scripts to start or shut down the database, respectively. For each database for which you want to automate shutdown and startup, first determine the instance identifier (SID) for that database, which is identified by the SID in the first field. Then, change the last field for each to `Y`.

You can set `dbstart` to autostart a single-instance database which uses an Automatic Storage Management installation auto-started by Oracle Clusterware. This is the default behavior for an Automatic Storage Management cluster. To do this, you must change the `oratab` entry of the database and the Automatic Storage Management installation to use a third field with the value `w` and `N`, respectively. These values specify that `dbstart` auto-starts the database only after the Automatic Storage Management instance is started.

 **Note:**

If you add new database instances to the system and automate the startup for them, then you must edit the entries for those instances in the oratab file.

3. Change directory to one of the following, depending on the operating system:

Platform	Initialization File Directory
Linux and Oracle Solaris	/etc/init.d
IBM AIX on POWER Systems (64-Bit)	/etc

4. Create a file called dbora, and copy the following lines into this file:

 **Note:**

Change the value of the ORA_HOME environment variable to specify the Oracle home directory for the installation. Change the value of the ORA_OWNER environment variable to the user name of the owner of the database installed in the Oracle home directory (typically, oracle).

```
#!/bin/sh
# description: Oracle auto start-stop script.
#
# Set ORA_HOME to be equivalent to the $ORACLE_HOME
# from which you wish to execute dbstart and dbshut;
#
# Set ORA_OWNER to the user id of the owner of the
# Oracle database in ORACLE_HOME.

ORA_HOME=<Type your ORACLE_HOME in full path here>
ORA_OWNER=<Type your Oracle account name here>

case "$1" in
'start')
    # Start the Oracle databases:
    # The following command assumes that the oracle login
    # will not prompt the user for any values
    # Remove "&" if you don't want startup as a background process.
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbstart $ORA_HOME" &
    touch /var/lock/subsys/dbora
    ;;
'stop')
    # Stop the Oracle databases:
    # The following command assumes that the oracle login
    # will not prompt the user for any values
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbshut $ORA_HOME" &
```

```
rm -f /var/lock/subsys/dbora
;;
esac
```

 **Note:**

This script can only stop Oracle Net listener for which a password has not been set. In addition, if the listener name is not the default name, LISTENER, then you must specify the listener name in the `stop` and `start` commands:

```
$ORACLE_HOME/bin/lsnrctl {start|stop} listener_name
```

5. Change the group of the `dbora` file to the OSDBA group (typically `dba`), and set the permissions to 750:

```
# chgrp dba dbora
# chmod 750 dbora
```

6. Create symbolic links to the `dbora` script in the appropriate run-level script directories, as follows:

Platform	Symbolic Links Commands
Oracle Solaris	<pre># ln -s /etc/init.d/dbora /etc/rc0.d/K01dbora # ln -s /etc/init.d/dbora /etc/rc3.d/S99dbora</pre>
Linux	<pre># ln -s /etc/init.d/dbora /etc/rc.d/rc0.d/K01dbora # ln -s /etc/init.d/dbora /etc/rc.d/rc3.d/S99dbora # ln -s /etc/init.d/dbora /etc/rc.d/rc5.d/S99dbora</pre>
IBM AIX on POWER Systems (64- Bit)	<pre># ln -s /etc/dbora /etc/rc.d/rc2.d/S99dbora # ln -s /etc/dbora /etc/rc.d/rc0.d/K01dbora</pre>

3

Configuring Oracle Database

This chapter describes how to configure Oracle Database for Oracle products. It contains the following sections:

- [Using Configuration Assistants](#)
- [Relinking Executables](#)

3.1 Using Configuration Assistants

Oracle Database software is supplied with configuration assistants that you can use to perform various database administration operations.

This section contains the following topics:

- [Using Oracle Net Configuration Assistant](#)
- [Using Oracle Database Upgrade Assistant](#)
- [Using Oracle Database Configuration Assistant](#)

3.1.1 Using Oracle Net Configuration Assistant

Oracle Net Configuration Assistant enables you to configure basic network components during installation, including listener names and protocol addresses, naming methods, net service names in `tnsnames.ora` file, and directory server usage.

After installation is complete, you can use Oracle Net Configuration Assistant to create a more detailed configuration by entering the following command:

```
$ $ORACLE_HOME/bin/netca
```

 **Note:**

When you use DBCA to create a database, it automatically updates the network configuration files to include information for the new database.

3.1.2 Using Oracle Database Upgrade Assistant

Oracle Database Upgrade Assistant (DBUA) interactively guides you through a database upgrade and configures the database for the new release. DBUA automates the upgrade by performing all tasks normally performed manually. DBUA makes recommendations for configuration options such as tablespaces and the online redo log.

To start DBUA, run the following command:

```
$ $ORACLE_HOME/bin/dbua
```

For information about the command-line options available with DBUA, use the `-help` or `-h` command-line arguments. For example:

```
$ $ORACLE_HOME/bin/dbua -help
```

Related Topics

- *Oracle Database Upgrade Guide*

3.1.3 Using Oracle Database Configuration Assistant

Oracle Database Configuration Assistant (DBCA) is a graphical user interface that assists you in creating and configuring a default or customized database. It also enables you to configure an existing database to add Oracle Database features, and create Oracle Automatic Storage Management disk groups. DBCA also generates a set of shell and SQL scripts that you can inspect, modify, and run at a later time to create a database.

To start DBCA, run the following command:

```
$ $ORACLE_HOME/bin/dbca
```

For information about the command-line options available with DBCA, use the `-help` or `-h` command-line arguments. For example:

```
$ $ORACLE_HOME/bin/dbca -help
```

3.2 Relinking Executables

You can relink the product executables manually by using the `relink` shell script located in the `$ORACLE_HOME/bin` directory. You must relink the product executables every time you apply an operating system patch or after an operating system upgrade.

Note:

Before relinking executables, you must shut down all the relinking executables which run in the Oracle home directory. In addition, shut down applications linked with Oracle shared libraries. The `relink` script takes `all` and `as_installed` as arguments. If you do not specify any argument, then the `all` argument is considered.

Depending on the products that have been installed in the Oracle home directory, the `relink` script relinks all Oracle product executables.

See Also:

Oracle Database Installation Guide for Linux for more information about how to use the `relink` script with Automatic Storage Manager

To relink the product executables, run the following command:

\$ relink

4

Administering SQL*Plus

This chapter describes how to administer SQL*Plus. It contains the following sections:

- [Administering Command-Line SQL*Plus](#)
- [Using Command-Line SQL*Plus](#)
- [SQL*Plus Restrictions](#)

Related Topics

- *SQL*Plus User's Guide and Reference*

4.1 Administering Command-Line SQL*Plus

This section describes how to administer command-line SQL*Plus. In the examples, SQL*Plus replaces the question mark (?) with the value of the `ORACLE_HOME` environment variable.

- [Using Setup Files](#)
- [Using the PRODUCT_USER_PROFILE Table](#)
- [Using Oracle Database Sample Schemas](#)
- [Installing and Removing SQL*Plus Command-Line Help](#)

4.1.1 Using Setup Files

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

Using the Site Profile File

The global site profile file is `$ORACLE_HOME/sqlplus/admin/glogin.sql`. If a site profile already exists at this location, then it is overwritten when you install SQL*Plus. If SQL*Plus is removed, then the site profile file is also removed.

Using the User Profile File

The user profile file is `login.sql`. SQL*Plus looks for this file in the current directory, and then in the directories specified by the `SQLPATH` environment variable. The value of this environment variable is a colon-separated list of directories. SQL*Plus searches these directories for the `login.sql` file in the order that they are listed in the `SQLPATH` environment variable.

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

**See Also:**

*SQL*Plus User's Guide and Reference* for more information about profile files

4.1.2 Using the PRODUCT_USER_PROFILE Table

Oracle Database provides the PRODUCT_USER_PROFILE table that you can use to disable the specified SQL and SQL*Plus commands. This table is automatically created when you choose an installation type that installs a preconfigured database.

**See Also:**

Oracle Database Installation Guide for more information about installation options

To re-create the PRODUCT_USER_PROFILE table, run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` script in the SYSTEM schema. For example, run the following commands, where `SYSTEM_PASSWORD` is the password of the SYSTEM user:

```
$ sqlplus
SQL> CONNECT SYSTEM
Enter password: system_password
SQL> @?/sqlplus/admin/pupbld.sql
```

You can also re-create the PRODUCT_USER_PROFILE table manually in the SYSTEM schema by using the `$ORACLE_HOME/bin/pupbld` shell script. This script prompts for the SYSTEM password. To run the pupbld script without interaction, set the `SYSTEM_PASS` environment variable to the SYSTEM user name and password.

4.1.3 Using Oracle Database Sample Schemas

When you install Oracle Database or use Oracle Database Configuration Assistant to create a database, you can choose to install Oracle Database Sample Schemas.

Related Topics

- [Oracle Database Sample Schemas](#)

4.1.4 Installing and Removing SQL*Plus Command-Line Help

This section describes how to install and remove the SQL*Plus command-line Help.

- [Installing SQL*Plus Command-Line Help](#)
- [Removing SQL*Plus Command-Line Help](#)

Related Topics

- [SQL*Plus User's Guide and Reference](#)

4.1.4.1 Installing SQL*Plus Command-Line Help

There are three ways to install the SQL*Plus command-line Help:

- Complete an installation that installs a preconfigured database.

When you install a preconfigured database as part of an installation, SQL*Plus automatically installs the SQL*Plus command-line Help in the SYSTEM schema.

- Install the command-line Help manually in the SYSTEM schema by using the `$ORACLE_HOME/bin/helpins` shell script.

The `helpins` script prompts for the SYSTEM password. To run this script without interaction, set the `SYSTEM_PASS` environment variable to the SYSTEM user name and password. For example:

- Bourne, Bash, or Korn shell:

```
$ SYSTEM_PASS=SYSTEM/system_password; export SYSTEM_PASS
```

- C shell:

```
% setenv SYSTEM_PASS SYSTEM/system_password
```

- Install the command-line Help manually in the SYSTEM schema by using the `$ORACLE_HOME/sqlplus/admin/help/helpbld.sql` script.

For example, run the following commands, where `system_password` is the password of the SYSTEM user:

```
$ sqlplus
SQL> CONNECT SYSTEM
Enter password: system_password
SQL> @?/sqlplus/admin/help/helpbld.sql ?/sqlplus/admin/help/helpus.sql
```

Note:

Both the `helpins` shell script and the `helpbld.sql` script drop existing command-line Help tables before creating new tables.

4.1.4.2 Removing SQL*Plus Command-Line Help

To manually drop the SQL*Plus command-line Help tables from the SYSTEM schema, run the `$ORACLE_HOME/sqlplus/admin/help/helpdrop.sql` script. To do this, run the following commands, where `system_password` is the password of the SYSTEM user:

```
$ sqlplus
SQL> CONNECT SYSTEM
Enter password: system_password
SQL> @?/sqlplus/admin/help/helpdrop.sql
```

4.2 Using Command-Line SQL*Plus

This section describes how to use command-line SQL*Plus. It contains the following topics:

- [Using a System Editor from SQL*Plus](#)

- [Running Operating System Commands from SQL*Plus](#)
- [Interrupting SQL*Plus](#)
- [Using the SPOOL Command](#)

4.2.1 Using a System Editor from SQL*Plus

If you run an `ED` or `EDIT` command at the SQL*Plus prompt, then the system starts an operating system editor, such as `ed`, `emacs`, `ned`, or `vi`. However, the `PATH` environment variable must include the directory where the editor executable is located.

The `HISTORY` command enables users to run, edit, or delete previously used SQL*Plus, SQL, or PL/SQL commands from the history list in the current session. You can enable or disable the history of commands issued in the current SQL*Plus session.

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 should start by defining the SQL*Plus `_EDITOR` variable. You can define this variable in the `glogin.sql` site profile or the `login.sql` user profile. Alternatively, you can define it during the SQL*Plus session. For example, to set the default editor to `vi`, run the following command:

```
SQL> DEFINE _EDITOR=vi
```

If you do not set the `_EDITOR` variable, then the value of either the `EDITOR` or the `VISUAL` environment variable is used. If both environment variables are set, then the value of the `EDITOR` variable is used. If `_EDITOR`, `EDITOR`, and `VISUAL` are not specified, then the default editor is `vi`. The defined editor is used in the `edit` option of the `HISTORY` command. Consider the following SQL statements:

```
SQL> hist
1 select * from dual;
2 desc dual
```

To open the first entry in the above SQL statements in the `vi` editor, use the following command:

```
SQL> hist 1 edit
```

When 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 specify a different file name. For example:

```
SQL> SET EDITFILE /tmp/myfile.sql
```

SQL*Plus does not delete the temporary file.

4.2.2 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 subshell. The `SHELL` environment variable sets the shell used to run operating system commands.

The default shell is the Bourne shell. If the shell cannot be run, then SQL*Plus displays an error message.

To return to SQL*Plus, run the `exit` command or press **Ctrl+D**.

For example, to run a single command, use the following command syntax:

```
SQL> ! command
```

In this example, *command* represents the operating system command that you want to run.

To run multiple operating system commands from SQL*Plus, run the `HOST` or `!` command. Press **Enter** to return to the operating system prompt.

4.2.3 Interrupting SQL*Plus

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

4.2.4 Using the SPOOL Command

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

```
SQL> SPOOL query.txt
```

4.3 SQL*Plus Restrictions

This section describes the following SQL*Plus restrictions:

- [Resizing Windows](#)
- [Return Codes](#)
- [Hiding the Password](#)

4.3.1 Resizing Windows

The default values for the SQL*Plus `LINESIZE` and `PAGESIZE` system variables do not automatically adjust for the window size. If the window size is changed, then you must set the `LINESIZE` and `PAGESIZE` system variables.

4.3.2 Return Codes

Operating system 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.

4.3.3 Hiding the Password

If you pass the password on the command line or set the `SYSTEM_PASS` environment variable to the user name and password of the `SYSTEM` user, then the output of the `ps` command may display this information. To prevent unauthorized access, enter the `SYSTEM` password only when prompted by SQL*Plus.

To automatically run a script, consider using an authentication method that does not require you to store a password. For example, externally authenticated logins to Oracle Database. If you have a low-security environment, then you must consider using the operating system pipes in script files to pass a password to SQL*Plus. For example:

```
$ echo system_password | sqlplus SYSTEM @MYSCRIPT
```

Alternatively, run the following commands:

```
$ sqlplus <<EOF  
SYSTEM/system_password  
SELECT ...  
EXIT  
EOF
```

In the preceding examples, *system_password* is the password of the SYSTEM user.

5

Configuring Oracle Net Services

This chapter describes how to configure Oracle Net Services. It contains the following sections:

- [Locating Oracle Net Services Configuration Files](#)
- [Using Adapters Utility](#)
- [Using Oracle Protocol Support](#)
- [Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer](#)

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.1 Locating Oracle Net Services Configuration Files

Oracle Net Services configuration files are typically, but not always, located in the `$ORACLE_HOME/network/admin` directory. Depending on the type of file, Oracle Net uses a different search order to locate the file.

The search order for the `sqlnet.ora` and `ldap.ora` files is as follows:

1. The directory specified by the `TNS_ADMIN` environment variable, if this environment variable is set
2. The `$ORACLE_HOME/network/admin` directory

The search order for the `cman.ora`, `listener.ora`, and `tnsnames.ora` files is as follows:

1. The directory specified by the `TNS_ADMIN` environment variable, if this environment variable is set
2. One of the following directories:
 - On Oracle Solaris:
`/var/opt/oracle`
 - On other platforms:
`/etc`
3. The `$ORACLE_HOME/network/admin` directory

For some system-level configuration files, users may have a corresponding user-level configuration file stored in their home directory. The settings in the user-level file override the settings in the system-level file. The following table lists the system-level configuration files and the corresponding user-level configuration files:

System-Level Configuration File	User-Level Configuration File
<code>sqlnet.ora</code>	<code>\$HOME/.sqlnet.ora</code>

System-Level Configuration File	User-Level Configuration File
<code>tnsnames.ora</code>	<code>\$HOME/.tnsnames.ora</code>

Example 5-1 Sample Configuration Files

The `$ORACLE_HOME/network/admin/samples` directory contains samples of the `cman.ora`, `listener.ora`, `sqlnet.ora`, and `tnsnames.ora` configuration files.

Note:

- The `cman.ora` file is installed only if you select Connection Manager as part of a custom option during a client custom installation.
- In a Read-only Oracle Home, the configuration files are located in the `$ORACLE_BASE_HOME/network/admin` directory.

5.2 Using Adapters Utility

The `adapters` utility displays the transport protocols, naming methods, and Oracle Advanced Security options that Oracle Database supports on the system.

See Also:

Oracle Database Net Services Administrator's Guide for more information about the `adapters` utility

5.3 Using Oracle Protocol Support

Oracle protocol support is a component of Oracle Net. It includes the following:

- [IPC Protocol Support](#)
- [TCP/IP Protocol Support](#)
- [TCP/IP with Secure Sockets Layer Protocol Support](#)

Each of the IPC, TCP/IP, and TCP/IP with Secure Sockets Layer protocol support have an address specification that is used in the Oracle Net Services configuration files and in the `DISPATCHER` initialization parameter. The following sections describe the address specifications for each of the protocol supports.

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.3.1 IPC Protocol Support

The IPC protocol support can be used only when the client program and Oracle Database are installed on the same system. This protocol support requires a listener. It is installed and linked to all client tools and the `oracle` executable.

The IPC protocol support requires an address specification in the following format:

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

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol to be used. The value is IPC. It is not case-sensitive.
KEY	Any name that is different from any other name used for an IPC KEY on the same system.

The following is a sample IPC protocol address:

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

5.3.2 TCP/IP Protocol Support

TCP/IP is the standard communication protocol used for client/server communication over a network. The TCP/IP protocol support enables communication between client programs and Oracle Database, whether they are installed on the same or different systems. If the TCP/IP protocol is installed on the system, then the TCP/IP protocol support is installed and linked to all client tools and to the `oracle` executable.

The TCP/IP protocol support requires an address specification in the following format:

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

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol support 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. Specify the port as either a number or the alias name mapped to the port in the <code>/etc/services</code> file. Oracle recommends a value of 1521.

The following is a sample TCP/IP protocol address:

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

5.3.3 TCP/IP with Secure Sockets Layer Protocol Support

The TCP/IP with Secure Sockets Layer protocol support enables an Oracle application on a client to communicate with remote Oracle Database instances through TCP/IP and Secure Sockets Layer.

The TCP/IP with Secure Sockets Layer protocol support requires an address specification in the following format:

```
(ADDRESS = (PROTOCOL=TCPS)(HOST=hostname)(PORT=port))
```

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol to be used. The value is TCPS. It is not case-sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP with Secure Sockets Layer port. Specify the port as either a number or the alias name mapped to the port in the <code>/etc/services</code> file. Oracle recommends a value of 2484.

The following is a sample TCP/IP with Secure Sockets Layer protocol address:

```
(ADDRESS= (PROTOCOL=TCPS)(HOST=MADRID)(PORT=2484))
```

5.4 Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer

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

```
oraclelistener 1521/tcp
```

In this example, `oraclelistener` is the name of the listener as defined in the `listener.ora` file. Reserve multiple ports if you intend to start multiple listeners.

If you intend to use Secure Sockets Layer, then you should define a port for TCP/IP with Secure Sockets Layer in the `/etc/services` file. Oracle recommends a value of 2484. For example:

```
oraclelistenerssl 2484/tcps
```

In this example, `oraclelistenerssl` is the name of the listener as defined in the `listener.ora` file. Reserve multiple ports if you intend to start multiple listeners.

6

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes how to use Oracle precompilers and the Oracle Call Interface

Topics:

- [Overview of Oracle Precompilers](#)
- [Bit-Length Support for Client Applications](#)
- [Pro*C/C++ Precompiler](#)
- [Pro*COBOL Precompiler](#)
- [Pro*FORTRAN Precompiler](#)
- [SQL*Module for ADA](#)
- [OCI and OCCI](#)
- [Running Oracle JDBC/OCI Programs with a 64-Bit Driver](#)
- [Custom Make Files](#)
- [Correcting Undefined Symbols](#)
- [Multithreaded Applications](#)
- [Using Signal Handlers](#)
- [XA Functionality](#)

6.1 Overview of Oracle Precompilers

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

This section contains the following topics:

- [Precompiler Configuration Files](#)
- [Relinking Precompiler Executables](#)
- [Issues Common to All Precompilers](#)
- [Static and Dynamic Linking](#)
- [Client Shared and Static Libraries](#)

 **Note:**

- `ORACLE_HOME` in this section refers to `ORACLE_HOME` that is created while installing Oracle Database Client 19c by using the Administrator Install type.
- If you have configured a read-only Oracle home, then you must copy the demo directories from `ORACLE_HOME` to `ORACLE_BASE_HOME`. Refer to *Oracle Database Installation Guide for Linux* for more information.

6.1.1 Precompiler Configuration Files

Configuration files for the Oracle precompilers are located in the `$ORACLE_HOME/precomp/admin` directory.

The following table lists the names of the configuration files for each precompiler.

Table 6-1 System Configuration Files for Oracle Precompilers

Product	Configuration File
Pro*C/C++	<code>pcscfg.cfg</code>
Pro*COBOL	<code>pcbcfg.cfg</code>
Pro*FORTRAN (IBM AIX on POWER Systems (64-Bit), HP-UX, and Oracle Solaris)	<code>pccfor.cfg</code>
Object Type Translator	<code>ottcfg.cfg</code>
SQL*Module for Ada (IBM AIX on POWER Systems (64-Bit))	<code>pmscfg.cfg</code>

6.1.2 Relinking Precompiler Executables

Use the `$ORACLE_HOME/precomp/lib/ins_precomp.mk` make file to relink all precompiler executables. To manually relink a particular precompiler executable, enter the following command:

```
$ make -f ins_precomp.mk relink exename = executable_name
```

This command creates the new executable in the `$ORACLE_HOME/precomp/lib` directory, and then moves it to the `$ORACLE_HOME/bin` directory.

