Oracle® Database

Administrator's Reference

11*g* Release 2 (11.2) for Linux and UNIX-Based Operating Systems

E10839-11

October 2010

ORACLE®

Oracle Database Administrator's Reference, 11*g* Release 2 (11.2) for Linux and UNIX-Based Operating Systems

E10839-11

Copyright © 2006, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Ashmita Bose, Namrata Bhakthavatsalam, Reema Khosla

Contributing Authors: Kevin Flood, Pat Huey, Clara Jaeckel, Emily Murphy, Terri Winters, Prakash Jashnani

Contributors: 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, Douglas Williams, Joseph Therrattil 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 software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software 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 which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on 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. 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.

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documentation x
Conventions xii
Command Syntax xii
Terminology xii
Accessing Documentation xiii

1 Administering Oracle Database

Overview 1-1
Environment Variables 1-1
Oracle Database Environment Variables 1-2
UNIX Environment Variables 1-4
Setting a Common Environment 1-5
Setting the System Time Zone 1-6
Initialization Parameters 1-6
DB_BLOCK_SIZE Initialization Parameter 1-7
ASM_DISKSTRING Initialization Parameter 1-7
LOG_ARCHIVE_DEST_*n* Initialization Parameter 1-7
DISK_ASYNCH_IO Initialization Parameter (HP-UX) 1-7
Operating System Accounts and Groups 1-8
Creating Additional Operating System Accounts 1-8
Configuring the Accounts of Oracle Users 1-8
Using Trace Files 1-8

2 Stopping and Starting Oracle Software

Stopping and Starting Oracle Processes 2-1
Stopping and Starting Oracle Database and Automatic Storage Management Instances 2-1
Stopping an Oracle Database or Automatic Storage Management Instance 2-2
Restarting an Oracle Database or Automatic Storage Management Instance 2-3
Stopping and Starting Oracle Restart 2-3
Stopping and Starting Oracle Enterprise Manager Database Control 2-3
Stopping and Starting Oracle Management Agent 2-4
Automating Shutdown and Startup 2-5

Automating Database Startup and Shutdown on Other Operating Systems 2-5

3 Configuring Oracle Database

Using Configuration Assistants as Standalone Tools 3-1
Using Oracle Net Configuration Assistant 3-1
Using Oracle Database Upgrade Assistant 3-2
Using Oracle Database Configuration Assistant 3-2
Configuring New or Upgraded Databases 3-2
Relinking Executables 3-3

4 Administering SQL*Plus

Administering Command-Line SQL*Plus 4-1
Using Setup Files 4-1
Using the PRODUCT_USER_PROFILE Table 4-2
Using Oracle Database Sample Schemas 4-2
Installing and Removing SQL*Plus Command-Line Help 4-2
Installing SQL*Plus Command-Line Help 4-2
Removing SQL*Plus Command-Line Help 4-3
Using Command-Line SQL*Plus 4-3
Using a System Editor from SQL*Plus 4-3
Running Operating System Commands from SQL*Plus 4-4
Interrupting SQL*Plus 4-4
Using the SPOOL Command 4-4
SQL*Plus Restrictions 4-4
Resizing Windows 4-5
Return Codes 4-5
Hiding the Password 4-5

5 Configuring Oracle Net Services

Locating Oracle Net Services Configuration Files 5-1
Adapters Utility 5-2
Oracle Protocol Support 5-3
IPC Protocol Support 5-3
TCP/IP Protocol Support 5-3
TCP/IP with Secure Sockets Layer Protocol Support 5-4
Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer 5-4
Oracle Advanced Security 5-5

6 Using Oracle Precompilers and the Oracle Call Interface

Overview of Oracle Precompilers 6-1
Precompiler Configuration Files 6-2
Relinking Precompiler Executables 6-2
Precompiler README Files 6-3
Issues Common to All Precompilers 6-3
Static and Dynamic Linking 6-3
Client Shared and Static Libraries 6-3