In the preceding example, replace `executable` with one of the product executables listed in [Table 6-2](#).

The following table lists the executables for Oracle Precompilers:

Table 6-2 Executables for Oracle Precompilers

Product	Executable
Pro*FORTRAN 32-bit (Oracle Solaris, HP-UX and IBM AIX on POWER Systems (64-Bit))	<code>profor</code>

Table 6-2 (Cont.) Executables for Oracle Precompilers

Product	Executable
Pro*COBOL 32-bit (Oracle Solaris, HP-UX, and IBM AIX on POWER Systems (64-Bit))	procob
Pro*COBOL (Oracle Solaris, HP-UX, and IBM AIX on POWER Systems (64-Bit))	procob or rtsora
Pro*C/C++ 32 bit (HP-UX)	proc
Pro*FORTRAN (HP-UX)	profor
SQL*Module for Ada (IBM AIX on POWER Systems (64-Bit))	modada

6.1.3 Issues Common to All Precompilers

The following issues are common to all precompilers:

- **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, then verify that the case of the function or subprogram name in the option file matches the case used in the IAPXTB table.
- **Vendor Debugger Programs**
Precompilers and vendor-supplied debuggers can be incompatible. Oracle does not guarantee that a program run using a debugger performs the same way when it is run without the debugger.
- **Value of IRECLLEN and ORECLLEN parameters**
The IRECLLEN and ORECLLEN parameters do not have maximum values.

6.1.4 Static and Dynamic Linking

You can statically or dynamically link Oracle libraries with precompiler and OCI or OCCl applications. With static linking, the libraries and objects of the whole application are linked 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. The benefits of dynamic linking are:

- **Reduced disk space requirements:** Multiple applications or calls to the same application can use the same dynamic libraries.
- **Reduced main memory requirements:** The same dynamic library image is loaded into the main memory only once, and it can be shared by multiple applications.

6.1.5 Client Shared and Static Libraries

The client shared and static libraries are located in `$ORACLE_HOME/lib`. If you use the Oracle-provided `demo_product.mk` file to link an application, then the client shared library is linked by default.

If the shared library path environment variable setting does not include the directory that contains the client shared library, then you may see an error message similar to one of the following lines when starting an executable:

```
Cannot load library libclntsh.a
cannot open shared library: ../libclntsh.sl.10.1
libclntsh.so.10.1: can't open file: errno=2
can't open library: ../libclntsh.dylib.10.1
Cannot map libclntsh.so
```

To avoid this error, set the shared library path environment variable to specify the appropriate directory. The following table shows sample settings for this environment variable name. If the platform supports both 32-bit and 64-bit applications, then ensure that you specify the correct directory, depending on the application that you want to run.

Platform	Environment Variable	Sample Setting
Oracle Solaris (32-bit and 64-bit applications) and Linux	LD_LIBRARY_PATH	<code>\$ORACLE_HOME/lib</code>
IBM AIX on POWER Systems (32-bit and 64-bit applications)	LIBPATH	<code>\$ORACLE_HOME/lib</code>
HP-UX (32-bit applications)	SHLIB_PATH	<code>\$ORACLE_HOME/lib</code>
HP-UX (64-bit applications)	LD_LIBRARY_PATH	<code>\$ORACLE_HOME/lib</code>

The client shared library is created automatically during installation. If you must re-create it, then complete the following procedure:

1. Quit all client applications that use the client shared library, including all Oracle client applications such as SQL*Plus and Oracle Recovery Manager.
2. Log in as the `oracle` user, and run the following command:

```
$ $ORACLE_HOME/bin/genclntsh
```

Non-threaded Client Shared Library



Note:

The information in this section applies to HP-UX systems.

On HP-UX, you can use a non-threaded client shared library. However, you cannot use this library with any OCI application that uses or has a dependency on threads.

To use this library for applications that do not use threads, run the following command to build the OCI application for 32 and 64-bit:

```
$ make -f demo_rdbms.mk build_nopthread EXE=oci02 OBJS=oci02.o
```

6.1.6 Generating the Client Static Library

To link applications to the client static library, you must generate it first.

The client static library (`libclntst12.a`) is not generated during an Oracle Database installation. To link the applications to the client static library, you must first generate it:

1. Switch the user to the Oracle installation owner (`oracle`).
2. Set the `ORACLE_HOME` environment variable to specify the Oracle home directory used by the Oracle Database installation. For example:

- Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=/u01/app/oracle/product/19.0.0/dbhome_1
$ export ORACLE_HOME
```

- C shell:

```
% setenv ORACLE_HOME /u01/app/oracle/product/19.0.0/dbhome_1
```

3. Enter the following command:

```
$ $ORACLE_HOME/bin/genclntst
```

6.2 Bit-Length Support for Client Applications

The client application type (32-bit or 64-bit) is supported on the following platforms:

- Oracle Solaris
- Linux x86-64
- IBM: Linux on System z
- IBM AIX on POWER Systems (64-Bit)
- HP-UX Itanium

The following table lists the 32-bit and 64-bit client shared libraries:

Platform	32-Bit Client Shared Library	64-Bit Client Shared Library
Oracle Solaris, Linux x86-64, and IBM: Linux on System z	<code>\$ORACLE_HOME/lib/libclntsh.so</code>	<code>\$ORACLE_HOME/lib/libclntsh.so</code>
IBM AIX on POWER Systems (64-Bit)	<code>\$ORACLE_HOME/lib/libclntsh.a</code> <code>\$ORACLE_HOME/lib/libclntsh.so</code>	<code>\$ORACLE_HOME/lib/libclntsh.a</code> <code>\$ORACLE_HOME/lib/libclntsh.so</code>
HP-UX Itanium	<code>\$ORACLE_HOME/lib/libclntsh.sl</code>	<code>\$ORACLE_HOME/lib/libclntsh.sl</code>

To implement a mixed word-size installation:

1. Run the following command to generate the 32-bit and 64-bit client shared libraries:

```
$ $ORACLE_HOME/bin/genclntsh
```
2. Include the paths of the required 32-bit and 64-bit client shared libraries in one of the following environment variables, depending on the platform:

Platform	Environment Variable
Oracle Solaris, Linux x86-64, IBM: Linux on System z, and HP-UX	LD_LIBRARY_PATH
IBM AIX on POWER Systems (64-Bit)	LIBPATH
HP-UX (32-bit client applications)	SHLIB_PATH

Building 32-Bit Pro*C and OCI Customer Applications

If the operating system supports both 32-bit and 64-bit Pro*C and Oracle Call Interface (OCI) customer applications, then you can find more information about building 32-bit Pro*C and OCI applications in the following files:

For Information About. . . Refer to the Following Make Files. . .

Building 32-bit Pro*C applications	<code>\$ORACLE_HOME/precomp/demo/proc/demo_proc32.mk</code>
Building 32-bit OCI applications	<code>\$ORACLE_HOME/rdbms/demo/demo_rdbms32.mk</code>

6.3 Pro*C/C++ Precompiler

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

See Also:

- *Oracle Database Installation Guide* for information about supported compiler versions
- *Pro*C/C++ Programmer's Guide* for information about the Pro*C/C++ precompiler and interface features

This section contains the following topics:

- [Pro*C/C++ Demonstration Programs](#)
- [Pro*C/C++ User Programs](#)

6.3.1 Pro*C/C++ Demonstration Programs

Demonstration programs are provided to show the features of the Pro*C/C++ precompiler. There are three types of demonstration programs: C, C++, and Object programs. All demonstration programs are located in the `$ORACLE_HOME/precomp/`

demo/proc directory. By default, all programs are dynamically linked with the client shared library.

To run the demonstration programs, the programs require the demonstration tables created by the demobld.sql script to exist in the JONES schema with a password.

The demobld.sql script is located in the \$ORACLE_HOME/sqlplus/demo/ directory in a read-write Oracle home and in the \$(orabasehome)/sqlplus/demo/ directory in a read-only Oracle home.

 **Note:**

You must unlock the JONES account and set the password before creating the demonstrations.

Use the demo_proc.mk make file to create the demonstration programs. The demo_proc.mk file is located in the \$ORACLE_HOME/precomp/demo/proc/ directory in a read-write Oracle home and in the \$(orabasehome)/precomp/demo/proc/ in a read-only Oracle home. For example, to precompile, compile, and link the sample1 demonstration program, run the following command:

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

 **Note:**

On IBM AIX on POWER Systems (64-Bit), to ensure that the demonstration programs compile correctly, include the -r option of the make command in the following examples. For example:

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

To create all the C demonstration programs for Pro*C/C++, run the following command:

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

To create all the C++ demonstration programs for Pro*C/C++, run the following command:

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

To create all the Object demonstration programs for Pro*C/C++, run the following command:

```
$ 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, then a message prompting you to run it is displayed.

To build a demonstration program and run the corresponding SQL script, include the make macro argument RUNSQL=run at the command line. For example, to create the

sample9 demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample9.sql` script, run the following command:

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

To create all the Object demonstration programs and run all the required SQL scripts, run the following command:

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

6.3.2 Pro*C/C++ User Programs

You can use the `$ORACLE_HOME/precomp/demo/proc/demo_proc.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_proc32.mk` make file to build 32-bit user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*C/C++:

Platform	64-bit Make File	32-Bit Make File
Oracle Solaris, Linux x86-64, IBM: Linux on System z, IBM AIX on POWER Systems (64-Bit), and HP-UX	<code>demo_proc.mk</code>	<code>demo_proc32.mk</code>

See Also:

The make file for more information about creating user programs

Note:

On IBM AIX on POWER Systems (64-Bit), to ensure that the programs compile correctly, specify the `-r` option for the `make` command used in the following examples.

To create a program by using the `demo_proc.mk` make file, run a command similar to the following:

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

In this example:

- `target` is the make file target that you want to use
- `objfilen` is the object file to link the program
- `exename` is the executable program

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

- For C source dynamically linked with the client shared library, run the following command:

```
$ make -f demo_proc.mk build OBJS=myprog.o EXE=myprog
```
- For C source statically linked with the client shared library, run the following command:

```
$ make -f demo_proc.mk build_static OBJS=myprog.o EXE=myprog
```
- For C++ source dynamically linked with the client shared library, run the following command:

```
$ make -f demo_proc.mk cppbuild OBJS=myprog.o EXE=myprog
```
- For C++ source statically linked with the client shared library, run the following command:

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

6.4 Pro*COBOL Precompiler

Table 6-3 shows the naming conventions for the Pro*COBOL precompiler.

Table 6-3 Pro*COBOL Naming Conventions

Item	Naming Convention
Executable	procob
Demonstration directory	procob2
Make file	demo_procob.mk or demo_procob_32.mk

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

This section contains the following topics:

- [Pro*COBOL Environment Variables](#)
- [Pro*COBOL Oracle Runtime System](#)
- [Pro*COBOL Demonstration Programs](#)
- [Pro*COBOL User Programs](#)
- [FORMAT Precompiler Option](#)

6.4.1 Pro*COBOL Environment Variables

This section describes the environment variables required by Pro*COBOL:

- [Micro Focus Server Express COBOL Compiler](#)
- [Acucorp ACUCOBOL-GT COBOL Compiler](#)

6.4.1.1 Micro Focus Server Express COBOL Compiler

To use the Micro Focus Server Express COBOL compiler, you must set the `COBDIR` and `PATH` environment variables and the shared library path environment variable.

See Also:

The "[Client Shared and Static Libraries](#)" section for information about the shared library path environment variable

COBDIR

Set the `COBDIR` environment variable to the directory where the compiler is installed. For example, if the compiler is installed in the `/opt/lib/cobol` directory, then run the following command:

- Bourne, Bash, or Korn shell:

```
$ COBDIR=/opt/lib/cobol
$ export COBDIR
```
- C shell:

```
% setenv COBDIR /opt/lib/cobol
```

PATH

Set the `PATH` environment variable to include the `COBDIR/bin` directory:

- Bourne, Bash, or Korn shell:

```
$ PATH=$COBDIR/bin:$PATH
$ export PATH
```
- C shell:

```
% setenv PATH ${COBDIR}/bin:${PATH}
```

Shared Library Path

Set the `LIBPATH`, `LD_LIBRARY_PATH`, or `SHLIB_PATH` environment variable to the directory where the compiler library is installed. For example, if the platform uses the `LD_LIBRARY_PATH` environment variable and the compiler library is installed in the `COBDIR/coblib` directory, then run the following command:

- Bourne, Bash, or Korn shell:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$COBDIR/coblib
$ export LD_LIBRARY_PATH
```
- C shell:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:$COBDIR/coblib
```

6.4.1.2 Acucorp ACUCOBOL-GT COBOL Compiler

To use the Acucorp ACUCOBOL-GT COBOL compiler, you must set the `A_TERMCPAP`, `A_TERM`, `PATH`, and `LD_LIBRARY_PATH` environment variables. If the `LD_LIBRARY_PATH` environment variable setting does not include the correct directory, then an error message similar to the following is displayed when you compile or run a program:

```
runcbl: error while loading shared libraries: libcintsh.so:  
cannot open shared object file: No such file or directory
```

A_TERMCPAP and A_TERM

Set the `A_TERMCPAP` environment variable to specify the location of the `a_termcap` file and set the `A_TERM` environment variable to specify a supported terminal from that file. For example:

- Bourne, Bash, or Korn shell:

```
$ A_TERMCPAP=/opt/COBOL/etc/a_termcap  
$ A_TERM=vt100  
$ export A_TERMCPAP A_TERM
```

- C shell:

```
% setenv A_TERMCPAP /opt/COBOL/etc/a_termcap  
% setenv A_TERM vt100
```

PATH

Set the `PATH` environment variable to include the `/opt/COBOL/bin` directory:

- Bourne, Bash, or Korn shell:

```
$ PATH=/opt/COBOL/bin:$PATH  
$ export PATH
```

- C shell:

```
% setenv PATH opt/COBOL/bin:${PATH}
```

LD_LIBRARY_PATH

 **Note:**

On IBM AIX on POWER Systems (64-Bit), the `LIBPATH` variable is the `LD_LIBRARY_PATH` variable equivalent. You must use the `LIBPATH` variable on IBM AIX on POWER Systems (64-Bit) instead of the `LD_LIBRARY_PATH` variable in the following commands.

Set the `LD_LIBRARY_PATH` environment variable to the directory where the compiler library is installed. For example, if the compiler library is installed in the `/opt/COBOL/lib` directory, then run the following command:

- Bourne, Bash, or Korn shell:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/COBOL/lib  
$ export LD_LIBRARY_PATH
```

- C shell:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/COBOL/lib
```

6.4.2 Pro*COBOL Oracle Runtime System

Oracle provides its own complete run-time system, called `rtsora`, to run dynamically loadable Pro*COBOL programs. Use the `rtsora` run-time system instead of the `cobrun` run-time system to run dynamically loadable Pro*COBOL programs. If you attempt to run a Pro*COBOL program with `cobrun`, then an error message similar to the following is displayed:

```
$ 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
```

6.4.3 Pro*COBOL Demonstration Programs

Demonstration programs are provided to show the features of the Pro*COBOL precompiler. The demonstration programs are located in the `$ORACLE_HOME/precomp/demo/procob2` directory. By default, all programs are dynamically linked with the client shared library.

To run the demonstration programs, the programs require the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script to exist in the JONES schema with a password.



Note:

You must unlock the JONES account and set the password before creating the demonstrations.

Use the following make file to create the demonstration programs:

```
$ORACLE_HOME/precomp/demo/procob2/demo_procob.mk
```

To precompile, compile, and link the `sample1` demonstration program for Pro*COBOL, run the following command:

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

To create the Pro*COBOL demonstration programs, run the following command:

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

To create and run a dynamically loadable `sample1.gnt` program to be used with the `rtsora` run-time system, run the following command:

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

Some demonstration programs require you to run a SQL script, which is located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, then a message requesting you to run it is displayed.

To build a demonstration program and run the corresponding SQL script, include the make macro argument `RUNSQL=run` in the command. For example, to create the `sample9` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample9.sql` script, run the following command:

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

To create the Pro*COBOL demonstration programs and run all required SQL scripts, run the following command:

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

6.4.4 Pro*COBOL User Programs

You can use the `$ORACLE_HOME/precomp/demo/procob2/demo_procob.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_procob_32.mk` make file to build 32-bit user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*COBOL:

Platform	64-bit Make File	32-Bit Make File
Oracle Solaris, Linux x86-64, IBM: Linux on System z, IBM AIX on POWER Systems (64-Bit), and HP-UX	<code>demo_procob.mk</code>	<code>demo_procob_32.mk</code>

See Also:

The make file for more information about creating user programs

To create a program using the `demo_procob.mk` make file, run a command similar to the following:

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

In this example:

- `target` is the make file target that you want to use
- `cobfilen` is the COBOL source file for the program
- `exename` is the executable program

For example, to create the program `myprog`, run one of the following commands, depending on the source and type of executable that you want to create:

- For COBOL source, dynamically linked with the client shared library, run the following command:

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

- For COBOL source, statically linked, run the following command:

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

- For COBOL source, dynamically loadable for use with `rtsofa`, run the following command:

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

6.4.5 FORMAT Precompiler Option

The `FORMAT` precompiler option specifies the format of input lines for COBOL. If you specify the default value `ANSI`, then 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 the value `TERMINAL`, then columns 1 to 6 are dropped, making column 7 the left most column.

6.5 Pro*FORTRAN Precompiler

Before you use the Pro*FORTRAN precompiler, verify that the correct version of the compiler is installed. This section contains the following topics:

- [Pro*FORTRAN Demonstration Programs](#)
- [Pro*FORTRAN User Programs](#)

See Also:

- *Oracle Database Installation Guide* for information about supported compiler versions
- *Pro*FORTRAN Supplement to the Oracle Precompilers Guide* for information about the Pro*FORTRAN precompiler and interface features

6.5.1 Pro*FORTRAN Demonstration Programs

Demonstration programs are provided to show the features of the Pro*FORTRAN precompiler. All demonstration programs are located in the `$ORACLE_HOME/precomp/demo/profor` directory. By default, all programs are dynamically linked with the client shared library.

To run the demonstration programs, the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the JONES schema with a password.

Note:

You must unlock the JONES account and set the password before creating the demonstrations.

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

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

To create the Pro*FORTRAN demonstration programs, run the following command:

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

Some demonstration programs require you to run a SQL script that is located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, then a message prompting you to run it is displayed.

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, run the following command:

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

To create the Pro*FORTRAN demonstration programs and run all the required SQL scripts, run the following command:

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

6.5.2 Pro*FORTRAN User Programs

You can use the `$ORACLE_HOME/precomp/demo/profor/demo_profor.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_profor_32.mk` make file to build 32-bit user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*FORTRAN:

Platform	64-bit Make File	32-Bit Make File
Oracle Solaris, IBM AIX on POWER Systems (64-Bit), and HP-UX	<code>demo_profor.mk</code>	<code>demo_profor_32.mk</code>

See Also:

The make file for more information about creating user programs

To create a program using the `demo_proc.mk` make file, run a command similar to the following:

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

In this example:

- `target` is the make file target that you want to use
- `forfilen` is the FORTRAN source for the program
- `exename` is the executable program

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

- For an executable dynamically linked with the client shared library, run the following command:

```
$ make -f demo_profor.mk build FORS=myprog.f EXE=myprog
```
- For an executable statically linked with the client shared library, run the following command:

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

6.6 SQL*Module for ADA



Note:

The information in this section applies to the IBM AIX on POWER Systems (64-Bit) platform.

Before using SQL*Module for Ada, verify that the correct version of the compiler is installed.



See Also:

- *Oracle Database Installation Guide* for information about required compiler versions
- *Oracle SQL*Module for Ada Programmer's Guide* for information about SQL*Module for Ada

This section contains the following topics:

- [SQL*Module for Ada Demonstration Programs](#)
- [SQL*Module for Ada User Programs](#)

6.6.1 SQL*Module for Ada Demonstration Programs

Demonstration programs are provided to show the features of SQL*Module for Ada. All demonstration programs are located in the `$ORACLE_HOME/precomp/demo/modada` directory. By default, all programs are dynamically linked with the client shared library.

To run the `chl_drv` demonstration program, the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the `JONES` schema with a password.

 **Note:**

You must unlock the JONES account and set the password before creating the demonstrations.

The `demcalsp` and `demohost` demonstration programs require that the sample college database exists in the `MODTEST` schema. You can use the appropriate `make` command to create the `MODTEST` schema and load the sample college database.

Run the following command to create the SQL*Module for Ada demonstration programs, run the necessary SQL scripts to create the `MODTEST` user, and create the sample college database:

```
$ make -f demo_modada.mk all RUNSQL=run
```

To create a single demonstration program (`demohost`) and run the necessary SQL scripts to create the `MODTEST` user, and create the sample college database, run the following command:

```
$ make -f demo_modada.mk makeuser loaddb demohost RUNSQL=run
```

To create the SQL*Module for Ada demonstration programs, without re-creating the sample college database, run the following command:

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

To create a single demonstration program (`demohost`), without re-creating the sample college database, run the following command:

```
$ make -f demo_modada.mk demohost
```

To run the programs, you must define an Oracle Net connect string or alias named `INST1_ALIAS` that is used to connect to the database where the appropriate tables exist.

6.6.2 SQL*Module for Ada User Programs

You can use the `$(ORACLE_HOME)/precomp/demo/modada/demo_modada.mk` `make` file to create user programs. To create a user program with the `demo_modada.mk` `make` file, run a command similar to the following:

```
$ make -f demo_modada.mk ada OBJS="module1 module2 ..." \  
EXE=exename MODARGS=SQL_Module_arguments
```

In this example:

- `modulen` is a compiled Ada object
- `exename` is the executable program
- `SQL_Module_arguments` are the command-line arguments to be passed to the SQL*Module

 **See Also:**

[Oracle SQL*Module for Ada Programmer's Guide](#) for information about SQL*Module for Ada

6.7 OCI and OCCI

Before you use the Oracle Call Interface (OCI) or Oracle C++ Call Interface (OCCI), verify that the correct version of C or C++ is installed.

 **See Also:**

- [Oracle Database Installation Guide](#) for information about supported compiler versions
- [Oracle Call Interface Programmer's Guide](#) or [Oracle C++ Call Interface Programmer's Guide](#) for information about OCI and OCCI

This section contains the following topics:

- [OCI and OCCI Demonstration Programs](#)
- [OCI and OCCI User Programs](#)

6.7.1 OCI and OCCI Demonstration Programs

Demonstration programs that show the features of OCI and OCCI are provided with the Oracle Database 19c Examples software. There are two types of demonstration programs: C and C++. All demonstration programs are located in the `$ORACLE_HOME/rdbms/demo` directory. By default, all programs are dynamically linked with the client shared library.

To run the demonstration programs, the programs require the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script to exist in the JONES schema with a password. Some demonstration programs require specific `.sql` files to be run, as mentioned in the demonstration source files. OCCI demonstration programs require `occidemo.sql` to be run.

 **Note:**

You must unlock the JONES account and set the password before creating the demonstrations.

Use the `demo_rdbms.mk` make file, which is located in the `$ORACLE_HOME/rdbms/demo` directory, to create the demonstration programs. For example, to compile and link the `cdemo1` demonstration program, run the following command:

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

To create the C demonstration programs for OCI, run the following command:

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

To create the C++ demonstration programs for OCCI, run the following command:

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

6.7.2 OCI and OCCI User Programs

You can use the `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk` make file to build user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_rdbms32.mk` to build 32-bit user programs on a 64-bit operating system. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*FORTRAN:

Platform	64-bit Make File	32-Bit Make File
Oracle Solaris, Linux x86-64, IBM AIX on POWER Systems (64-Bit), and HP-UX	<code>demo_rdbms.mk</code>	<code>demo_rdbms32.mk</code>

See Also:

The make file for more information about building user programs

6.8 Running Oracle JDBC/OCI Programs with a 64-Bit Driver

Note:

- The information in this section applies to Oracle Solaris, Linux x86-64, IBM: Linux on System z, IBM AIX on POWER Systems (64-Bit), and HP-UX platforms.
- You can use the instructions and make files described in this section to create JDBC/OCI user programs that use a 64-bit driver.

To run JDBC/OCI demonstration programs with a 64-bit driver:

1. Add `$ORACLE_HOME/jdbc/lib/ojdbc5.jar` to the start of the `CLASSPATH` environment variable value for each of the following files:

```
jdbc/demo/samples/jdbcoci/Makefile
jdbc/demo/samples/generic/Inheritance/Inheritance1/Makefile
jdbc/demo/samples/generic/Inheritance/Inheritance2/Makefile
jdbc/demo/samples/generic/Inheritance/Inheritance3/Makefile
jdbc/demo/samples/generic/JavaObject1/Makefile
jdbc/demo/samples/generic/NestedCollection/Makefile
```

2. Modify the `JAVA` and `JAVAC` variables in the `$(ORACLE_HOME)/jdbc/demo/samples/generic/Makefile` file to specify the JDK location and the `-d64` flag, as follows:

```
JAVA=${ORACLE_HOME}/java/bin/java -d64  
JAVAC=${ORACLE_HOME}/java/bin/javac -d64
```

3. Set the `LD_LIBRARY_PATH_64` environment variable to include the `$(ORACLE_HOME)/lib` directory.

 **Note:**

On IBM AIX on POWER Systems (64-Bit), the `LIBPATH` variable is the `LD_LIBRARY_PATH_64` variable equivalent. You must use the `LIBPATH` variable on IBM AIX on POWER Systems (64-Bit) instead of the `LD_LIBRARY_PATH_64` variable.

6.9 Custom Make Files

Oracle recommends that you use the `demo_product.mk` make files provided with the software to create user programs as described in the product-specific sections of this chapter. If you modify the provided make file or if you choose to use a custom-written make file, then remember that 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.
Except for IBM AIX on POWER Systems (64-Bit), the order of the Oracle libraries is essential on all platforms for the following reasons:
 - Oracle libraries are mutually referential. For example, functions in library A call functions in library B, and functions in library B call functions in library A.
 - The HP-UX linkers are one-pass linkers. The IBM AIX on POWER Systems (64-Bit), Linux, and Oracle Solaris linkers are two-pass linkers.
- Add the library 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`, then you must be aware of how macro and suffix processing differs from the `make` utility provided with the operating system. Oracle make files are tested and supported with the `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.

6.10 Correcting Undefined Symbols

Oracle provides the `symfind` utility to assist you in locating a library or object file where a symbol is defined. When linking a program, undefined symbols are a common error that produce 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: irrecoverable: 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 is displayed, then verify 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, which is used to locate the `sqlcex` symbol:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <*>.a, .o, .so
-----
Command:          /u01/app/oracle/product/19.0.0/bin/symfind sqlcex
Local Directory:  /u01/app/oracle/product/19.0.0
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.sl
[35]   |          0|    44|FUNC |GLOB |0    |5    |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./.lib/libsql.a
```

6.11 Multithreaded Applications

The Oracle libraries provided with this release are thread-safe, they support multithreaded applications.



See Also:

*Pro*C/C++ Programmer's Guide* for more information on Multithreaded Applications

6.12 Using Signal Handlers

Oracle Database uses signals for two-task communication. Signals are installed in a user process when the process connects to the database and are removed when it disconnects.

The following table describes the signals that Oracle Database uses for two-task communication.

Table 6-4 Signals for Two-Task Communication

Signal	Description
SIGCLD	The pipe driver uses SIGCLD, also referred to as SIGCHLD, when an Oracle process terminates. The operating system kernel sends a SIGCLD signal to the user process. The signal handler uses the <code>wait()</code> routine to determine 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	Oracle Net 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, then 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 it applies only 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	Oracle Net TCP/IP drivers use SIGURG to send out-of-band breaks from the user process to the Oracle process.

The listed signals affect all 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, then you can have multiple signal handles 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, then you can have multiple signal handlers.

[Example 6-1](#) shows how to set up a signal routine and a catching routine.

Example 6-1 Signal Routine and 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.
*/

```

[Example 6-2](#) shows how to use the `osnsui()` and the `osncui()` routines in an application program.

Example 6-2 `osnsui()` and `osncui()` Routine Template

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

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

    int handle, err;
    ...

    /* Set up the user interrupt handler */
    if (err = osnsui(&handle, sig_handler, (char *) 0))
    {
        /* If the return value is nonzero, then an error has occurred
        Take appropriate action for the error. */
        ...
    }

    ...

    /* Clear the interrupt handler */
    if (err = osncui(handle))
    {
        /* If the return value is nonzero, then an error has occurred
        Take appropriate action for the error. */
        ...
    }

    ...
}
```

6.13 XA Functionality

Oracle XA is the Oracle implementation of the X/Open Distributed Transaction Processing XA interface. The XA standard specifies a bidirectional interface between resource managers that provide access to shared resources within transactions, and between a transaction service that monitors and resolves transactions.

Oracle Call Interface has the 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 when using the XA dynamic registration (Oracle XA switch `xaoswd`).

Oracle Database XA calls are defined in both the client shared library (`libclntsh.a`, `libclntsh.sl`, `libclntsh.so`, or `libclntsh.dylib` depending on the platform) and the client static library (`libclntst11.a`). These libraries are located in the `$ORACLE_HOME/lib` directory.

7

SQL*Loader and PL/SQL Demonstrations

This chapter describes how to build and run the SQL*Loader and PL/SQL demonstration programs available with Oracle Database. It contains the following sections:

- [SQL*Loader Demonstrations](#)
- [PL/SQL Demonstrations](#)
- [Calling 32-Bit External Procedures from 64-Bit Oracle Database PL/SQL](#)

Note:

To use the demonstrations described in this chapter, you must install Oracle Database Examples included on the Oracle Database 19c Examples media. You must unlock JONES account and set the password before creating the demonstrations.

7.1 SQL*Loader Demonstrations

Run the `ulcase.sh` file to run the SQL*Loader demonstrations. To run an individual demonstration, read the information contained in the file to determine how to run it.

7.2 PL/SQL Demonstrations

PL/SQL includes many demonstration programs. You must build database objects and load sample data before using these programs. To build the objects and load the sample data:

1. Change directory to the PL/SQL demonstrations directory:

```
$ cd $ORACLE_HOME/plsql/demo
```

2. Start SQL*Plus, and enter the following command:

```
$ sqlplus
SQL> CONNECT JONES
Enter password: password
```

3. Run the following commands to build the objects and load the sample data:

```
SQL> @exampbld.sql
SQL> @examplod.sql
```

 **Note:**

Build the demonstrations as any Oracle user with sufficient privileges.
Run the demonstrations as the same Oracle user.

PL/SQL Kernel Demonstrations

The following PL/SQL kernel demonstrations are available with the software:

- `examp1.sql` to `examp8.sql`
- `examp11.sql` to `examp14.sql`
- `sample1.sql` to `sample4.sql`
- `extproc.sql`

To compile and run the `exampn.sql` or `samplen.sql` PL/SQL kernel demonstrations:

1. Start SQL*Plus, and enter the following command:

```
$ cd $ORACLE_HOME/plsql/demo
$ sqlplus
SQL> CONNECT JONES
Enter password: password
```

2. Run a command similar to the following to run a demonstration, where `demo_name` is the name of the demonstration:

```
SQL> @demo_name
```

To run the `extproc.sql` demonstration:

1. If required, add an entry for external procedures to the `tnsnames.ora` file, similar to the following:

```
EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL = IPC)( KEY = EXTPROC))
    )
    (CONNECT_DATA =
      (SID = PLSExtProc)
    )
  )
```

2. If required, add an entry for external procedures to the `listener.ora` file, similar to the following:

 **Note:**

The value that you specify for `SID_NAME` in the `listener.ora` file must match the value that you specify for `SID` in the `tnsnames.ora` file.

- On Oracle Solaris, Linux, and HP-UX:

```
SID_LIST_LISTENER =
  (SID_LIST =
```



```

(SID_DESC=
  (SID_NAME=PLSExtProc)
  (ORACLE_HOME=oracle_home_path)
  (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
    LD_LIBRARY_PATH=oracle_home_path/plsql/demo)
  (PROGRAM=extproc)
)
)

```

- On IBM AIX on POWER Systems (64-Bit):

```

SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC=
  (SID_NAME=PLSExtProc)
  (ORACLE_HOME=oracle_home_path)
  (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
    LIBPATH=oracle_home_path/plsql/demo)
  (PROGRAM=extproc)
)
)
)

```

3. Change directory to \$ORACLE_HOME/plsql/demo.
4. Run the following command to create the `extproc.so` shared library, build the required database objects, and load the sample data:

```
$ make -f demo_plsql.mk extproc.so exampbld examplod
```

Alternatively, if you have already built the database objects and loaded the sample data, then run the following command:

```
$ make -f demo_plsql.mk extproc.so
```

5. From SQL*Plus, run the following commands:

```

SQL> CONNECT SYSTEM
Enter password: system_password
SQL> GRANT CREATE LIBRARY TO JONES;
SQL> CONNECT JONES
Enter password: password
SQL> CREATE OR REPLACE LIBRARY demolib IS
  2 'oracle_home_path/plsql/demo/extproc.so';
  3 /

```



Note:

CREATE LIBRARY is a very high privilege. This privilege must be granted only to trusted users.

6. To start the demonstration, run the following command:

```
SQL> @extproc
```

Example 7-1 PL/SQL Precompiler Demonstrations

 **Note:**

The `make` commands shown in this section build the required database objects and load the sample data in the JONES schema.

The following precompiler demonstrations are available:

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

To build the PL/SQL precompiler demonstrations, set the library path environment variable to include the `$ORACLE_HOME/lib` directory, and run the following commands:

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

To build a single demonstration, run its name as the argument in the `make` command. For example, to build the `examp9` demonstration, run the following command:

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

To start the `examp9` demonstration, run the following command:

```
$ ./examp9
```

7.3 Calling 32-Bit External Procedures from 64-Bit Oracle Database PL/SQL

 **Note:**

This section applies to any 64-Bit Oracle Database.

Starting with Oracle Database 11g Release 2 (11.2), `extproc32` is no longer available from 64-bit Oracle database install. Therefore, if you have a requirement to run 32-bit external procedures from 64-bit Oracle database, you must obtain 32-bit `extproc` by installing the corresponding 32-bit client software for your platform. Specifically, you must choose custom install within 32-bit client installation, and then select both Oracle Database Utilities and Oracle listener.

In other words, you need a separate Oracle home (32-bit) to run the 32-bit `extproc`. The executable name is not `extproc32` anymore, but simply `extproc`.

To enable 32-bit external procedures on 64-bit Oracle database environment, you must configure 32-bit listener for `extproc` and specify Oracle home (from the 32-bit client install) for the `extproc listener.ora` entry.

8

Tuning Oracle Database

This chapter describes how to tune Oracle Database. It contains the following sections:

- [Importance of Tuning](#)
- [Operating System Tools](#)
- [Tuning Memory Management](#)
- [Tuning Disk Input-Output](#)
- [Monitoring Disk Performance](#)
- [System Global Area](#)
- [Tuning the Operating System Buffer Cache](#)

8.1 Importance of Tuning

The intent of this section is to efficiently tune and optimize the performance of Oracle Database. Frequent tuning enhances system performance and prevents data bottlenecks.

Before tuning the database, you must observe its normal behavior by using the tools described in the "[Operating System Tools](#)" section.

8.2 Operating System Tools

Several operating system tools are available to enable you 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, context switching, and I/O for the entire system.

This section provides information about the following common tools:

- [vmstat](#)
- [sar](#)
- [iostat](#)
- [swap_ swapinfo_ swapon_ or lsofs](#)
- [Oracle Solaris Tools](#)
- [Linux Tools](#)
- [IBM AIX on POWER Systems \(64-Bit\) Tools](#)
- [HP-UX Tools](#)

 **See Also:**

The operating system documentation and man pages for more information about these tools

8.2.1 vmstat

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

Use the `vmstat` command to view process, virtual memory, disk, trap, and CPU activity, depending on the switches that you supply with the command. Run one of the following commands to display a summary of CPU activity six times, at five-second intervals:

- On Oracle Solaris and HP-UX:

```
$ vmstat -S 5 6
```

 **Note:**

The `-s` option displays the swapping statistics.

- On Linux and IBM AIX on POWER Systems (64-bit):

```
$ vmstat 5 6
```

The following is a sample output of this command on Linux:

procs		memory				swap		io		system			cpu			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	130668	103604	198144	5029000	0	0	1	68	8	6	0	0	100	0	0
0	0	130668	103604	198144	5029000	0	0	0	86	226	352	0	0	100	0	0
0	0	130668	103604	198148	5029000	0	0	0	58	223	357	0	0	100	0	0
0	0	130668	103604	198152	5029004	0	0	0	68	223	358	0	0	100	0	0
0	0	130668	103604	198152	5029004	0	0	0	56	223	357	0	0	100	0	0
0	0	130668	103604	198152	5029004	0	0	0	57	228	362	0	0	100	0	0

The following is a sample output of the `$ vmstat -S 1 2` command on HP-UX:

procs		memory			page				faults		cpu						
r	b	w	avm	free	si	so	pi	po	fr	de	sr	in	sy	cs	us	sy	id
1	0	0	112085	2189167	0	0	3	0	0	0	1	1033	32186	108	1	0	98
1	0	0	112085	2189074	0	0	4	0	0	0	0	1022	508	60	0	0	100

The `w` sub column, under the `procs` column, shows the number of potential processes that have been swapped out and written to disk. If the value is not zero, then swapping occurs and the system is short of memory.

For HP-UX and Oracle Solaris, the `si` and `so` columns under the `page` column indicate the number of processes swapped-in and swapped-out per second, respectively. Ideally these values should be zero.

For Linux, the `si` and `so` columns represent amount of memory swapped-in and swapped-out.

The `sr` column under the `page` column indicates the scan rate. High scan rates are caused by a shortage of available memory.

The `pi` and `po` columns under the `page` column indicate the number of page-ins and page-outs per second, respectively. It is normal for the number of page-ins and page-outs to increase. Some paging always occurs even on systems with sufficient available memory.

See Also:

Refer to the `vmstat` man page for your platform for information about interpreting the output

8.2.2 sar

Depending on the switches that you supply with the command, use the `sar` (system activity reporter) command to display cumulative activity counters in the operating system.

On UNIX systems, the following command displays a summary of the input and output activity every ten seconds:

```
$ sar -b 10 10
```

The following example shows the output of this command on a Linux system:

```
10:28:01      tps      rtps      wtps      bread/s      bwrtn/s
10:28:11      17.20      0.00      17.20      0.00      300.80
10:28:21      46.40      0.00      46.40      0.00      467.20
10:28:31      16.40      0.00      16.40      0.00      283.20
10:28:41      15.60      0.00      15.60      0.00      275.20
10:28:51      17.02      0.00      17.02      0.00      254.65
10:29:01      35.80      0.00      35.80      0.00      414.40
10:29:11      15.80      0.00      15.80      0.00      273.60
10:29:21      17.40      0.00      17.40      0.00      262.40
10:29:31      32.20      0.00      32.20      0.00      406.40
10:29:41      20.98      0.00      20.98      0.00      354.85

Average:      23.48      0.00      23.48      0.00      329.28
```

The `sar` output provides a snapshot of system input and output activity at a given point in time. If you specify the interval time with multiple options, then the output can become difficult to read. If you specify an interval time of less than 5, then the `sar` activity itself can affect the output.

See Also:

The man page for more information about `sar`

8.2.3 iostat

Use the `iostat` command to view terminal and disk activity, depending on the switches that you supply with the command. The output from the `iostat` command does not include disk request queues, but it shows which disks are busy. You can use this information to balance the Input-Output loads.

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

```
$ iostat 5 5
```

The following is sample output of the command on Oracle Solaris:

```

tty          blkdev0          sd1          sd2          sd3          cpu
tin tout kps tps serv kps tps serv kps tps serv kps tps serv us sy st id
0 1 0 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 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 0 100
0 16 0 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

```

Use the `iostat` command to look for large disk request queues. A request queue shows how long the Input-Output requests on a particular disk device must wait to be serviced. Request queues are caused by a high volume of Input-Output requests to that disk or by Input-Output with long average seek times. Ideally, disk request queues should be at or near zero.

8.2.4 swap, swapinfo, swapon, or lssps

Use the `swap`, `swapinfo`, `swapon`, or `lssps` command to report information about swap space usage. A shortage of swap space can stop processes to respond, leading to process failures with Out of Memory errors. The following table lists the appropriate command to use for each platform:

Platform	Command
Oracle Solaris	<code>swap -lh</code> , <code>swap -sh</code> , and <code>zfs list rpool/swap</code>
Linux	<code>swapon -s</code>
IBM AIX on POWER Systems (64-bit)	<code>lssps -a</code>
HP-UX	<code>swapinfo -m</code>

The following example shows sample output from the `swap -l` command on Oracle Solaris:

```

swapfile          dev      swaplo  blocks    free
/dev/zvol/dsk/rpool/swap  274,1      8    20971447    20971447

```

8.2.5 Oracle Solaris Tools

On Oracle Solaris systems, use the `mpstat` command to view statistics for each processor in a multiprocessor system. Each row of the table represents the activity of one processor. The first row summarizes all activity since the last system restart. Each

subsequent row summarizes activity for the preceding interval. All values are events per second unless otherwise noted. The arguments are for time intervals between statistics and number of iterations.

The following example shows sample output from the `mpstat` command:

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  st idl
  0  3  0  2    319 103  109  0  19  11  0  73  0  1  0  99
  1  4  0  1    71  1   95  0  14  12  0  96  0  0  0  99
```

8.2.6 Linux Tools

On Linux systems, use the `top`, `free`, and `cat /proc/meminfo` commands to view information about swap space, memory, and buffer usage.

8.2.7 IBM AIX on POWER Systems (64-Bit) Tools

The following sections describe tools available on IBM AIX on POWER Systems (64-bit):

- [Base Operation System Tools](#)
- [Performance Toolbox](#)
- [System Management Interface Tool](#)

See Also:

The IBM AIX on POWER Systems (64-bit) operating system documentation and man pages for more information about these tools

8.2.7.1 Base Operation System Tools

The IBM AIX on POWER Systems (64-bit) Base Operation System contains performance tools that are historically part of UNIX systems or are required to manage the implementation-specific features of IBM AIX on POWER Systems (64-bit). The following table lists the most important Base Operation System tools.

Tool	Function
<code>lsattr</code>	Displays the attributes of devices
<code>lslv</code>	Displays information about a logical volume or the logical volume allocations of a physical volume
<code>netstat</code>	Displays the contents of network-related data structures
<code>nfsstat</code>	Displays statistics about Network File System and Remote Procedure Call activity
<code>nice</code>	Changes the initial priority of a process
<code>no</code>	Displays or sets network options
<code>ps</code>	Displays the status of one or more processes
<code>reorgvg</code>	Reorganizes the physical-partition allocation within a volume group

Tool	Function
time	Displays the elapsed execution, user CPU processing, and system CPU processing time
trace	Records and reports selected system events
vmo	Manages Virtual Memory Manager tunable parameters

8.2.7.2 Performance Toolbox

The IBM AIX on POWER Systems (64-bit) Performance Toolbox contains tools for monitoring and tuning system activity locally and remotely. The Performance Tool Box consists of two main components, the Performance Tool Box Manager and the Performance Tool Box Agent. The Performance Tool Box Manager collects and displays data from various systems in the configuration by using the `xmperf` utility. The Performance Tool Box Agent collects and transmits data to the Performance Tool Box Manager by using the `xmserd` daemon. The Performance Tool Box Agent is also available as a separate product called Performance Aide for IBM AIX on POWER Systems (64-bit).

Both Performance Tool Box and Performance Aide include the monitoring and tuning tools listed in the following table:

Tool	Description
fdpr	Optimizes an executable program for a particular workload
filemon	Uses the trace facility to monitor and report the activity of the file system
fileplace	Displays the placement of blocks of a file within logical or physical volumes
lockstat	Displays statistics about contention for kernel locks
lvedit	Facilitates interactive placement of logical volumes within a volume group
netpmon	Uses the trace facility to report on network Input-Output and network-related CPU usage
rmss	Simulates systems with various memory sizes for performance testing
svmon	Captures and analyzes information about virtual-memory usage
syscalls	Records and counts system calls
tprof	Uses the <code>trace</code> facility to report CPU usage at module and source-code-statement levels
BigFoot	Reports the memory access patterns of processes
stem	Permits subroutine-level entry and exit instrumentation of existing executables

 **See Also:**

- *Performance Toolbox Version 2 and 3 Guide and Reference* for information about these tools
- *AIX 5L Performance Management Guide* for information about the syntax of some of these tools

8.2.7.3 System Management Interface Tool

The IBM AIX on POWER Systems (64-bit) System Management Interface Tool (SMIT) provides a menu-driven interface to various system administrative and performance tools. By using SMIT, you can navigate through large numbers of tools and focus on the jobs that you want to perform.

8.2.8 HP-UX Tools

The following performance analysis tools are available on HP-UX systems:

- GlancePlus/UX

This HP-UX utility is an online diagnostic tool that measures the activities of the system. GlancePlus displays information about how system resources are used. It displays dynamic information about the system Input-Output, CPU, and memory usage on a series of screens. You can use the utility to monitor how individual processes are using resources.

- HP Programmer's Analysis Kit

HP Programmer's Analysis Kit consists of the following tools:

- Puma

This tool collects performance statistics during a program run. It provides several graphical displays for viewing and analyzing the collected statistics.

- Thread Trace Visualizer

This tool displays trace files produced by the instrumented thread library, `libpthread_tr.sl`, in a graphical format. It enables you to view how threads are interacting and to find where threads are blocked waiting for resources.

HP Programmer's Analysis Kit is bundled with the HP Fortran 77, HP Fortran 90, HP C, HP C++, HP ANSI C++, and HP Pascal compilers.

The following table lists the performance tuning tools that you can use for additional performance tuning on HP-UX:

Tool	Description
<code>caliper</code> (Itanium only)	Collects run-time application data for system analysis tasks such as cache misses, translation look-aside buffer or instruction cycles, along with fast dynamic instrumentation. It is a dynamic performance measurement tool for C, C++, Fortran, and assembly applications
<code>gprof</code>	Creates an execution profile for programs

Tool	Description
<code>monitor</code>	Monitors the program counter and calls to certain functions
<code>netfmt</code>	Monitors the network
<code>netstat</code>	Reports statistics on network performance
<code>nfsstat</code>	Displays statistics about Network File System and Remote Procedure Call activity
<code>nettl</code>	Captures network events or packets by logging and tracing
<code>prof</code>	Creates an execution profile of C programs and displays performance statistics for the program, showing where the program is spending most of its execution time
<code>profil</code>	Copies program counter information into a buffer
<code>top</code>	Displays the top processes on the system and periodically updates the information

8.3 Tuning Memory Management

Start the memory tuning process by measuring paging and swapping space to determine how much memory is available. After you determine the system memory usage, tune the Oracle buffer cache.

The Oracle buffer manager ensures that the most frequently accessed data is cached longer. If you monitor the buffer manager and tune the buffer cache, then you can significantly improve Oracle Database performance. The optimal Oracle Database buffer size for the system depends on the overall system load and the relative priority of Oracle Database over other applications.

This section includes the following topics:

- [Allocating Sufficient Swap Space](#)
- [Monitoring Paging](#)
- [Adjusting Oracle Block Size](#)
- [Allocating Memory Resource](#)

8.3.1 Allocating Sufficient Swap Space

Try to minimize swapping because it causes significant operating system overhead. To check for swapping, use the `sar` or `vmstat` commands. For information about the appropriate options to use with these commands, refer to the man pages.

If the system is swapping and you must conserve memory, then:

- Avoid running unnecessary system daemon processes or application processes.
- Decrease the number of database buffers to free some memory.
- Decrease the number of operating system file buffers.

To determine the amount of swap space, run one of the following commands, depending on the platform:

Platform	Command
Oracle Solaris	<code>swap -l, swap -s, and zfs get volsize rpool/swap</code>
Linux	<code>swapon -s</code>
IBM AIX on POWER Systems (64-bit)	<code>lsps -a</code>
HP-UX	<code>swapinfo -m</code>

Monitor the use of swap space, and increase it as required. The following table describes the initially recommended relationship between installed RAM and the configured swap space requirement:

RAM	Swap Space
Between 1 GB and 2 GB	1.5 times the size of RAM
Between 2 GB and 16 GB	Equal to the size of RAM
More than 16 GB	16 GB

To add swap space to the system, run one of the following commands, depending on the platform:

Platform	Command
Oracle Solaris	Use one of the following options: <ul style="list-style-type: none"> For ZFS file system, extend the swap volume using the following command: <pre>zfs set volsize=newsize rpool/swap</pre> where <i>newsize</i> is the size of the swap volume that you want to increase. For a non-ZFS file system, use the following command: <pre>swap -a</pre>
Linux	<code>swapon -a</code>
IBM AIX on POWER Systems (64-bit)	<code>chps</code> or <code>mkps</code>
HP-UX	<code>swapon</code>

See Also:

- [My Oracle Support note 1587357.1](#)
- The operating system documentation for more information about these commands

 **Note:**

Starting with 12c, Oracle Database uses the Optimized Shared Memory (OSM) model of Oracle Solaris to implement Automatic Memory Management. Unlike DISM, OSM does not require the double allocation of swap disk space. For swap space requirements refer to the following note:

[My Oracle Support note 1010818.1](#)

8.3.2 Monitoring Paging

Paging may 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 may not noticeably affect the performance of the 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 `vmstat` or `sar` command to monitor paging.

 **See Also:**

The man pages or the operating system documentation for information about interpreting the results for the platform

In Oracle Solaris, `vmstat -p` indicates the number of address translation page faults. Address translation faults occur when a process refers to a valid page not in memory.

Analyzing memory related issues should start with checking the amount of free memory in the `vmstat` output. If free memory is low, then the `sr` (scan rate) column should be checked for non-zero values. This indicates that the page scanner is scanning memory pages to put back on the free list to be reused.

Anonymous (bad) paging can be observed in the `vmstat -p` command output under the `api` (anonymous page-in) and `apo` (anonymous page-out) columns. This kind of paging takes place when the system moves anonymous pages to the swap device during a memory shortage.

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

- Install more memory.
- Move some work to another system.
- Configure the System Global Area (SGA) to use less memory.

Related Topics

- [MOS Document: 1007494.1](#)

8.3.3 Adjusting Oracle Block Size

During read operations, entire operating system blocks are read from the disk. If the database block size is smaller than the operating system file system block size, then Input-Output bandwidth is inefficient. If you set Oracle Database block size to be a multiple of the file system block size, then you can increase performance by up to 5 percent.

The `DB_BLOCK_SIZE` initialization parameter sets the database block size. However, to change the value of this parameter, you must re-create the database.

To see the current value of the `DB_BLOCK_SIZE` parameter, run the `SHOW PARAMETER DB_BLOCK_SIZE` command in SQL*Plus.

8.3.4 Allocating Memory Resource

You can set parameters to automatically allocate memory based on the demands of workload and the requirements of various database instances running on the same system. The `MEMORY_TARGET` parameter specifies the Oracle systemwide usable memory for that instance and automatically tunes SGA and Process Global Area (PGA) components. The `MEMORY_MAX_TARGET` parameter identifies the value up to which the `MEMORY_TARGET` parameter can grow dynamically.

By default, the value for both these parameters is zero and there is no auto-tuning. You can activate auto-tuning by setting the `MEMORY_TARGET` parameter to a nonzero value. To dynamically enable the `MEMORY_TARGET` parameter, the `MEMORY_MAX_TARGET` parameter must be set at startup.

Note:

If you just set the `MEMORY_TARGET` parameter to a nonzero value, the `MEMORY_MAX_TARGET` parameter automatically acquires this value.

The `MEMORY_TARGET` and `MEMORY_MAX_TARGET` parameters are only supported on Linux, Oracle Solaris, HP-UX, and IBM AIX on POWER Systems (64-bit) platforms.

On Oracle Solaris, Dynamic Intimate Shared Memory is enabled for `MEMORY_TARGET` or `MEMORY_MAX_TARGET`. For more information, refer to [“Administering Oracle Database on Oracle Solaris”](#) section.

On Linux, some shared resource requirements are increased when `MEMORY_TARGET` or `MEMORY_MAX_TARGET` are enabled. For more information, refer to the [“Allocating Shared Resources”](#) section.

Tip:

You can set the `MEMORY_TARGET` and `MEMORY_MAX_TARGET` parameters based on original setup, memory available for Oracle on the computer, and workload memory requirements.

8.4 Tuning Disk Input-Output

Balance Input-Output evenly across all available disks to reduce disk access times. For smaller databases and those not using RAID, ensure that different data files and tablespaces are distributed across the available disks.

This section contains the following topics:

- [Using Automatic Storage Management](#)
- [Choosing the Appropriate File System Type](#)

8.4.1 Using Automatic Storage Management

If you choose to use Automatic Storage Management for database storage, then all database Input-Output is balanced across all available disk devices in the Automatic Storage Management disk group.

By using Automatic Storage Management, you avoid manually tuning disk Input-Output.

8.4.2 Choosing the Appropriate File System Type

Depending on the operating system, you can choose from a range of file system types. Each file system type has different characteristics. This fact can have a substantial impact on database performance. The following table lists common file system types:

File System	Platform	Description
ZFS	Oracle Solaris	Oracle Solaris ZFS file system
S5	HP-UX	UNIX System V file system
UFS	Oracle Solaris, IBM AIX on POWER Systems (64-bit), and HP-UX	Unix file system, derived from BSD UNIX
VxFS	Oracle Solaris, IBM AIX on POWER Systems (64-bit), and HP-UX	VERITAS file system
ext2/ext3	Linux	Extended file system for Linux
OCFS2	Linux	Oracle cluster file system
JFS/JFS2	IBM AIX on POWER Systems (64-bit)	Journalled file system
GPFS	IBM AIX on POWER Systems (64-bit)	General parallel file system

The suitability of a file system for an application is usually not documented. For example, even different implementations of the UFS are hard to compare. Depending on the file system that you choose, performance differences can be up to 20 percent. If you choose to use a file system, then:

- Make a new file system partition to ensure that the hard disk is clean and unfragmented.
- Perform a file system check on the partition before using it for database files.

- Distribute disk Input-Output as evenly as possible.
- If you are not using a logical volume manager or a RAID device, then consider placing log files on a different file system from data files.

 **See Also:**

“Tuning ZFS for Database Products” in *Oracle Solaris 11.3 Tunable Parameters Reference Manual* for more information on tuning ZFS for Oracle Database

8.5 Monitoring Disk Performance

The following sections describe the procedure for monitoring disk performance:

- [Monitoring Disk Performance on Other Operating Systems](#)
- [Using Disk Resync to Monitor Automatic Storage Management Disk Group](#)

8.5.1 Monitoring Disk Performance on Operating Systems

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

The following table describes the columns of the `sar -b` command output that are significant for analyzing disk performance:

Columns	Description
<code>bread/s, bwrit/s</code>	Blocks read and blocks written per second (important for file system databases)
<code>pread/s, pwrit/s</code>	Number of I/O operations per second on raw devices (important for raw partition database systems)

Key indicators are:

- The sum of the `bread`, `bwrit`, `pread`, and `pwrit` column values indicates the level of activity of the disk Input-Output subsystem. The higher the sum, the busier the Input-Output subsystem. The larger the number of physical drives, the higher the sum threshold number can be.
- The `%rcache` column value should be greater than 90 and the `%wcache` column value should be greater than 60. Otherwise, the system may be disk Input-Output bound.

8.5.2 Using Disk Resync to Monitor Automatic Storage Management Disk Group

Use the `alter diskgroup disk online` and `alter diskgroup disk offline` commands to temporarily suspend Input-Output to a set of disks. You can use these commands to perform regular maintenance tasks or upgrades such as disk firmware

upgrade. If transient failures occur on some disks in a disk group, then use `alter diskgroup disk online` to quickly recover the disk group.

8.6 System Global Area

The SGA is the Oracle structure that is located in shared memory. It contains static data structures, locks, and data buffers.

The maximum size of a single shared memory segment is specified by the `shmmax` kernel parameter.

The following table shows the recommended value for this parameter, depending on the platform:

Platform	Recommended Value
Oracle Solaris	4294967295 or 4 GB minus 16 MB
Linux	Minimum of the following values: <ul style="list-style-type: none">• Half the size of the physical memory installed on the system• 4 GB - 1 byte
IBM AIX on POWER Systems (64-bit)	NA
HP-UX	The size of the physical memory installed on the system

If the size of the SGA exceeds the maximum size of a shared memory segment (`shmmax` or `shm_max`), then Oracle Database attempts to attach more contiguous segments to fulfill the requested SGA size. The `shmseg` kernel parameter specifies the maximum number of segments that can be attached by any process. Set the following initialization parameters to control the size of the SGA:

- `DB_CACHE_SIZE`
- `DB_BLOCK_SIZE`
- `JAVA_POOL_SIZE`
- `LARGE_POOL_SIZE`
- `LOG_BUFFERS`
- `SHARED_POOL_SIZE`

Alternatively, set the `SGA_TARGET` initialization parameter to enable automatic tuning of the SGA size.

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

An Oracle Database configured with Shared Server requires a higher setting for the `SHARED_POOL_SIZE` initialization parameter, or a custom configuration that uses the `LARGE_POOL_SIZE` initialization parameter. If you installed the database with Oracle Universal Installer, then the value of the `SHARED_POOL_SIZE` parameter is set automatically by Oracle Database Configuration Assistant. However, if you created a database manually, then increase the value of the `SHARED_POOL_SIZE` parameter in the parameter file by 1 KB for each concurrent user.

Sufficient shared memory must be available to each Oracle process to address the entire SGA:

- [Determining the Size of the SGA](#)
- [System Resource Verifier Utility](#)
- [Guidelines for Setting Semaphore Parameters](#)
- [Shared Memory on IBM AIX on POWER Systems \(64-Bit\)](#)

8.6.1 Determining the Size of the SGA

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

- Run the following SQL*Plus command to display the size of the SGA for a running database:

```
SQL> SHOW SGA
```

The result is shown in bytes.

- Start the database instance to view the size of the SGA displayed next to the Total System Global Area heading.
- Run the `ipcs` command as the `oracle` user.

8.6.2 System Resource Verifier Utility

The System Resource Verifier utility (`sysresv`) is available with Oracle8i and later releases. It provides Oracle instance and operating system resource information for the Oracle system identifiers (`ORACLE_SID`) that you specify. This utility is located in `$ORACLE_HOME/bin`, but it can be used from other locations.

8.6.2.1 Purpose of the `sysresv` Utility

Use the `sysresv` utility to display the status of an Oracle instance and identify the operating system resources it uses, such as the memory and semaphore parameters. This utility is especially useful when multiple instances are running. For example, if an instance is not responsive, then you can use this utility to remove operating system resources.

You can use this utility when an Oracle instance has crashed or was aborted, and memory and semaphores related to this instance were not cleaned up automatically. This utility is also useful in determining which Oracle instance is running.

8.6.2.2 Preconditions for Using `sysresv`

To use the `sysresv` utility, you must have access to the System Global Area (SGA). To access the SGA, you must be the Oracle owner or a member of the group that owns the Oracle binary.

8.6.2.3 Syntax for `sysresv`

The syntax for the `sysresv` utility is as follows:

```
sysresv [-i] [-f] [-d on|off] [-l sid1[ sid2 ...]]
```

Where:

- `-i` Prompt before removing IPC resources for each `sid`
- `-f` Remove IPC resources without prompting for confirmation. This flag overrides the `-i` option
- `-d on|off` List IPC resources for each `sid` if on. If not specified, the default for `-d` is on
- `-l sid1 [sid2 sid3]` run the `sysresv` check against one or more space-delimited system identifiers

If `sysresv` is used without flags, then it reports IPC resources for the Oracle instance identified by the `$ORACLE_SID` environment variable in the Oracle installation owner user profile list of environment variables.

8.6.2.4 Examples of Using `sysresv`

The following example shows how to use the `sysresv` utility:

```
$ sysresv
IPV Resources for ORACLE_SID "sales" :
Shared Memory:
ID                KEY
10345             0x51c051ad
Semaphores
ID
10345             0x51c051ad
Oracle Instance alive for sid "sales"
```

8.6.3 Guidelines for Setting Semaphore Parameters

Use the following guidelines only if the default semaphore parameter values are too low to accommodate all Oracle processes:

Note:

Oracle recommends that you see the operating system documentation for more information about setting semaphore parameters.

1. Calculate the minimum total semaphore requirements using the following formula:

$$\text{sum (process parameters of all database instances on the system) + overhead for oracle background processes + system and other application requirements}$$
2. Set `semtns` (maximum number of semaphores allowed for the server) to this total.
3. Set `semmsl` (maximum number of semaphores in a semaphore set) to 250.
4. Set `semnli` (maximum number of semaphore sets) to `semtns/semmsl` rounded up to the nearest multiple of 1024.

 **See Also:**

My Oracle Support note 226209.1, "Linux: How to Check Current Shared Memory, Semaphore Values," at the following URL:

<https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=226209.1>

The semaphore parameters `semms`, `semmsl`, and `semnmi` are obsolete on Oracle Solaris 10 and later. See My Oracle Support note 1006158.1:

<https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1006158.1>

8.6.4 Shared Memory on IBM AIX on POWER Systems (64-Bit)

 **Note:**

The information in this section applies only to IBM AIX on POWER Systems (64-bit).

Shared memory uses common virtual memory resources across processes. Processes share virtual memory segments through a common set of virtual memory translation resources, for example, tables and cached entries, for improved performance.

Shared memory can be pinned to prevent paging and to reduce Input-Output overhead. To perform this, set the `LOCK_SGA` parameter to `true`. On IBM AIX on POWER Systems (64-bit) 5L, the same parameter activates the large page feature whenever the underlying hardware supports it.

Run the following command to make pinned memory available to Oracle Database:

```
$ /usr/sbin/vmo -r -o v_pinshm=1
```

Run a command similar to the following to set the maximum percentage of real memory available for pinned memory, where `percent_of_real_memory` is the maximum percent of real memory that you want to set:

```
$ /usr/sbin/vmo -r -o maxpin percent=percent_of_real_memory
```

When using the `maxpin percent` option, it is important that the amount of pinned memory exceeds the Oracle SGA size by at least 3 percent of the real memory on the system, enabling free pinnable memory for use by the kernel. For example, if you have 2 GB of physical memory and you want to pin the SGA by 400 MB (20 percent of the RAM), then run the following command:

```
$ /usr/sbin/vmo -r -o maxpin percent=23
```

 **Note:**

The default `maxpin_percent` value, which is set at 80 percent, works for most installations.

Use the `svmon` command to monitor the use of pinned memory during the operation of the system. Oracle Database attempts to pin memory only if the `LOCK_SGA` parameter is set to `true`. If the SGA size exceeds the size of memory available for pinning, then the portion of the SGA exceeding these sizes is allocated to ordinary shared memory.

Large Page Feature on IBM AIX on POWER Systems (64-Bit) POWER4 and POWER5 Based Systems

To turn on and reserve 10 large pages each of size 16 MB on a POWER4 or POWER5 system, run the following command:

```
$ /usr/sbin/vmo -r -o lgpg_regions=10 -o lgpg_size=16777216
```

This command proposes `bosboot` and warns that a restart is required for the changes to take affect.

Oracle recommends specifying enough large pages to contain the entire SGA. The Oracle database instance attempts to allocate large pages when the `LOCK_SGA` parameter is set to `true`.

The 16 MB pages are always pinned, and cannot be used for standard memory. If a pool of 16 MB size pages is configured, then this memory is unusable for allocation of standard memory even if no other application is currently using large pages.

The POWER5 based systems support 64 K pages. Oracle uses them for SGA if they are available. These 64K pages do not require any additional configuration and do not depend on `LOCK_SGA` parameter setting.

To monitor use of large pages, use the following command:

```
$ vmstat -P all
```

For the IBM AIX on POWER Systems (64-bit) operating system to use 16 MB pages, or pinned memory when allocating shared memory, the Oracle user ID must have `CAP_BYPASS_RAC_VMM` and `CAP_PROPAGATE` capabilities. User ID that is used to start the database instance must also have the same capabilities. In particular, when using large pages on an Oracle Real Application Cluster (Oracle RAC) database, where the `srvctl` command is used to start and stop the Oracle RAC database instances, it is also necessary to set the `CAP_BYPASS_RAC_VMM` and `CAP_PROPAGATE` capabilities for the root user.

 **See Also:**

The IBM AIX on POWER Systems (64-bit) documentation for more information about enabling and tuning pinned memory and large pages

Capabilities can be set and examined using the following commands:

- Run the following command to check the current capabilities:

```
$ lsuser -a capabilities oracle
```
- Add the `CAP_BYPASS_RAC_VMM` and `CAP_PROPAGATE` capabilities to this user ID:

```
$ chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE oracle
```

 **Note:**

Only the `root` user can display and set the capabilities attribute.

8.7 Tuning the Operating System Buffer Cache

Adjust the size of Oracle Database buffer cache. If memory is limited, then adjust the operating system buffer cache.

The operating system buffer cache holds blocks of data in memory while they are being transferred from memory to disk, or from disk to memory.

Oracle Database buffer cache is the area in memory that stores Oracle Database buffers.

If the amount of memory on the system is limited, then make a corresponding decrease in the operating system buffer cache size.

Use the `sar` command to determine which buffer caches you must increase or decrease.

A

Administering Oracle Database on Linux

This appendix contains information about administering Oracle Database on Linux.

It contains the following topics:

- [Supporting Asynchronous Input-Output](#)
- [Asynchronous Input-Output Support](#)
- [Enabling Direct Input-Output Support](#)
- [Enabling Simultaneous Multithreading](#)
- [Allocating Shared Resources](#)
- [About Creating cgroups on Linux Systems](#)
- [Overview of HugePages](#)

Note:

Starting with Oracle Database 11g Release 2 (11.2), Linux x86-64 and IBM: Linux on System z media does not contain Linux x86 binaries.

A.1 Supporting Asynchronous Input-Output

Note:

On Linux, Automatic Storage Management uses asynchronous Input-Output by default. Asynchronous Input-Output is not supported for database files stored on Network File Systems.

Oracle Database supports kernel asynchronous Input-Output. Asynchronous Input-Output is enabled by default on raw volumes. Automatic Storage Management uses asynchronous Input-Output by default.

By default, the `DISK_ASYNC_IO` initialization parameter in the parameter file is set to `TRUE`. To enable asynchronous Input-Output on file system files:

1. Ensure that all Oracle Database files are located on file systems that support asynchronous Input-Output.
2. Set the `FILESYSTEMIO_OPTIONS` initialization parameter in the parameter file to `ASYNCH` or `SETALL`.

 **Note:**

If the file system files are managed through ODM library interface or Direct NFS Client, asynchronous Input-Output is enabled by default. There is no need to set `FILESYSTEMIO_OPTIONS` to enable asynchronous Input-Output in these environments.

A.2 Asynchronous Input-Output Support

 **Note:**

On Linux, Automatic Storage Management uses asynchronous Input-Output by default. Asynchronous Input-Output is not supported for database files stored on Network File Systems.

Oracle Database supports kernel asynchronous Input-Output. This feature is disabled by default.

By default, the `DISK_ASYNC_IO` initialization parameter in the parameter file is set to `TRUE` to enable asynchronous I/O on raw devices. To enable asynchronous Input-Output on file system files:

1. Ensure that all Oracle Database files are located on file systems that support asynchronous Input-Output.
2. Set the `FILESYSTEMIO_OPTIONS` initialization parameter in the parameter file to `ASYNCH` to enable asynchronous Input-Output. If you want to enable both asynchronous Input-Output and direct Input-Output, set the `FILESYSTEMIO_OPTIONS` initialization parameter in the parameter file to `SETALL`.

A.3 Enabling Direct Input-Output Support

Direct Input-Output support is available and supported on Linux.

To enable direct Input-Output support:

- Set the `FILESYSTEMIO_OPTIONS` initialization parameter to `DIRECTIO`.
- Set the `FILESYSTEMIO_OPTIONS` initialization parameter in the parameter file to `SETALL`, which will enable both asynchronous Input-Output and direct Input-Output.

A.4 Enabling Simultaneous Multithreading

If Simultaneous Multithreading is enabled, then the `v$osstat` view reports two additional rows corresponding to the online logical (`NUM_LCPUS`) and virtual CPUs (`NUM_VCPUS`).

A.5 Allocating Shared Resources

To use the `MEMORY_TARGET` or `MEMORY_MAX_TARGET` feature, the following kernel parameters must be modified.

- `/dev/shm` mount point should be equal in size or larger than the value of `SGA_MAX_SIZE`, if set, or should be set to be at least `MEMORY_TARGET` or `MEMORY_MAX_TARGET`, whichever is larger. For example, with `MEMORY_MAX_TARGET=4GB` only set, to create a 4 GB system on the `/dev/shm` mount point:

- Run the following command as the `root` user:

```
# mount -t tmpfs shmfs -o size=4g /dev/shm
```

- Ensure that the in-memory file system is mounted when the system restarts, add an entry in the `/etc/fstab` file similar to the following:

```
tmpfs /dev/shm tmpfs size=4g 0
```

- The number of file descriptors for each Oracle instance are increased by `512*PROCESSES`. Therefore, the maximum number of file descriptors should be at least this value, plus some more for the operating system requirements. For example, if the `cat /proc/sys/fs/file-max` command returns 32768 and `PROCESSES` are 100, you can set it to 6815744 or higher as `root`, to have 51200 available for Oracle. Use one of the following options to set the value for the `file-max` descriptor.

- Run the following command:

```
echo 6815744 > /proc/sys/fs/file-max
```

OR

- Modify the following entry in the `/etc/sysctl.conf` file and restart the system as `root`.

```
fs.file-max = 6815744
```

- Per-process number of file descriptors must be at least 512. For example, as `root` run the following command.

- On `bash` and `sh`:

```
# ulimit -n
```

- On `csh`:

```
# limit descriptors
```

If the preceding command returns 200, then run the following command to set the value for the per processor file descriptors limit, for example to 1000:

- On `bash` and `sh`:

```
# sudo sh  
# ulimit -n 1000
```

- On `csh`:

```
# sudo sh  
# limit descriptors 1000
```


- `MEMORY_TARGET` and `MEMORY_MAX_TARGET` cannot be used when `LOCK_SGA` is enabled. `MEMORY_TARGET` and `MEMORY_MAX_TARGET` also cannot be used with huge pages on Linux.

A.6 About Creating Cgroups on Linux Systems

Cgroups, or control groups, improve database performance by associating a dedicated set of CPUs to a database instance. Each database instance can only use the resources in its cgroup.

When consolidating on a large server, you may want to restrict the database to a specific subset of the CPU and memory. This feature makes it easy to enable CPU and memory restrictions for an Oracle Database instance.

Use the `setup_processor_group.sh` script to create cgroups. Download this script from note 1585184.1 on the My Oracle Support website:

<https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1585184.1>

A.7 Overview of HugePages

HugePages is a feature integrated into the Linux kernel 2.6. Enabling HugePages makes it possible for the operating system to support memory pages greater than the default (usually 4 KB). Using very large page sizes can improve system performance by reducing the amount of system resources required to access page table entries. HugePages is useful for both 32-bit and 64-bit configurations. HugePage sizes vary from 2 MB to 256 MB, depending on the kernel version and the hardware architecture. For Oracle Databases, using HugePages reduces the operating system maintenance of page states, and increases Translation Lookaside Buffer (TLB) hit ratio.



Note:

Transparent Hugepages is currently not an alternative to manually configure HugePages.

This section includes the following topics:

- [Reviewing HugePages Memory Allocation](#)
- [Using HugePages on Linux](#)
- [Tuning SGA With HugePages](#)
- [Configuring HugePages on Linux](#)
- [Restrictions for HugePages Configurations](#)
- [Disabling Transparent HugePages](#)

A.7.1 Reviewing HugePages Memory Allocation

Review this information if your operating system has HugePages enabled.

On Linux platform installations, Oracle recommends that you use HugePages to obtain the best performance for Oracle Databases. When you upgrade Oracle Grid

Infrastructure and Oracle Databases on servers that have HugePages enabled, Oracle recommends that you review your HugePages memory allocation requirements.

GIMR and HugePages Memory

Oracle Grid Infrastructure installations include the Grid Infrastructure Management Repository (GIMR). When HugePages is configured on cluster member nodes, the GIMR system global area (SGA) is installed into HugePages memory. The GIMR SGA occupies up to 1 GB of HugePages memory. Oracle Grid Infrastructure starts up before Oracle Databases installed on the cluster.

If your cluster member node operating system memory allocations to HugePages are insufficient for the size of the SGAs for all of the Oracle Database instances on the cluster, then you may find that one or more of your Oracle Database SGAs are mapped to regular pages, instead of Huge Pages, which reduces expected performance. To avoid this issue, when you plan your upgrade, ensure that the memory you reserve for HugePages is large enough to accommodate your memory requirements.

Allocate memory to HugePages large enough for all databases planned to run SGA on the cluster, and to accommodate the SGA for the Grid Infrastructure Management Repository.

A.7.2 Using HugePages on Linux

To enable Oracle Database to use large pages (sometimes called HugePages) on Linux, set the value of the `vm.nr_hugepages` kernel parameter to specify the number of large pages that you want to reserve. You must specify adequate large pages to hold the entire SGA for the database instance. To determine the required parameter value, divide the SGA size for the instance by the size of a large page, then round up the result to the nearest integer.

To determine the default large page size, run the following command:

```
# grep Hugepagesize /proc/meminfo
```

For example, if `/proc/meminfo` lists the large page size as 2 MB, and the total SGA size for the instance is 1.6 GB, then set the value for the `vm.nr_hugepages` kernel parameter to 820 ($1.6 \text{ GB} / 2 \text{ MB} = 819.2$).

A.7.3 Tuning SGA With HugePages

Without HugePages, the operating system keeps each 4 KB of memory as a page. When it allocates pages to the database System Global Area (SGA), the operating system kernel must continually update its page table with the page lifecycle (dirty, free, mapped to a process, and so on) for each 4 KB page allocated to the SGA.

With HugePages, the operating system page table (virtual memory to physical memory mapping) is smaller, because each page table entry is pointing to pages from 2 MB to 256 MB.

Also, the kernel has fewer pages whose lifecycle must be monitored. For example, if you use HugePages with 64-bit hardware, and you want to map 256 MB of memory, you may need one page table entry (PTE). If you do not use HugePages, and you want to map 256 MB of memory, then you must have $256 \text{ MB} * 1024 \text{ KB} / 4 \text{ KB} = 65536$ PTEs.

HugePages provides the following advantages:

- Increased performance through increased TLB hits
- Pages are locked in memory and never swapped out, which provides RAM for shared memory structures such as SGA
- Contiguous pages are preallocated and cannot be used for anything else but for System V shared memory (for example, SGA)
- Less bookkeeping work for the kernel for that part of virtual memory because of larger page sizes

A.7.4 Configuring HugePages on Linux

Complete the following steps to configure HugePages on the computer:

1. Run the following command to determine if the kernel supports HugePages:

```
$ grep Huge /proc/meminfo
```

2. Some Linux systems do not support HugePages by default. For such systems, build the Linux kernel using the `CONFIG_HUGETLBFS` and `CONFIG_HUGETLB_PAGE` configuration options. `CONFIG_HUGETLBFS` is located under File Systems and `CONFIG_HUGETLB_PAGE` is selected when you select `CONFIG_HUGETLBFS`.
3. Edit the `memlock` setting in the `/etc/security/limits.conf` file. The `memlock` setting is specified in KB, and the maximum locked memory limit should be set to at least 90 percent of the current RAM when HugePages memory is enabled and at least 3145728 KB (3 GB) when HugePages memory is disabled. For example, if you have 64 GB RAM installed, then add the following entries to increase the maximum locked-in-memory address space:

```
* soft memlock 60397977
* hard memlock 60397977
```

You can also set the `memlock` value higher than your SGA requirements.

4. Log in as `oracle` user again and run the `ulimit -l` command to verify the new `memlock` setting:

```
$ ulimit -l
60397977
```

5. Run the following command to display the value of `Hugepagesize` variable:

```
$ grep Hugepagesize /proc/meminfo
```

6. Complete the following procedure to create a script that computes recommended values for `hugepages` configuration for the current shared memory segments:

- a. Create a text file named `hugepages_settings.sh`.

- b. Add the following content in the file:

```
#!/bin/bash
#
# hugepages_settings.sh
#
# Linux bash script to compute values for the
# recommended HugePages/HugeTLB configuration
#
# Note: This script does calculation for all shared memory
# segments available when the script is run, no matter it
```

```

# is an Oracle RDBMS shared memory segment or not.
# Check for the kernel version
KERN=`uname -r | awk -F. '{ printf("%d.%d\n", $1, $2); }'`
# Find out the HugePage size
HPG_SZ=`grep Hugepagesize /proc/meminfo | awk {'print $2}'`
# Start from 1 pages to be on the safe side and guarantee 1 free HugePage
NUM_PG=1
# Cumulative number of pages required to handle the running shared memory
segments
for SEG_BYTES in `ipcs -m | awk {'print $5'} | grep "[0-9][0-9]*"`
do
    MIN_PG=`echo "$SEG_BYTES/($HPG_SZ*1024)" | bc -q`
    if [ $MIN_PG -gt 0 ]; then
        NUM_PG=`echo "$NUM_PG+$MIN_PG+1" | bc -q`
    fi
done
# Finish with results
case $KERN in
    '2.4') HUGETLB_POOL=`echo "$NUM_PG*$HPG_SZ/1024" | bc -q`;
        echo "Recommended setting: vm.hugetlb_pool = $HUGETLB_POOL" ;;
    '2.6'|'3.8') echo "Recommended setting: vm.nr_hugepages = $NUM_PG" ;;
    *) echo "Unrecognized kernel version $KERN. Exiting." ;;
esac
# End

```

- c. Run the following command to change the permission of the file:

```
$ chmod +x hugepages_settings.sh
```

7. Run the `hugepages_settings.sh` script to compute the values for hugepages configuration:

```
$ ./hugepages_settings.sh
```



Note:

Before running this script, ensure that all the applications that use hugepages run.

8. Set the following kernel parameter, where *value* is the HugePages value that you determined in step 7:

```
# sysctl -w vm.nr_hugepages=value
```

9. To ensure that HugePages is allocated after system restarts, add the following entry to the `/etc/sysctl.conf` file, where *value* is the HugePages value that you determined in step 7:

```
vm.nr_hugepages=value
```

10. Run the following command to check the available hugepages:

```
$ grep Huge /proc/meminfo
```

11. Restart the instance.

12. Run the following command to check the available hugepages (1 or 2 pages free):

```
$ grep Huge /proc/meminfo
```

 **Note:**

If you cannot set your HugePages allocation using `nr_hugepages`, then your available memory may be fragmented. Restart your server for the Hugepages allocation to take effect.

A.7.5 Restrictions for HugePages Configurations

HugePages has the following limitations:

- You must unset both the `MEMORY_TARGET` and `MEMORY_MAX_TARGET` initialization parameters. For example, to unset the parameters for the database instance, use the command `ALTER SYSTEM RESET`.
- Automatic Memory Management (AMM) and HugePages are not compatible. When you use AMM, the entire SGA memory is allocated by creating files under `/dev/shm`. When Oracle Database allocates SGA with AMM, HugePages are not reserved. To use HugePages on Oracle Database 19c, You must disable AMM.
- If you are using VLM in a 32-bit environment, then you cannot use HugePages for the Database Buffer cache. You can use HugePages for other parts of the SGA, such as `shared_pool`, `large_pool`, and so on. Memory allocation for VLM (buffer cache) is done using shared memory file systems (`ramfs/tmpfs/shmfs`). Memory file systems do not reserve or use HugePages.
- HugePages are not subject to allocation or release after system startup, unless a system administrator changes the HugePages configuration, either by modifying the number of pages available, or by modifying the pool size. If the space required is not reserved in memory during system startup, then HugePages allocation fails.
- Ensure that HugePages is configured properly as the system may run out of memory if excess HugePages is not used by the application.
- If there is insufficient HugePages when an instance starts and the initialization parameter `use_large_pages` is set to `only`, then the database fails to start and an alert log message provides the necessary information on Hugepages.

A.7.6 Disabling Transparent HugePages

Oracle recommends that you disable Transparent HugePages before you start installation.

Transparent HugePages memory differs from standard HugePages memory because the kernel `khugepaged` thread allocates memory dynamically during runtime. Standard HugePages memory is pre-allocated at startup, and does not change during runtime.

 **Note:**

Although Transparent HugePages is disabled on UEK2 and later UEK kernels, Transparent HugePages may be enabled by default on your Linux system.

Transparent HugePages memory is enabled by default with Red Hat Enterprise Linux 6, Red Hat Enterprise Linux 7, SUSE 11, Oracle Linux 6, and Oracle Linux 7 with earlier releases of Oracle Linux with the Unbreakable Enterprise Kernel 2 (UEK2) kernels.

Transparent HugePages can cause memory allocation delays during runtime. To avoid performance issues, Oracle recommends that you disable Transparent HugePages on all Oracle Database servers. Oracle recommends that you instead use standard HugePages for enhanced performance.

To check if Transparent HugePages is enabled, run one of the following commands as the root user:

Red Hat Enterprise Linux kernels:

```
# cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

Other kernels:

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
```

The following is a sample output that shows Transparent HugePages are being used as the [always] flag is enabled.

```
[always] never
```

 **Note:**

If Transparent HugePages is removed from the kernel, then neither `/sys/kernel/mm/transparent_hugepage` nor `/sys/kernel/mm/redhat_transparent_hugepage` files exist.

To disable Transparent HugePages:

1. For Oracle Linux 6 or earlier releases, add the following entry to the kernel boot line in the `/etc/grub.conf` file:

```
transparent_hugepage=never
```

For example:

```
title Oracle Linux Server (2.6.32-300.25.1.el6uek.x86_64)
  root (hd0,0)
  kernel /vmlinuz-2.6.32-300.25.1.el6uek.x86_64 ro root=LABEL=/
transparent_hugepage=never
  initrd /initramfs-2.6.32-300.25.1.el6uek.x86_64.img
```

The file name may vary for Oracle Linux 7 or later operating systems. Check your operating system documentation for the exact file name and the steps to disable Transparent HugePages.

For example, for Oracle Linux 7.3, the procedure to disable Transparent HugePages involves editing the `/etc/default/grub` file and then running the command `grub2-mkconfig`.

2. Restart the system to make the changes permanent.

B

Administering Oracle Database on Oracle Solaris

This appendix contains information about administering Oracle Database on Oracle Solaris.

It contains the following topics:

[Oracle Solaris Shared Memory Environment](#)

[About Creating Resource Pools on Oracle Solaris Systems](#)

[About Multi-CPU Binding Functionality](#)

B.1 Oracle Solaris Shared Memory Environment

This section describes how Oracle Database uses shared memory models like Optimized Shared Memory (OSM), Intimate Shared Memory (ISM), and Dynamic Intimate Shared Memory (DISM).

It contains the following topics:

- [About Optimized Shared Memory](#)
- [Checking for Optimized Shared Memory](#)
- [About ISM and DISM](#)
- [Checking for ISM or DISM](#)
- [About the oradism Utility](#)
- [How Oracle Database Decides Between OSM_ ISM and DISM](#)

B.1.1 About Optimized Shared Memory

Starting with 12c, Oracle Database uses the Optimized Shared Memory (OSM) model of Oracle Solaris on Oracle Solaris 10 1/13 or later and Oracle Solaris 11 SRU 7.5 or later systems to implement Automatic Memory Management.

OSM allows dynamic resizing of System Global Area (SGA) without restarting the instance. It does not use the `oradism` utility and swap disk space. OSM is NUMA-optimized.

B.1.2 Checking for Optimized Shared Memory



Note:

Ensure that you set the `MEMORY_MAX_TARGET` to a greater value than `MEMORY_TARGET` to utilize Optimized Shared Memory.

To verify if Oracle Solaris uses Optimized Shared Memory (OSM), enter the following command:

```
$ ipcs -dm
```

If the column `ALLOC` shows an integer, it specifies that OSM is in use. If the column `ALLOC` shows a hyphen, it specifies that OSM is not in use.

B.1.3 About ISM and DISM

On Oracle Solaris systems, Oracle Database uses Intimate Shared Memory (ISM) for shared memory segments because it shares virtual memory resources between Oracle processes. ISM causes the physical memory for the entire shared memory segment to be locked automatically.

On Oracle Solaris 10 systems prior to Oracle Solaris 10 1/13 and Oracle Solaris 11 SRU 7.5, Dynamic Intimate Shared Memory (DISM) is available. This enables Oracle Database to share virtual memory resources between processes sharing the segment, and at the same time, enables memory paging. The operating system does not have to lock down physical memory for the entire shared memory segment.

B.1.4 Checking for ISM or DISM

On Oracle Solaris, to determine if shared memory is in use, use the `ipcs -im` command. For example:

```
% ipcs -im
IPC status from <system> as of Thu Aug 19 01:09:30 PDT 2013
T ID   KEY             MODE          OWNER   GROUP   ISMATTCH
Shared Memory:
m 11  0xacea4150  --rw-rw----  oracle  dba     160
```

The `ISMATTCH` field shows 160 processes attached to this shared memory segment. However, `ISMATTCH` does not distinguish between Intimate Shared Memory (ISM) and Dynamic Intimate Shared Memory (DISM).

On Oracle Solaris 10 systems prior to Oracle Solaris 10 1/13 and Oracle Solaris 11 SRU 7.5, to identify if ISM or DISM is in use, or which memory locking service is active, use the `pmap -xs` command. For example:

```
% ps -ef | grep ora | grep smon

oracle 12524 1 0 05:40:13 ? 0:25 ora_smon_prod

% pmap -xs 12524 | grep ism
```

```
0000000380000000 38010880 38010880 - 38010880 256M rwxsR [ ism shmids=0xb ]
0000000C90000000 131072 131072 - 131072 4M rwxsR [ ism shmids=0xb ]
0000000C98000000 16 16 - 16 8K rwxsR [ ism shmids=0xb ]
```

 **Note:**

The `ps -ef` command lists the background processes that are running. This information is required to determine if an Oracle database instance is running.

The output from the `pmap -xs` command shows three `ism` address ranges implying that ISM is in use. If DISM locks the memory ranges, then the output shows `dism` address ranges.

B.1.5 About the oradism Utility

Oracle Database uses the `oradism` utility to lock and unlock shared memory. The `oradism` utility is automatically set up during installation. It is not required to perform any configuration tasks to use dynamic SGA.

The process name for the `oradism` utility is `ora_dism_sid`, where `sid` is the system identifier. When using DISM, this process is started during instance startup, and automatically quits when the instance is shut down.

If a message is displayed in the alert log saying that the `oradism` utility is not set up correctly, then verify that the `oradism` utility is located in the `$ORACLE_HOME/bin` directory and that it has superuser privileges.

 **Note:**

Optimized Shared Memory (OSM) does not use the `oradism` utility.

B.1.6 How Oracle Database Decides Between OSM, ISM and DISM

Oracle Database automatically uses Optimized Shared Memory (OSM) on Oracle Solaris systems where OSM is available. See "[About Optimized Shared Memory](#)" for more information on OSM.

On systems where OSM is not available, Oracle Database automatically selects Intimate Shared Memory (ISM) or Dynamic Intimate Shared Memory (DISM) based on the following criteria:

- Oracle Database uses DISM if it is available on the system, and if the value of the `SGA_MAX_SIZE` initialization parameter is larger than the size required for all SGA components combined. This enables Oracle Database to lock only the amount of physical memory that is used.
- Oracle Database uses ISM if the entire shared memory segment is in use at startup or if the value of the `SGA_MAX_SIZE` parameter equals or smaller than the size required for all SGA components combined.

Regardless of whether Oracle Database uses ISM or DISM, it can always exchange the memory between dynamically sizable components such as the buffer cache, the shared pool, and the large pool after it starts an instance. Oracle Database can relinquish memory from one dynamic SGA component and allocate it to another component.

Because shared memory segments are not implicitly locked in memory, when using DISM, Oracle Database explicitly locks shared memory that is currently in use at startup. When a dynamic SGA operation uses more shared memory, Oracle Database explicitly performs a lock operation on the memory that is put to use. When a dynamic SGA operation releases shared memory, Oracle Database explicitly performs an unlock operation on the memory that is freed, so that it becomes available to other applications.

 **Note:**

Do not set the `LOCK_SGA` parameter to `TRUE` in the server parameter file. If you do, then Oracle Database 12c cannot start.

B.2 About Creating Solaris Resource Pools

Solaris Resource Pools improve database performance by associating a dedicated set of CPUs to a database instance. Each database instance can only use the resources in its resource pool.

When consolidating on a large server, you may want to restrict the database to a specific subset of the CPU and memory. This feature makes it easy to enable CPU and memory restrictions for an Oracle Database instance.

Use the `setup_resource_pool.sh` script to create Solaris Resource Pools. Download this script from note 1928328.1 on the My Oracle Support website:

<https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1928328.1>

B.3 About Multi-CPU Binding Functionality

You can use Multi-CPU Binding (MCB) as part of your resource management policy to improve performance.

Multi-CPU binding (MCB) is an Oracle Solaris projects resource management functionality that binds a project to a specific set of CPUs, but does not bind the CPUs exclusively. MCB allows other processes also to use these CPUs, and allows overlapping of partitions. MCB is supported on Oracle Solaris 11.3 and later.

The resource pools feature also allows binding of CPUs. However, this method requires hard partitioning of processors in the system. Resource pools does not allow overlapping of partitions.

You can assign, modify, or remove MCBs through Oracle Solaris projects. Use the standard command-line tools `projadd(1M)` and `projmod(1M)` to create or modify the project file.

For database users, the advantages of using MCB over resource pools include:

- The ability to bind an Oracle instance to a particular NUMA location for performance, such as, near a specific I/O or networking device.
- The ability to bind multiple Oracle instances to different cores or sockets on a host to increase performance isolation without the need of a privileged administrator to partition the system.

 **See Also:**

"Using Projects to Assign, Modify, and Remove Multi-CPU Binding" and "How to Use Projects to Assign, Modify, and Remove Multi-CPU Binding" in *Administering Resource Management in Oracle Solaris 11.3*.

C

Using Oracle ODBC Driver

This appendix provides information related to using Oracle ODBC Driver.

It contains the following sections:

- [Oracle ODBC Features Not Supported](#)
- [Implementation of Data Types](#)
- [Limitations on Data Types](#)
- [Format of the Connection String for the SQLDriverConnect Function](#)
- [Reducing Lock Timeout in a Program](#)
- [Linking ODBC Applications](#)
- [Obtaining Information About ROWIDs](#)
- [ROWIDs in a WHERE Clause](#)
- [Enabling Result Sets](#)
- [Enabling EXEC Syntax](#)
- [Supported Functionality](#)
- [Unicode Support](#)
- [Performance and Tuning](#)
- [Error Messages](#)

See Also:

Oracle Database Installation Guide for your respective platform for the ODBC driver certification information

C.1 Oracle ODBC Features Not Supported

Oracle ODBC Driver does not support the following Oracle ODBC 3.0 features:

- Interval data types
- `SQL_C_UBIGINT` and `SQL_C_SBIGINT` C data type identifiers
- Shared connections
- Shared environments
- The `SQL_LOGIN_TIMEOUT` attribute of `SQLSetConnectAttr`
- The expired password option

Oracle ODBC Driver does not support the SQL functions listed in the following table:

String Functions	Numeric Functions	Time, Date, and Interval Functions
BIT_LENGTH	ACOS	CURRENT_DATE
CHAR_LENGTH	ASIN	CURRENT_TIME
CHARACTER_LENGTH	ATAN	CURRENT_TIMESTAMP
DIFFERENCE	ATAN2	EXTRACT
OCTET_LENGTH	COT	TIMESTAMPDIFF
POSITION	DEGREES	
	RADIANS	
	RAND	
	ROUND	

C.2 Implementation of Data Types

This section discusses the `DATE`, `TIMESTAMP`, and floating point data types.

DATE and TIMESTAMP

The semantics of Oracle `DATE` and `TIMESTAMP` data types do not correspond exactly with the ODBC data types with the same names. The Oracle `DATE` data type contains both date and time information. The `SQL_DATE` data type contains only date information. The Oracle `TIMESTAMP` data type also contains date and time information, but it has greater precision in fractional seconds. Oracle ODBC Driver reports the data types of both Oracle `DATE` and `TIMESTAMP` columns as `SQL_TIMESTAMP` to prevent information loss. Similarly, Oracle ODBC Driver binds `SQL_TIMESTAMP` parameters as Oracle `TIMESTAMP` values.



See Also:

"[DATE and TIMESTAMP Data Types](#)" for information about `DATE` and `TIMESTAMP` data types related to performance and tuning

Floating Point Data Types

When connected to an Oracle Database 12c Release 2 (12.2) or later, Oracle ODBC Driver maps the Oracle floating point data types `BINARY_FLOAT` and `BINARY_DOUBLE` to the ODBC data types `SQL_REAL` and `SQL_DOUBLE`, respectively. In earlier releases, `SQL_REAL` and `SQL_DOUBLE` mapped to the generic Oracle numeric data type.

C.3 Limitations on Data Types

Oracle ODBC Driver and Oracle Database impose limitations on data types. The following table describes these limitations:

Limited Data Type	Description
Literals	Oracle Database limits literals in SQL statements to 4000 bytes.

Limited Data Type	Description
<code>SQL_LONGVARCHAR</code> and <code>SQL_WLONGVARCHAR</code>	The Oracle limit for <code>SQL_LONGVARCHAR</code> data, where the column type is <code>LONG</code> , is 2,147,483,647 bytes. The Oracle limit for <code>SQL_LONGVARCHAR</code> data, where the column type is <code>CLOB</code> , is 4 gigabytes. The limiting factor is the client workstation memory.
<code>SQL_LONGVARCHAR</code> and <code>SQL_LONGVARBINARY</code>	Oracle Database permits only a single long data column in each table. The long data types are <code>SQL_LONGVARCHAR</code> (<code>LONG</code>) and <code>SQL_LONGVARBINARY</code> (<code>LONG RAW</code>). Oracle recommends that you use <code>CLOB</code> and <code>BLOB</code> columns instead. There is no restriction on the number of <code>CLOB</code> and <code>BLOB</code> columns in a table.

C.4 Format of the Connection String for the `SQLDriverConnect` Function

The `SQLDriverConnect` function is one of the functions implemented by Oracle ODBC Driver. The following table describes the keywords that you can include in the connection string argument of the `SQLDriverConnect` function call:

Keyword	Meaning	Value
<code>DSN</code>	ODBC data source name	User-supplied name This is a mandatory keyword.
<code>DBQ</code>	TNS service name	User-supplied name This is a mandatory keyword.
<code>UID</code>	User ID or user name	User-supplied name This is a mandatory keyword.
<code>PWD</code>	Password	User-supplied name Specify <code>PWD=;</code> for an empty password. This is a mandatory keyword.
<code>DBA</code>	Database attribute	<code>W</code> implies write access <code>R</code> implies read-only access
<code>APA</code>	Applications attributes	<code>T</code> implies that thread safety is to be enabled. <code>F</code> implies that thread safety is to be disabled.
<code>RST</code>	Result sets	<code>T</code> implies that result sets are to be enabled. <code>F</code> implies that result sets are to be disabled.
<code>QTO</code>	Query timeout option	<code>T</code> implies that query timeout is to be enabled. <code>F</code> implies that query timeout is to be disabled.

Keyword	Meaning	Value
CSR	Close cursor	T implies that close cursor is to be enabled. F implies that close cursor is to be disabled.
BAM	Batch autocommit mode	IfAllSuccessful implies commit only if all statements are successful (old behavior). UpToFirstFailure implies commit up to first failing statement. This is ODBC version 7 behavior. AllSuccessful implies commit all successful statements.
FBS	Fetch buffer size	User-supplied numeric value (specify a value in bytes of 0 or greater). The default is 60,000 bytes.
FEN	Failover	T implies failover is to be enabled. F implies failover is to be disabled.
FRC	Failover retry count	User-supplied numeric value. The default is 10.
FDL	Failover delay	User-supplied numeric value. The default is 10.
LOB	LOB writes	T implies LOBs are to be enabled. F implies LOBs are to be disabled.
FWC	Force <code>SQL_WCHAR</code> support	T implies <code>Force SQL_WCHAR</code> is to be enabled. F implies <code>Force SQL_WCHAR</code> is to be disabled.
EXC	EXEC syntax	T implies EXEC syntax is to be enabled. F implies EXEC syntax is to be disabled.
XSM	Schema field	Default implies that the default value is to be used. Database implies that the database name is to be used. Owner implies that the name of the owner is to be used.

Keyword	Meaning	Value
MDI	Set metadata ID default	T implies that the default value of <code>SQL_ATTR_METADATA_ID</code> is <code>SQL_TRUE</code> . F implies that the default value of <code>SQL_ATTR_METADATA_ID</code> is <code>SQL_FALSE</code> .
DPM	Disable <code>SQLDescribeParam</code>	T implies that <code>SQLDescribeParam</code> is to be disabled. F implies that <code>SQLDescribeParam</code> is to be enabled.
BTD	Bind <code>TIMESTAMP</code> as <code>DATE</code>	T implies that <code>SQL_TIMESTAMP</code> is to be bound as Oracle <code>DATE</code> . F implies that <code>SQL_TIMESTAMP</code> is to be bound as Oracle <code>TIMESTAMP</code> .
NUM	Numeric settings	NLS implies that the Globalization Support numeric settings are to be used (to determine the decimal and group separator).

C.5 Reducing Lock Timeout in a Program

Oracle Database waits indefinitely for lock conflicts between transactions to be resolved. However, you can limit the amount of time that Oracle Database waits for locks to be resolved. To do this, set the `SQL_ATTR_QUERY_TIMEOUT` attribute of the ODBC `SQLSetStmtAttr` function while calling this function before connecting to the data source.

C.6 Linking ODBC Applications

When you link the program, you must link it with the Driver Manager library, `libodbc.so`.

C.7 Obtaining Information About ROWIDs

The ODBC `SQLSpecialColumns` function returns information about the columns in a table. When used with Oracle ODBC Driver, it returns information about the Oracle ROWIDs associated with an Oracle table.

C.8 ROWIDs in a WHERE Clause

ROWIDs may be used in the `WHERE` clause of an SQL statement. However, the ROWID value must be presented in a parameter marker.

C.9 Enabling Result Sets

Oracle reference cursors, also known as result sets, enable an application to retrieve data using stored procedures and stored functions. The following information describes how to use reference cursors to enable result sets through ODBC:

- You must use the ODBC syntax for calling stored procedures. Native PL/SQL is not supported through ODBC. The following code sample identifies how to call the procedure or function without a package and within a package. The package name in this case is RSET.

```
Procedure call:
{CALL Example1(?)}
{CALL RSET.Example1(?)}
Function Call:
{? = CALL Example1(?)}
{? = CALL RSET.Example1(?)}
```

- The PL/SQL reference cursor parameters are omitted when calling the procedure. For example, assume procedure `Example2` is defined to have four parameters. Parameters 1 and 3 are reference cursor parameters and parameters 2 and 4 are character strings. The call is specified as:

```
{CALL RSET.Example2("Literal 1", "Literal 2")}
```

The following sample application shows how to return a result set by using Oracle ODBC Driver:

```
/*
 * Sample Application using Oracle reference cursors through ODBC
 *
 * Assumptions:
 *
 * 1) Oracle Sample database is present with data loaded for the EMP table.
 *
 * 2) Two fields are referenced from the EMP table, ename and mgr.
 *
 * 3) A data source has been setup to access the sample database.
 *
 * Program Description:
 *
 * Abstract:
 *
 * This program demonstrates how to return result sets using
 * Oracle stored procedures
 *
 * Details:
 *
 * This program:
 * Creates an ODBC connection to the database.
 * Creates a Packaged Procedure containing two result sets.
 * Executes the procedure and retrieves the data from both result sets.
 * Displays the data to the user.
 * Deletes the package then logs the user out of the database.
 *
 * The following is the actual PL/SQL this code generates to
 * create the stored procedures.
 */
```

```
DROP PACKAGE ODBCRefCur;
CREATE PACKAGE ODBCRefCur AS
    TYPE ename_cur IS REF CURSOR;
    TYPE mgr_cur IS REF CURSOR;
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2);

END;

/

CREATE PACKAGE BODY ODBCRefCur AS
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2)
AS
    BEGIN
        IF NOT Ename%ISOPEN
        THEN
            OPEN Ename for SELECT ename from emp;
        END IF;

        IF NOT Mgr%ISOPEN
        THEN
            OPEN Mgr for SELECT mgr from emp where job = pjob;
        END IF;
    END;
END;

/

*
* End PL/SQL for Reference Cursor.
*/

/*
* Include Files
*/
#include <stdio.h>
#include <sql.h>
#include <sqlext.h>
/*
* Defines
*/
#define JOB_LEN 9
#define DATA_LEN 100
#define SQL_STMT_LEN 500
/*
* Procedures
*/
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module );
/*
* Main Program
*/
int main()
{
    SQLHENV hEnv;
    SQLHDBC hDbc;
    SQLHSTMT hStmt;
    SQLRETURN rc;
    char *DefUserName ="jones";
    char *DefPassWord ="password";
    SQLCHAR ServerName[DATA_LEN];
    SQLCHAR *pServerName=ServerName;
    SQLCHAR UserName[DATA_LEN];
```

```
SQLCHAR *pUserName=UserName;
SQLCHAR PassWord[DATA_LEN];
SQLCHAR *pPassWord=PassWord;
char Data[DATA_LEN];
SQLINTEGER DataLen;
char error[DATA_LEN];
char *charptr;
SQLCHAR SqlStmt[SQL_STMT_LEN];
SQLCHAR *pSqlStmt=SqlStmt;
char *pSalesMan = "SALESMAN";
SQLINTEGER sqlnts=SQL_NTS;
/*
 * Allocate the Environment Handle
 */
rc = SQLAllocHandle( SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv );
if (rc != SQL_SUCCESS)
{
    printf( "Cannot Allocate Environment Handle\n");
    printf( "\nHit Return to Exit\n");
    charptr = gets ((char *)error);
    exit(1);
}
/*
 * Set the ODBC Version
 */
rc = SQLSetEnvAttr( hEnv,SQL_ATTR_ODBC_VERSION,(void *)SQL_OV_ODBC3,0);
if (rc != SQL_SUCCESS)
{
    printf( "Cannot Set ODBC Version\n");
    printf( "\nHit Return to Exit\n");
    charptr = gets ((char *)error);
    exit(1);
}
/*
 * Allocate the Connection handle
 */
rc = SQLAllocHandle( SQL_HANDLE_DBC, hEnv, &hDbc );
if (rc != SQL_SUCCESS)
{
    printf( "Cannot Allocate Connection Handle\n");
    printf( "\nHit Return to Exit\n");
    charptr = gets ((char *)error);
    exit(1);
}
/*
 * Get User Information
 */
strcpy ((char *) pUserName, DefUserName );
strcpy ((char *) pPassWord, DefPassWord );
/*
 * Data Source name
 */
printf( "\nEnter the ODBC Data Source Name\n" );
charptr = gets ((char *) ServerName);
/*
 * User Name
 */
printf ( "\nEnter User Name Default [%s]\n", pUserName);
charptr = gets ((char *) UserName);
if (*charptr == '\0')
{
```

```

    strcpy ((char *) pUserName, (char *) DefUserName );
}
/*
 * Password
 */
printf ( "\nEnter Password Default [%s]\n", pPassWord);
charptr = gets ((char *)PassWord);
if (*charptr == '\0')
{
    strcpy ((char *) pPassWord, (char *) DefPassWord );
}
/*
 * Connection to the database
 */
rc = SQLConnect( hDbc,pServerName,(SQLSMALLINT) strlen((char
*)pServerName),pUserName,(SQLSMALLINT) strlen((char *)pUserName),pPassWord,
(SQLSMALLINT) strlen((char *)pPassWord));
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_DBC, hDbc, "SQLConnect");
}
/*
 * Allocate a Statement
 */
rc = SQLAllocHandle( SQL_HANDLE_STMT, hDbc, &hStmt );
if (rc != SQL_SUCCESS)
{
    printf( "Cannot Allocate Statement Handle\n");
    printf( "\nHit Return to Exit\n");
    charptr = gets ((char *)error);
    exit(1);
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Create the Package Header
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE ODBCRefCur AS\n");
strcat( (char *) pSqlStmt, " TYPE ename_cur IS REF CURSOR;\n");
strcat( (char *) pSqlStmt, " TYPE mgr_cur IS REF CURSOR;\n\n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur,pjob IN VARCHAR2);\n\n");
strcat( (char *) pSqlStmt, "END;\n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Create the Package Body
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE BODY ODBCRefCur AS\n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur, pjob IN VARCHAR2)\n AS\n BEGIN\n");
strcat( (char *) pSqlStmt, " IF NOT Ename%ISOPEN\n THEN\n");
strcat( (char *) pSqlStmt, " OPEN Ename for SELECT ename from emp;\n");
strcat( (char *) pSqlStmt, " END IF;\n\n");
strcat( (char *) pSqlStmt, " IF NOT Mgr%ISOPEN\n THEN\n");

```

```
strcat( (char *) pSqlStmt, " OPEN Mgr for SELECT mgr from emp where job = pjob;\n");
strcat( (char *) pSqlStmt, " END IF;\n");
strcat( (char *) pSqlStmt, " END;\n");
strcat( (char *) pSqlStmt, "END;\n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Parameter
 */
rc = SQLBindParameter(hStmt,1,SQL_PARAM_INPUT,SQL_C_CHAR,SQL_CHAR,JOB_LEN,
0,pSalesMan,0,&sqlnts);
/*
 * Call the Store Procedure which executes the Result Sets
 */
strcpy( (char *) pSqlStmt, "{CALL ODBCRefCur.EmpCurs(?)}");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Data
 */
rc = SQLBindCol( hStmt,1,SQL_C_CHAR,Data,sizeof(Data),&DataLen);
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLBindCol");
}
/*
 * Get the data for Result Set 1
 */
printf( "\nEmployee Names\n\n");
while ( rc == SQL_SUCCESS )
{
    rc = SQLFetch( hStmt );
    if ( rc == SQL_SUCCESS )
    {
        printf("%s\n", Data);
    }
    else
    {
        if (rc != SQL_NO_DATA)
        {
            DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
        }
    }
}
printf( "\nFirst Result Set - Hit Return to Continue\n");
charptr = gets ((char *)error);
/*
 * Get the Next Result Set
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_SUCCESS)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
```

```

    * Get the data for Result Set 2
    */
printf( "\nManagers\n\n");
while ( rc == SQL_SUCCESS )
{
    rc = SQLFetch( hStmt );
    if ( rc == SQL_SUCCESS )
    {
        printf("%s\n", Data);
    }
    else
    {
        if (rc != SQL_NO_DATA)
        {
            DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
        }
    }
}
printf( "\nSecond Result Set - Hit Return to Continue\n");
charptr = gets ((char *)error);
/*
 * Should Be No More Results Sets
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_NO_DATA)
{
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Free handles close connections to the database
 */
SQLFreeHandle( SQL_HANDLE_STMT, hStmt );
SQLDisconnect( hDbc );
SQLFreeHandle( SQL_HANDLE_DBC, hDbc );
SQLFreeHandle( SQL_HANDLE_ENV, hEnv );
printf( "\nAll Done - Hit Return to Exit\n");
charptr = gets ((char *)error);
return(0);
}
/*
 * Display Error Messages
 */
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module )
{
    SQLCHAR MessageText[255];
    SQLCHAR SQLState[80];
    SQLRETURN rc=SQL_SUCCESS;
    long NativeError;
    SWORD RetLen;
    SQLCHAR error[25];
    char *charptr;
    rc =
    SQLGetDiagRec(HandleType,hHandle,1,SQLState,&NativeError,MessageText,255,&RetLen);
    printf( "Failure Calling %s\n", Module );
    if (rc == SQL_SUCCESS || rc == SQL_SUCCESS_WITH_INFO)
    {

```

```
printf( "\t\t\t State: %s\n", SQLState);  
printf( "\t\t\t Native Error: %d\n", NativeError );  
printf( "\t\t\t Error Message: %s\n", MessageText );  
}  
printf( "\nHit Return to Exit\n");  
charptr = gets ((char *)error);  
exit(1);  
}
```

C.10 Enabling EXEC Syntax

If the syntax of the SQL Server `EXEC` statement can be readily translated to an equivalent Oracle procedure call without requiring any change to it, then Oracle ODBC Driver can translate it if you enable this option.

The complete name of a SQL Server procedure consists of up to four identifiers:

- Server name
- Database name
- Owner name
- Procedure name

The format for the name is:

```
[[[server.][database.][owner_name].]procedure_name
```

During the migration of Microsoft SQL Server database to Oracle Database, the definition of each SQL Server procedure or function is converted to its equivalent Oracle Database syntax and is defined in a schema in Oracle Database. Migrated procedures are often reorganized (and created in schemas) in one of the following ways:

- All procedures are migrated to one schema (the default option).
- All procedures defined in one SQL Server database are migrated to the schema named with that database name.
- All procedures owned by one user are migrated to the schema named with that user's name.

To support these three ways of organizing migrated procedures, you can specify one of these schema name options for translating procedure names. Object names in the translated Oracle procedure call are not case-sensitive.

C.11 Supported Functionality

This section provides information about the functionality supported by Oracle ODBC Driver. It contains the following sections:

- [API Conformance](#)
- [Implementation of ODBC API Functions](#)
- [Implementation of the ODBC SQL Syntax](#)
- [Implementation of Data Types](#)

C.11.1 API Conformance

Oracle ODBC Driver release 10.2.0.1.0 and higher supports all Core, Level 2, and Level 1 functions.

C.11.2 Implementation of ODBC API Functions

The following table describes how Oracle ODBC Driver implements specific functions:

Function	Description
SQLConnect	SQLConnect requires only a DBQ, user ID, and password.
SQLDriverConnect	SQLDriverConnect uses the DSN, DBQ, UID, and PWD keywords.
SQLSpecialColumns	If SQLSpecialColumns is called with the SQL_BEST_ROWID attribute, then it always returns the ROWID column.
SQLProcedures and SQLProcedureColumns	Refer to the information in the following row.
All catalog functions	If the SQL_ATTR_METADATA_ID statement attribute is set to SQL_TRUE, then a string argument is treated as an identifier argument, and its case is not significant. In this case, the underscore (_) and the percent sign (%) are treated as the actual character, and not as a search pattern character. In contrast, if this attribute is set to SQL_FALSE, then it is either an ordinary argument or a pattern value argument and is treated literally, and its case is significant.

SQLProcedures and SQLProcedureColumns

The SQLProcedures and SQLProcedureColumns calls have been modified to locate and return information about all procedures and functions even if they are contained within a package. In earlier releases, the calls only found procedures and functions that were outside of packages. The following examples and scenarios show what procedures or functions are returned if the SQL_ATTR_METADATA_ID attribute is set to SQL_FALSE.

Suppose that you have the following stored procedures:

```
"BAR"
"BARX"
"XBAR"
"XBARX"
"SQLPROCTEST.BAR"
"SQLPROCTEST.BARX"
"SQLPROCTEST.XBAR"
"SQLPROCTEST.XBARX"
```

When you look for % or %%%%, you get all eight procedures.

When you look for %_ or _%, you get the following:

```
BAR
BARX
XBAR
XBARX
```

When you look for . or .% or %.% or SQLPROC%. or SQLPROC%.%, you get the following:

```
SQLPROCTEST.BAR
SQLPROCTEST.BARX
SQLPROCTEST.XBAR
SQLPROCTEST.XBARX
```

When you look for %BAR, you get the following:

```
BAR
XBAR
```

When you look for .%BAR or %.%BAR, you get the following:

```
SQLPROCTEST.BAR
SQLPROCTEST.XBAR
```

When you look for SQLPROC% or .SQLPROC%, you get the following:

```
nothing (0 rows)
```

C.11.3 Implementation of the ODBC SQL Syntax

If a comparison predicate has a parameter marker as the second expression in the comparison and the value of that parameter is set to `SQL_NULL_DATA` with `SQLBindParameter`, then the comparison fails. This is consistent with the null predicate syntax in ODBC SQL.

C.11.4 Implementation of Data Types

For programmers, the most important part of the implementation of the data types concerns the `CHAR`, `VARCHAR`, and `VARCHAR2` data types.

For an `fSqlType` value of `SQL_VARCHAR`, `SQLGetTypeInfo` returns the Oracle Database data type `VARCHAR2`. For an `fSqlType` value of `SQL_CHAR`, `SQLGetTypeInfo` returns the Oracle Database data type `CHAR`.

C.12 Unicode Support

This section provides information about Unicode support. It contains the following topics:

- [Unicode Support Within the ODBC Environment](#)
- [Unicode Support in ODBC API](#)
- [SQLGetData Performance](#)
- [Unicode Samples](#)

C.12.1 Unicode Support Within the ODBC Environment

ODBC Driver Manager makes all ODBC drivers, regardless of whether they support Unicode, appear as if they are Unicode compliant. This allows ODBC applications to be written independent of the Unicode capabilities of underlying ODBC drivers.

The extent to which the Driver Manager can emulate Unicode support for ANSI ODBC drivers is limited by the conversions possible between the Unicode data and the local code page. Data loss is possible when the Driver Manager is converting from Unicode

to the local code page. Full Unicode support is not possible unless the underlying ODBC driver supports Unicode. Oracle ODBC Driver provides full Unicode support.

C.12.2 Unicode Support in ODBC API

The ODBC API supports both Unicode and ANSI entry points using the `w` and `a` suffix convention. An ODBC application developer does not explicitly call entry points with the suffix. An ODBC application that is compiled with the `UNICODE` and `_UNICODE` preprocessor definitions generates the appropriate calls. For example, a call to `SQLPrepare` compiles as `SQLPrepareW`.

The C data type, `SQL_C_WCHAR`, was added to the ODBC interface to allow applications to specify that an input parameter is encoded as Unicode or to request column data returned as Unicode. The macro `SQL_C_TCHAR` is useful for applications that must be built as both Unicode and ANSI. The `SQL_C_TCHAR` macro compiles as `SQL_C_WCHAR` for Unicode applications and as `SQL_C_CHAR` for ANSI applications.

The SQL data types, `SQL_WCHAR`, `SQL_WVARCHAR`, and `SQL_WLONGVARCHAR`, have been added to the ODBC interface to represent columns defined in a table as Unicode. Potentially, these values are returned from calls to `SQLDescribeCol`, `SQLColAttribute`, `SQLColumns`, and `SQLProcedureColumns`.

Unicode encoding is supported for SQL column types `NCHAR`, `NVARCHAR2`, and `NCLOB`. In addition, Unicode encoding is also supported for SQL column types `CHAR` and `VARCHAR2` if the character semantics are specified in the column definition.

Oracle ODBC Driver supports these SQL column types and maps them to ODBC SQL data types. The following table lists the supported SQL data types and the equivalent ODBC SQL data type:

SQL Data Types	ODBC SQL Data Types
CHAR	SQL_CHAR or SQL_WCHAR
VARCHAR2	SQL_VARCHAR or SQL_WVARCHAR
NCHAR	SQL_WCHAR
NVARCHAR2	SQL_WVARCHAR
NCLOB	SQL_WLONGVARCHAR

C.12.3 SQLGetData Performance

The `SQLGetData` function allows an ODBC application to specify the data type to receive a column after the data has been fetched. OCI requires Oracle ODBC Driver to specify the data type before it is fetched. In this case, Oracle ODBC Driver uses information about the data type of the column (as defined in the database) to determine how to best default to fetching the column through OCI.

If a column that contains character data is not bound by `SQLBindCol`, then Oracle ODBC Driver must determine if it should fetch the column as Unicode or as the local code page. The driver could always default to receiving the column as Unicode. However, this may result in as many as two unnecessary conversions. For example, if the data were encoded in the database as ANSI, then there would be an ANSI to Unicode conversion to fetch the data into Oracle ODBC Driver. If the ODBC

application then requested the data as `SQL_C_CHAR`, then there would be an additional conversion to revert the data to its original encoding.

The default encoding of Oracle Database Client is used when fetching data. However, an ODBC application may overwrite this default and fetch the data as Unicode by binding the column or the parameter as the `WCHAR` data type.

C.12.4 Unicode Samples

Because Oracle ODBC Driver itself was implemented using `TCHAR` macros, it is recommended that ODBC application programs use `TCHAR` to take advantage of the driver.

The following examples show how to use `TCHAR`, which becomes the `WCHAR` data type if you compile with `UNICODE` and `_UNICODE`:

Example C-1 Connection to Database

To use this code, you only must specify the Unicode literals for `SQLConnect`.

```
HENV          envHnd;
HDBC          conHnd;
HSTMT        stmtHnd;
RETCODE      rc;

rc = SQL_SUCCESS;

// ENV is allocated
rc = SQLAllocEnv(&envHnd);
// Connection Handle is allocated
rc = SQLAllocConnect(envHnd, &conHnd);
rc = SQLConnect(conHnd, _T("stpc19"), SQL_NTS, _T("jones"), SQL_NTS, _T("password"),
SQL_NTS);
.
.
.
if (conHnd)
    SQLFreeConnect(conHnd);
if (envHnd)
    SQLFreeEnv(envHnd);
```

Example C-2 Simple Retrieval

The following example retrieves the employee names and the job titles from the `EMP` table. With the exception that you must specify `TCHAR` compliant data to every ODBC function, there is no difference to the ANSI case. If the case is a Unicode application, then you must specify the length of the buffer to the `BYTE` length when you call `SQLBindCol`. For example, `sizeof(ename)`.

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
**   SQLBindCol
**   SQLFetch
**
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename, job FROM emp");
SQLTCHAR  ename[50];
```

```

SQLTCHAR    job[50];
SQLINTEGER  enamelen, joblen;

_tprintf(_T("Retrieve ENAME and JOB using SQLBindCol 1.../n[%s]/n"), sqlStmt);

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Columns
rc = SQLBindCol(stmtHnd,
                1,
                SQL_C_TCHAR,
                ename,
                sizeof(ename),
                &enamelen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindCol(stmtHnd,
                2,
                SQL_C_TCHAR,
                job,
                sizeof(job),
                &joblen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{
    // Step 3: Fetch Data
    rc = SQLFetch(stmtHnd);
    if (rc == SQL_NO_DATA)
        break;
    checkSQLErr(envHnd, conHnd, stmtHnd, rc);
    _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));

```

Example C-3 Retrieval Using SQLGetData (Binding After Fetch)

This example shows how to use SQLGetData. There is no difference to the ANSI application in terms of Unicode-specific issues.

```

/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
**   SQLFetch
**   SQLGetData
**
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename,job FROM emp"); // same as Case 1.
SQLTCHAR        ename[50];
SQLTCHAR        job[50];

_tprintf(_T("Retrieve ENAME and JOB using SQLGetData.../n[%s]/n"), sqlStmt);
if (rc != SQL_SUCCESS)
{
    _tprintf(_T("Failed to allocate STMT/n"));
    goto exit2;
}

```

```

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{
    // Step 2: Fetch
    rc = SQLFetch(stmtHnd);
    if (rc == SQL_NO_DATA)
        break;
    checkSQLErr(envHnd, conHnd, stmtHnd, rc);

    // Step 3: GetData
    rc = SQLGetData(stmtHnd,
        1,
        SQL_C_TCHAR,
        (SQLPOINTER)ename,
        sizeof(ename),
        NULL);
    checkSQLErr(envHnd, conHnd, stmtHnd, rc);
    rc = SQLGetData(stmtHnd,
        2,
        SQL_C_TCHAR,
        (SQLPOINTER)job,
        sizeof(job),
        NULL);
    checkSQLErr(envHnd, conHnd, stmtHnd, rc);
    _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));

```

Example C-4 Simple Update

This example shows how to update data. The length of data for `SQLBindParameter` has to be specified with the BYTE length, even in Unicode application.

```

/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
*/
static SQLTCHAR *sqlStmt = _T("INSERT INTO emp(empno,ename,job) VALUES(?,?,?)");
static SQLTCHAR *empno   = _T("9876"); // Emp No
static SQLTCHAR *ename   = _T("ORACLE"); // Name
static SQLTCHAR *job     = _T("PRESIDENT"); // Job

_tprintf(_T("Insert User ORACLE using SQLBindParameter.../n[%s]/n"), sqlStmt);

// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Parameter
rc = SQLBindParameter(stmtHnd,
    1,
    SQL_PARAM_INPUT,
    SQL_C_TCHAR,

```

```

        SQL_DECIMAL,
        4,           // 4 digit
        0,
        (SQLPOINTER)empno,
        0,
        NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
    2,
    SQL_PARAM_INPUT,
    SQL_C_TCHAR,
    SQL_CHAR,
    lstrlen(ename)*sizeof(TCHAR),
    0,
    (SQLPOINTER)ename,
    lstrlen(ename)*sizeof(TCHAR),
    NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
    3,
    SQL_PARAM_INPUT,
    SQL_C_TCHAR,
    SQL_CHAR,
    lstrlen(job)*sizeof(TCHAR),
    0,
    (SQLPOINTER)job,
    lstrlen(job)*sizeof(TCHAR),
    NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

```

Example C-5 Update and Retrieval for Long Data (CLOB)

This example may be the most complicated case to update and retrieve data for long data, like CLOB, in Oracle Database. Because the length of data should always be the BYTE length, the expression `lstrlen(TCHAR data)*sizeof(TCHAR)` is needed to derive the BYTE length.

```

/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
**   SQLParamData
**   SQLPutData
**
**   SQLExecDirect
**   SQLFetch
**   SQLGetData
*/
static SQLTCHAR *sqlStmt1 = _T("INSERT INTO clobtbl(clob1) VALUES(?)");
static SQLTCHAR *sqlStmt2 = _T("SELECT clob1 FROM clobtbl");
SQLTCHAR      clobdata[1001];
SQLTCHAR      resultdata[1001];

```

```
SQLINTEGER      ind = SQL_DATA_AT_EXEC;
SQLTCHAR       *bufp;
int            clobdatalen, chunksize, dtsize, retchklen;

_tprintf(_T("Insert CLOB1 using SQLPutData.../n[%s]/n"), sqlStmt1);

// Set CLOB Data
{
    int i;
    SQLTCHAR ch;
    for (i=0, ch=_T('A'); i < sizeof(clobdata)/sizeof(SQLTCHAR); ++i, ++ch)
    {
        if (ch > _T('Z'))
            ch = _T('A');
        clobdata[i] = ch;
    }
    clobdata[sizeof(clobdata)/sizeof(SQLTCHAR)-1] = _T('/0');
}
clobdatalen = strlen(clobdata); // length of characters
chunksize = clobdatalen / 7; // 7 times to put

// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt1, SQL_NTS);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
// Step 2: Bind Parameter with SQL_DATA_AT_EXEC
rc = SQLBindParameter(stmtHnd,
                      1,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_LONGVARCHAR,
                      clobdatalen*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)clobdata,
                      clobdatalen*sizeof(TCHAR),
                      &ind);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 4: ParamData (initiation)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp); // set value
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

for (dtsize=0, bufp = clobdata;
     dtsize < clobdatalen;
     dtsize += chunksize, bufp += chunksize)
{
    int len;
    if (dtsize+chunksize<clobdatalen)
        len = chunksize;
    else
        len = clobdatalen-dtsize;

    // Step 5: PutData
    rc = SQLPutData(stmtHnd, (SQLPOINTER)bufp, len*sizeof(TCHAR));
    checkSQLErr(envHnd, conHnd, stmtHnd, rc);
}

// Step 6: ParamData (termination)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp);
```



```

checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLFreeStmt(stmtHnd, SQL_CLOSE);
_tprintf(_T("Finished Update/n/n"));
rc = SQLAllocStmt(conHnd, &stmtHnd);
if (rc != SQL_SUCCESS)
{
    _tprintf(_T("Failed to allocate STMT/n/n"));
    goto exit2;
}

// Clear Result Data
memset(resultdata, 0, sizeof(resultdata));
chunksize = clobdatalen / 15; // 15 times to put

// Step 1: Prepare
rc = SQLExecDirect(stmtHnd, sqlStmt2, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Fetch
rc = SQLFetch(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
for(dtsize=0, bufp = resultdata;
    dtsize < sizeof(resultdata)/sizeof(TCHAR) && rc != SQL_NO_DATA;
    dtsize += chunksize-1, bufp += chunksize-1)
{
    int len; // len should contain the space for NULL termination
    if (dtsize+chunksize<sizeof(resultdata)/sizeof(TCHAR))
        len = chunksize;
    else
        len = sizeof(resultdata)/sizeof(TCHAR)-dtsize;

    // Step 3: GetData
    rc = SQLGetData(stmtHnd,
        1,
        SQL_C_TCHAR,
        (SQLPOINTER)bufp,
        len*sizeof(TCHAR),
        &retchklen);
}
if (!_tcscmp(resultdata, clobdata))
{
    _tprintf(_T("Succeeded!!/n/n"));
}
else
{
    _tprintf(_T("Failed!!/n/n"));
}

```

C.13 Performance and Tuning

This section contains the following topics:

- [General ODBC Programming Guidelines](#)
- [Data Source Configuration Options](#)
- [DATE and TIMESTAMP Data Types](#)

C.13.1 General ODBC Programming Guidelines

Apply the following programming guidelines to improve the performance of an ODBC application:

- Enable connection pooling if the application frequently connects and disconnects from a data source. Reusing pooled connections is extremely efficient compared to reestablishing a connection.
- Minimize the number of times a statement must be prepared. Where possible, use bind parameters to make a statement reusable for different parameter values. Preparing a statement once and running it several times is much more efficient than preparing the statement for every `SQLExecute`.
- Do not include columns in a `SELECT` statement of which you know the application does not retrieve; especially `LONG` columns. Because of the nature of the database server protocols, Oracle ODBC Driver must fetch the entire contents of a `LONG` column if it is included in the `SELECT` statement, regardless of whether the application binds the column or performs a `SQLGetData` operation.
- If you are performing transactions that do not update the data source, then set the `SQL_ATTR_ACCESS_MODE` attribute of the ODBC `SQLSetConnectAttr` function to `SQL_MODE_READ_ONLY`.
- If you are not using ODBC escape clauses, then set the `SQL_ATTR_NOSCAN` attribute of the ODBC `SQLSetConnectAttr` function or the ODBC `SQLSetStmtAttr` function to true.
- Use the ODBC `SQLFetchScroll` function instead of the ODBC `SQLFetch` function for retrieving data from tables that have a large number of rows.

C.13.2 Data Source Configuration Options

This section discusses the performance implications of the following ODBC data source configuration options:

- **Enable Result Sets**
This option enables the support of returning result sets (for example, `RefCursor`) from procedure calls. The default is enabling the returning of result sets.
Oracle ODBC Driver must query the database server to determine the set of parameters for a procedure and their data types in order to determine if there are any `RefCursor` parameters. This query incurs an additional network round trip the first time any procedure is prepared and executed.
- **Enable LOBs**
This option enables the support of inserting and updating LOBs. The default is enabled.
Oracle ODBC Driver must query the database server to determine the data types of each parameter in an `INSERT` or `UPDATE` statement to determine if there are any LOB parameters. This query incurs an additional network round trip the first time any `INSERT` or `UPDATE` is prepared and run.

 **See Also:**

Oracle Database SecureFiles and Large Objects Developer's Guide for more information about LOBs

 **Note:**

LOB data compression enables you to compress SecureFiles to gain disk, Input-Output, and redo logging savings. This reduces costs as compression utilizes space most efficiently and improves the performance of SecureFiles as compression reduces Input-Output and redo logging.

LOB data encryption provides enhanced database security. While the encrypted data is available for random reads and writes, the data is more secure.

Data compression and encryption consumes some additional memory.

- **Bind `TIMESTAMP AS DATE`**

Binds `SQL_TIMESTAMP` parameters as the appropriate Oracle Database data type. If this option is set to `TRUE`, then `SQL_TIMESTAMP` binds as the Oracle `DATE` data type. If this option is set to `FALSE`, then `SQL_TIMESTAMP` binds as the Oracle `TIMESTAMP` data type, which is the default.

- **Enable Closing Cursors**

The `SQL_CLOSE` option of the ODBC function, `SQLFreeStmt`, is supposed to close associated cursors with a statement and discard all pending results. The application can reopen the cursor by running the statement again without doing a `SQLPrepare` again. A typical scenario for this would be an application that expects to be idle for a while but reuses the same SQL statement again. While the application is idle, it may want to free up any associated server resources.

The OCI, on which Oracle ODBC Driver is layered, does not support the functionality of closing cursors. Therefore, by default, the `SQL_CLOSE` option has no effect in Oracle ODBC Driver. The cursor and associated resources remain open on the database.

Enabling this option causes the associated cursor to be closed on the database server. However, this results in the parse context of the SQL statement being lost. The ODBC application can run the statement again without calling `SQLPrepare`. However, internally, Oracle ODBC Driver must prepare and run the statement all over. Enabling this option has a severe performance impact on applications that prepare a statement once and run it repeatedly.

This option should only be enabled if freeing the resources on the server is necessary.

- **Fetch Buffer Size**

Set the Fetch Buffer Size (`FetchBufferSize`) in the `odbc.ini` file to a value specified in bytes. This value is the amount of memory needed that determines how many rows of data Oracle ODBC Driver pre-fetches at a time from an Oracle

Database to the client's cache regardless of the number of rows the application program requests in a single query, thus improving performance.

There is an improvement in the response time of applications that typically fetch fewer than 20 rows of data at a time, particularly over slow network connections or from heavily loaded servers. Setting this too high can have an adverse effect on response time or consume large amounts of memory. The default is 64,000 bytes. You should choose an optimal value for the application.

When the `LONG` and `LOB` data types are present, the number of rows pre-fetched by Oracle ODBC Driver is not determined by the Fetch Buffer Size. The inclusion of the `LONG` and `LOB` data types minimizes the performance improvement and could result in excessive memory use. Oracle ODBC Driver ignores the Fetch Buffer Size and only pre-fetches a set number of rows in the presence of the `LONG` and `LOB` data types.



See Also:

["Format of the Connection String for the SQLDriverConnect Function"](#)

C.13.3 DATE and TIMESTAMP Data Types

If a `DATE` column in the database is used in a `WHERE` clause and the column has an index, then there can be an impact on the performance. For example:

```
SELECT * FROM EMP WHERE HIREDATE = ?
```

In this example, an index on the `HIREDATE` column could be used to make the query run quickly. However, because `HIREDATE` is a `DATE` value and Oracle ODBC Driver is supplying the parameter value as `TIMESTAMP`, the query optimizer of Oracle Database must apply a conversion function. To prevent incorrect results (as might happen if the parameter value had nonzero fractional seconds), the optimizer applies the conversion to the `HIREDATE` column resulting in the following statement:

```
SELECT * FROM EMP WHERE TO_TIMESTAMP(HIREDATE) = ?
```

However, this has the effect of disabling the use of the index on the `HIREDATE` column. Instead, the server performs a sequential scan of the table. If the table has many rows, then this can take a long time. As a workaround for this situation, Oracle ODBC Driver has the connection option to bind `TIMESTAMP` as `DATE`. When this option is enabled, Oracle ODBC Driver binds `SQL_TIMESTAMP` parameters as the Oracle `DATE` data type instead of the Oracle `TIMESTAMP` data type. This enables the query optimizer to use any index on the `DATE` columns.



Note:

This option is intended only for use with Microsoft Access or other similar programs that bind `DATE` columns as `TIMESTAMP` columns. It should not be used when there are actual `TIMESTAMP` columns present or when data loss may occur. Microsoft Access runs such queries using whatever columns are selected as the primary key.

C.14 Error Messages

When an error occurs, Oracle ODBC Driver returns the native error number, the `SQLSTATE` (an ODBC error code), and an error message. The driver derives this information both from errors detected by the driver and errors returned by Oracle Database.

Native Error

For errors that occur in the data source, Oracle ODBC Driver returns the native error returned to it by Oracle Database. When Oracle ODBC Driver or the Driver Manager detects an error, Oracle ODBC Driver returns a native error of zero.

SQLSTATE

For errors that occur in the data source, Oracle ODBC Driver maps the returned native error to the appropriate `SQLSTATE`. When Oracle ODBC Driver or the Driver Manager detects an error, it generates the appropriate `SQLSTATE`.

Error Message

For errors that occur in the data source, Oracle ODBC Driver returns an error message based on the message returned by Oracle Database. For errors that occur in Oracle ODBC Driver or the Driver Manager, Oracle ODBC Driver returns an error message based on the text associated with the `SQLSTATE`.

Error messages have the following format:

```
[vendor] [ODBC-component] [data-source] error-message
```

The prefixes in brackets ([]) identify the source of the error. The following table shows the values of these prefixes returned by Oracle ODBC Driver. When the error occurs in the data source, the `vendor` and `ODBC-component` prefixes identify the vendor and name of the ODBC component that received the error from the data source.

Error Source	Prefix	Value
Driver Manager	[vendor]	[unixODBC]
	[ODBC-component]	[Driver Manager]
	[data-source]	Not applicable
Oracle ODBC Driver	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	Not applicable
Oracle Database	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	[Oracle OCI]

For example, if the error message does not contain the `Ora` prefix shown in the following format, then error is an Oracle ODBC Driver error and should be self-explanatory.

```
[Oracle][ODBC]Error message text here
```

If the error message contains the `Ora` prefix shown in the following format, then it is not an Oracle ODBC Driver error.

```
[Oracle][ODBC][Ora]Error message text here
```



Note:

Although the error message contains the `ORA-` prefix, the actual error may originate from one of several sources.

If the error message text starts with the `ORA-` prefix, then you can obtain more information about the error in Oracle Database documentation.

D

Database Limits

This appendix describes database limits.

D.1 Database Limits

The following table lists the default and maximum values for parameters in a `CREATE DATABASE` or `CREATE CONTROLFILE` statement.

 **Note:**

Interdependencies between these parameters may affect permissible values.

Table D-1 CREATE CONTROLFILE and CREATE DATABASE Parameters

Parameter	Default	Maximum Value
MAXLOGFILES	16	4220
MAXLOGMEMBERS	2	5
MAXLOGHISTORY	100	65534
MAXDATAFILES	30	65534
MAXINSTANCES	1	1055

The following table lists the Oracle Database file size limits in bytes.

Table D-2 File Size Limits

File Type	Platform	File size limit
Data files	Any	4,194,303 multiplied by the value of the <code>DB_BLOCK_SIZE</code> parameter
Import/ Export files and SQL*Loader files	Oracle Solaris, Linux, IBM AIX on POWER Systems (64-Bit), and HP- UX: 32-bit files	2,147,483,647 bytes
Import/ Export files and SQL*Loader files	Oracle Solaris, Linux, IBM AIX on POWER Systems (64-Bit), and HP- UX: 64-bit files	Unlimited Note: File size is limited by limits allowed by the operating system on different file systems.
Control files	Oracle Solaris, Linux, IBM AIX on POWER Systems (64-Bit), and HP-UX	25000 control file blocks with a block size of 4096 bytes.

Index

Symbols

@ abbreviation, [1-1](#)

A

A_TERM environment variable, [6-11](#)

A_TERMCAP environment variable, [6-11](#)

ADA_PATH environment variable, [1-5](#)

adapters utility, [5-2](#)

administering command line SQL, [4-1](#)

AIX tools

Base Operation System tools, [8-5](#)

Performance Tool Box Agent, [8-6](#)

Performance Tool Box Manager, [8-6](#)

SMIT, [8-7](#)

System Management Interface tools, [8-7](#)

ASM_DISKSTRING initialization parameter, [1-8](#)

assistants

Oracle Database Configuration Assistant, [3-2](#)

Oracle Database Upgrade Assistant, [3-1](#)

Oracle Net Configuration Assistant, [3-1](#)

Automatic Storage Management

restarting, [2-3](#)

stopping, [2-1](#)

Automatic Storage Management, using, [8-12](#)

automating

shutdown, [2-3](#)

startup, [2-3](#)

B

bit-length support, [6-5](#)

block size

adjusting, [8-11](#)

buffer cache

tuning size, [8-19](#)

buffer manager, [8-8](#)

C

C library for, [6-5](#)

cache size, [8-19](#)

catching routine, [6-22](#)

example, [6-22](#)

CLASSPATH environment variable, [1-5](#)

client shared libraries, [6-4](#)

client static libraries, [6-4](#)

COBDIR environment variable, [6-10](#)

commands

iostat, [8-4](#)

lsps, [8-4](#)

sar, [8-3](#)

SPOOL, [4-5](#)

swap, [8-4](#)

swapinfo, [8-4](#)

swapon, [8-4](#)

vmstat, [8-2](#)

common environment

setting, [1-6](#)

configuration files

ottcfg.cfg, [6-2](#)

pcbcfg.cfg, [6-2](#)

pccfor.cfg, [6-2](#)

pcscfg.cfg, [6-2](#)

pmscfg.cfg, [6-2](#)

precompiler, [6-2](#)

CREATE CONTROLFILE parameter, [D-1](#)

CREATE DATABASE parameter, [D-1](#)

D

database limits, [D-1](#)

DB_BLOCK_SIZE initialization parameter, [8-14](#)

DB_CACHE_SIZE initialization parameter, [8-14](#)

dbhome file, [1-6](#)

debugger programs, [6-3](#)

demo_proc32.mk file, [6-8](#)

demo_procob.mk file, [6-13](#)

demonstration programs

for Pro*COBOL, [6-12](#)

Oracle Call Interface, [6-18](#)

Oracle JDBC/OCI, [6-19](#)

Pro*C/C++, [6-6](#)

Pro*FORTRAN, [6-14](#)

SQL*Module for Ada, [6-16](#)

demonstrations

PL/SQL, [7-1](#)

precompiler, [7-1](#)

SQL*Loader, [7-1](#)

disk input/output
 file system type, [8-12](#)
 tuning, [8-12](#)

disks
 monitoring performance, [8-13](#)

DISPLAY environment variable, [1-5](#)

dynamic linking
 Oracle libraries and precompilers, [6-3](#)

E

environment variables, [6-10](#)
 A_TERM, [6-11](#)
 A_TERMCAP, [6-11](#)
 ADA_PATH, [1-5](#)
 all, [1-2](#)
 CLASSPATH, [1-5](#)
 COBDIR, [6-10](#)
 DISPLAY, [1-5](#)
 for Pro*COBOL, [6-9](#)
 HOME, [1-5](#)
 LANG, [1-5](#)
 LANGUAGE, [1-5](#)
 LD_LIBRARY_PATH, [1-5](#), [6-10](#), [6-11](#)
 LD_OPTIONS, [1-5](#)
 LIBPATH, [6-10](#)
 LPDEST, [1-5](#)
 MicroFocus COBOL compiler, [6-10](#)
 ORA_TZFILE, [1-2](#)
 ORACLE_BASE, [1-2](#)
 ORACLE_HOME, [1-3](#)
 ORACLE_PATH, [1-3](#)
 ORACLE_SID, [1-1](#), [1-3](#)
 ORACLE_TRACE, [1-3](#)
 ORAENV_ASK, [1-3](#)
 PATH, [1-5](#), [4-4](#), [6-10](#), [6-11](#)
 PRINTER, [1-5](#)
 SHLIB_PATH, [6-10](#)
 SQLPATH, [1-3](#)
 TMPDIR, [1-5](#)
 TNS_ADMIN, [5-1](#)
 TWO_TASK, [1-4](#)
 USER, [1-5](#)

executables
 precompiler, [6-2](#)
 precompilers, [6-2](#)
 relinking, [3-2](#)

Extended file system, [8-12](#)

F

file systems
 ext2/ext3, [8-12](#)
 GPFS, [8-12](#)
 JFS, [8-12](#)

file systems (*continued*)
 OCFS2, [8-12](#)
 S5, [8-12](#)
 UFS, [8-12](#)
 VxFS, [8-12](#)

files
 dbhome, [1-6](#)
 demo_procob.mk, [6-13](#)
 glogin.sql, [4-1](#)
 ins_precomp.mk, [6-2](#)
 login.sql, [4-1](#)
 Oracle Net Services configuration, [5-1](#)
 ottcfg.cfg, [6-2](#)
 pcbcfg.cfg, [6-2](#)
 pccfor.cfg, [6-2](#)
 pcscfg.cfg, [6-2](#)
 pmscfg.cfg, [6-2](#)
 root.sh, [1-6](#)

FORMAT precompiler, [6-13](#)
 Pro*COBOL, [6-14](#)

G

glogin.sql file, [4-1](#)

H

HOME environment variable, [1-5](#)

hugetlbfs
 on SUSE, [A-5](#)

I

I/O support
 asynchronous, [A-2](#)
 direct, [A-2](#)

initialization parameters, [1-8](#)
 ASM_DISKSTRING, [1-8](#)
 DB_BLOCK_SIZE, [8-14](#)
 DB_CACHE_SIZE, [8-14](#)
 JAVA_POOL_SIZE, [8-14](#)
 LARGE_POOL_SIZE, [8-14](#)
 LOG_BUFFERS, [8-14](#)
 SHARED_POOL_SIZE, [8-14](#)

input/output
 tuning, [8-12](#)

ins_precomp.mk file, [6-2](#)

installing
 SQL*Plus command line Help, [4-3](#)

iostat command, [8-4](#)

IPC protocol, [5-3](#)

ireclen, [6-3](#)

J

JAVA_POOL_SIZE initialization parameters, [8-14](#)
 Journaled file system, [8-12](#)

L

LANG environment variable, [1-5](#)
 LANGUAGE environment variable, [1-5](#)
 LARGE_POOL_SIZE initialization parameters, [8-14](#)
 LD_LIBRARY_PATH environment variable, [1-5](#), [6-10](#), [6-11](#)
 LD_OPTIONS environment variable, [1-5](#)
 libcIntst12.a, [6-5](#)
 LIBPATH environment variable, [6-10](#)
 libraries
 client shared and static, [6-4](#)
 Linux
 resource management, [A-4](#)
 Linux tools, [8-5](#)
 listener
 setting up for TCP/IP or TCP/IP with Secure Sockets Layer, [5-4](#)
 LOG_BUFFERS initialization parameters, [8-14](#)
 login.sql file, [4-1](#)
 LPDEST environment variable, [1-5](#)
 lsps command, [8-4](#)

M

make files
 custom, [6-20](#)
 demo_proc32.mk, [6-8](#)
 demo_procomp.mk, [6-13](#)
 ins_precomp.mk, [6-2](#)
 MAXDATAFILES parameter, [D-1](#)
 MAXINSTANCES parameter, [D-1](#)
 MAXLOGFILES parameter, [D-1](#)
 MAXLOGHISTORY parameter, [D-1](#)
 MAXLOGMEMBERS parameter, [D-1](#)
 memory
 control paging, [8-10](#)
 swap space, [8-8](#)
 tuning, [8-8](#)
 MicroFocus COBOL compiler, [6-10](#)
 migrating, [3-1](#)
 Multi-CPU Binding (MCB), [B-4](#)
 multiple signal handlers, [6-22](#)
 multithreaded applications, [6-21](#)

O

OCCI, [6-18](#)
 user programs, [6-19](#)
 OCI, [6-18](#)
 user programs, [6-19](#)
 operating system buffer cache, tuning, [8-19](#)
 operating system commands
 running, [4-4](#)
 operating system tools
 for AIX, [8-5](#)
 iostat, [8-4](#)
 lsps, [8-4](#)
 sar, [8-3](#)
 swap, [8-4](#)
 swapinfo, [8-4](#)
 swapon, [8-4](#)
 vmstat, [8-2](#)
 ORA_NLS10 environment variable, [1-2](#)
 ORA_TZFILE environment variable, [1-2](#)
 Oracle block size, adjusting, [8-11](#)
 Oracle buffer manager, [8-8](#)
 Oracle C++ Call Interface, [6-18](#)
 Oracle Call Interface, [6-18](#)
 demonstration programs, [6-18](#)
 Oracle Cluster Services Synchronization Daemon
 starting, [2-3](#)
 stopping, [2-3](#)
 Oracle Database, [3-2](#)
 restarting, [2-3](#)
 Oracle Database Client:, [6-5](#)
 Oracle Database Configuration Assistant
 configuring, [3-2](#)
 Oracle Database environment variables
 Oracle Database variables, [1-2](#)
 Oracle Database process
 stopping, [2-1](#)
 Oracle Database Upgrade Assistant, [3-1](#)
 Oracle environment variables
 ORA_NLS10, [1-2](#)
 ORACLE_BASE, [1-2](#)
 ORACLE_HOME, [1-3](#)
 ORACLE_SID, [1-3](#)
 Oracle JDBC/OCI
 demonstration programs, [6-19](#)
 Oracle Net Configuration Assistant
 using, [3-1](#)
 Oracle Net Services
 configuration files, [5-1](#)
 IPC protocol, [5-3](#)
 protocol support, [5-2](#)
 protocols, [5-2](#)
 Secure Sockets Layer protocol, [5-3](#)
 TCP/IP protocol, [5-3](#)

Oracle ODBC Driver, [C-1](#)
 Oracle Protocol Support
 IPC protocol, [5-3](#)
 TCP/IP protocol, [5-3](#)
 TCP/IP with Secure Sockets Layer protocol,
 [5-3](#)
 Oracle Solaris
 resource management, [B-4](#)
 Oracle Solaris tools, [8-4](#)
 ORACLE_BASE environment variable, [1-2](#)
 ORACLE_HOME environment variable, [1-3](#)
 ORACLE_PATH environment variable, [1-3](#)
 ORACLE_SID environment variable, [1-1](#), [1-3](#)
 ORACLE_TRACE environment variable, [1-3](#)
 ORAENV_ASK environment variable, [1-3](#)
 oreclen, [6-3](#)
 ottcfg.cfg file, [6-2](#)

P

page-out activity, [8-10](#)
 paging
 tuning, [8-8](#)
 parameters
 CREATE CONTROLFILE, [D-1](#)
 CREATE DATABASE, [D-1](#)
 MAXDATAFILES, [D-1](#)
 MAXLOGFILES, [D-1](#)
 MAXLOGHISTORY, [D-1](#)
 MAXLOGMEMBERS, [D-1](#)
 SGA_MAX_SIZE, [B-3](#)
 shm_max, [8-14](#)
 shm_seg, [8-14](#)
 shmmax, [8-14](#)
 shmseg, [8-14](#)
 PATH environment variable, [1-5](#), [4-4](#), [6-10](#), [6-11](#)
 pbcfg.cfg file, [6-2](#)
 pccfor.cfg file, [6-2](#)
 pcscfg.cfg file, [6-2](#)
 Performance Tool Box Agent, [8-6](#)
 PL/SQL demonstrations, [7-1](#)
 PL/SQL kernel demonstrations, [7-1](#)
 pmscfg.cfg file, [6-2](#)
 postinstallation tasks
 configuration assistants, [3-1](#)
 precompiler executables
 relinking, [6-2](#)
 precompilers
 executables, [6-2](#)
 overview, [6-1](#)
 Pro*C/C++, [6-6](#)
 Pro*COBOL, [6-9](#)
 running demonstrations, [7-1](#)
 signals, [6-22](#)
 uppercase to lowercase conversion, [6-3](#)

precompilers (*continued*)
 value of ireclen and oreclen, [6-3](#)
 vendor debugger programs, [6-3](#)
 PRINTER environment variable, [1-5](#)
 Pro*C/C++
 demonstration programs, [6-6](#)
 make files, [6-7](#)
 signals, [6-22](#)
 user programs, [6-8](#)
 Pro*C/C++ precompiler, [6-6](#)
 Pro*COBOL
 demonstration programs, [6-12](#)
 environment variables, [6-9](#)
 FORMAT precompiler, [6-13](#), [6-14](#)
 naming differences, [6-9](#)
 Oracle Runtime system, [6-12](#)
 user programs, [6-13](#)
 Pro*FORTRAN demonstration programs, [6-14](#)
 PRODUCT_USER_PROFILE table, [4-2](#)
 protocols, [5-2](#)

R

raw devices
 buffer cache size, [8-19](#)
 relinking executables, [3-2](#)
 removing
 SQL*Plus command line Help, [4-3](#)
 restarting
 Automatic Storage Management, [2-3](#)
 Oracle Database, [2-3](#)
 restrictions, SQL*Plus, [4-5](#)
 passwords, [4-5](#)
 resizing windows, [4-5](#)
 return codes, [4-5](#)
 root.sh script, [1-6](#)

S

sar command, [8-3](#), [8-10](#)
 scripts
 root.sh, [1-6](#)
 SGA, [8-14](#)
 determining the size of, [8-15](#)
 SGA_MAX_SIZE parameter, [B-3](#)
 SHARED_POOL_SIZE initialization parameters,
 [8-14](#)
 SHLIB_PATH environment variable, [6-10](#)
 shm_max parameter, [8-14](#)
 shm_seg parameter, [8-14](#)
 shmmax parameter, [8-14](#)
 shmseg parameter, [8-14](#)
 shutdown
 automating, [2-3](#)
 SIGCLD signal, [6-22](#)

SIGCONT signal, [6-22](#)
 SIGINT signal, [6-22](#)
 SIGIO signal, [6-22](#)
 signal handlers, [6-21](#)
 signal routine, [6-22](#)
 example, [6-22](#)
 signals
 SIGCLD, [6-22](#)
 SIGCONT, [6-22](#)
 SIGINT, [6-22](#)
 SIGIO, [6-22](#)
 SIGPIPE, [6-22](#)
 SIGTERM, [6-22](#)
 SIGURG, [6-22](#)
 SIGPIPE signal, [6-22](#)
 SIGTERM signal, [6-22](#)
 SIGURG signal, [6-22](#)
 SPOOL command
 SQL*Plus, [4-5](#)
 SQL*Loader demonstrations, [7-1](#)
 SQL*Module for Ada, [6-16](#)
 demonstration programs, [6-16](#)
 user programs, [6-17](#)
 SQL*Plus
 command line Help, [4-2](#)
 default editor, [4-4](#)
 editor, [4-4](#)
 interrupting, [4-5](#)
 PRODUCT_USER_PROFILE table, [4-2](#)
 restrictions, [4-5](#)
 running operating system commands, [4-4](#)
 site profile, [4-1](#)
 SPOOL command, [4-5](#)
 system editor, [4-4](#)
 user profile, [4-1](#)
 using command-line SQL*Plus, [4-3](#)
 SQL*Plus command line Help
 installing, [4-3](#)
 removing, [4-3](#)
 SQL*Plus, interrupting, [4-5](#)
 SQLPATH environment variable, [1-3](#)
 starting
 Oracle Cluster Services Synchronization
 Daemon, [2-3](#)
 startup
 automating, [2-3](#)
 static linking
 Oracle libraries and precompilers, [6-3](#)
 stopping
 Oracle Cluster Services Synchronization
 Daemon, [2-3](#)
 swap command, [8-4](#)
 swap space, [8-8](#)
 tuning, [8-8](#)
 swap space allocation, [8-8](#)

swapinfo command, [8-4](#)
 swapon command, [8-4](#)
 symfind utility, [6-20](#)
 SYSDATE, [1-7](#)
 system editor
 SQL*Plus, [4-4](#)
 system time, [1-7](#)

T

tables
 PRODUCT_USER_PROFILE, [4-2](#)
 TCP/IP protocol, [5-3](#)
 TCP/IP with Secure Sockets Layer protocol, [5-3](#)
 thread support, [6-21](#)
 TMPDIR environment variable, [1-5](#)
 Transparent Hugepages
 disable for Oracle Database servers, [A-8](#)
 troubleshooting
 I/O lags, [A-8](#)
 ocssd.log threads blocked, [A-8](#)
 tuning, [8-8](#)
 disk input/output, [8-12](#)
 input/output bottlenecks, [8-12](#)
 memory management, [8-8](#)
 tuning tools
 iostat command, [8-4](#)
 lsps command, [8-4](#)
 Performance Tool Box Agent, [8-6](#)
 Performance Tool Box Manager, [8-6](#)
 sar command, [8-3](#)
 swap command, [8-4](#)
 swapinfo command, [8-4](#)
 swapon command, [8-4](#)
 vmstat command, [8-2](#)
 TWO_TASK environment variable, [1-4](#)

U

undefined symbols, [6-20](#)
 Unix file systems, [8-12](#)
 UNIX System V file system, [8-12](#)
 upgraded databases
 upgrading, [3-1](#)
 USER environment variable, [1-5](#)
 user interrupt handler, [6-22](#)
 user profile
 SQL*Plus, [4-1](#)
 user programs
 for Pro*C/C++, [6-8](#)
 OCCI, [6-19](#)
 OCI, [6-19](#)
 Pro*C/C++, [6-8](#)
 Pro*COBOL, [6-13](#)
 SQL*Module for Ada, [6-17](#)

using command-line SQL*Plus, [4-3](#)

utilities

adapters, [5-2](#)

symfind, [6-20](#)

V

Veritas file system, [8-12](#)

vmstat command, [8-2](#)

X

X/Open Distributed Transaction Processing XA
interface, [6-23](#)

XA functionality, [6-23](#)