Bit-Length Support for Client Applications 6-4
Pro*C/C++ Precompiler 6-6
Pro*C/C++ Demonstration Programs 6-6
Pro*C/C++ User Programs 6-7
Pro*COBOL Precompiler 6-8
Pro*COBOL Environment Variables 6-8
Micro Focus Server Express COBOL Compiler 6-9
Acucorp ACUCOBOL-GT COBOL Compiler 6-9
Pro*COBOL Oracle Runtime System 6-10
Pro*COBOL Demonstration Programs 6-11
Pro*COBOL User Programs 6-11
FORMAT Precompiler Option 6-12
Pro*FORTRAN Precompiler 6-12
Pro*FORTRAN Demonstration Programs 6-13
Pro*FORTRAN User Programs 6-13
SQL*Module for ADA 6-14
SQL*Module for Ada Demonstration Programs 6-14
SQL*Module for Ada User Programs 6-15
OCI and OCCI 6-16
OCI and OCCI Demonstration Programs 6-16
OCI and OCCI User Programs 6-16
Oracle JDBC/OCI Programs with a 64-Bit Driver 6-17
Custom Make Files 6-17
Correcting Undefined Symbols 6-18
Multithreaded Applications 6-19
Using Signal Handlers 6-19
XA Functionality 6-20

7 SQL*Loader and PL/SQL Demonstrations

SQL*Loader Demonstrations 7-1
PL/SQL Demonstrations 7-1
Calling 32-Bit External Procedures from 64-Bit Oracle Database PL/SQL 7-4

8 Tuning Oracle Database

Importance of Tuning 8-1
Operating System Tools 8-1
vmstat 8-2
sar 8-2
iostat 8-3
swap, swapinfo, swapon, or lsps 8-3
AIX Tools 8-4
Base Operation System Tools 8-4
Performance Toolbox 8-4
System Management Interface Tool 8-5
HP-UX Tools 8-5
Linux Tools 8-6

Solaris Tools 8-6
Tuning Memory Management 8-7
Allocating Sufficient Swap Space 8-7
Controlling Paging 8-8
Adjusting Oracle Block Size 8-8
Allocating Memory Resource 8-9
Tuning Disk Input-Output 8-9
Using Automatic Storage Management 8-9
Choosing the Appropriate File System Type 8-10
Monitoring Disk Performance 8-10
Monitoring Disk Performance on Other Operating Systems 8-10
Using Disk Resync to Monitor Automatic Storage Management Disk Group 8-11
System Global Area 8-11
Determining the Size of the SGA 8-12
Shared Memory on AIX 8-13
Tuning the Operating System Buffer Cache 8-14

A Administering Oracle Database on AIX

Memory and Paging A-1
Controlling Buffer-Cache Paging Activity A-1
Tuning the AIX File Buffer Cache A-2
Allocating Sufficient Paging Space A-3
Controlling Paging A-3
Setting the Database Block Size A-4
Tuning the Log Archive Buffers A-4
Input-Output Buffers and SQL*Loader A-5
Disk Input-Output Issues A-5
AIX Logical Volume Manager A-5
Using Journaled File Systems Compared to Raw Logical Volumes A-6
Using Asynchronous Input-Output A-9
Input-Output Slaves A-10
Using the DB_FILE_MULTIBLOCK_READ_COUNT Parameter A-11
Using Write Behind A-11
Tuning Sequential Read Ahead A-12
Tuning Disk Input-Output Pacing A-12
Resilvering with Oracle Database A-13
CPU Scheduling and Process Priorities A-13
Changing Process Running Time Slice A-14
Using Processor Binding on SMP Systems A-14
Setting the AIXTHREAD_SCOPE Environment Variable A-14
Network Information Service external naming support A-15
Simultaneous Multithreading on AIX 5.3 and Later Versions A-15
Configuring IBM JSSE Provider with Oracle JDBC Thin Driver A-15

B Administering Oracle Database on HP-UX

HP-UX Shared Memory Segments for an Oracle Instance B-1
HP-UX SCHED_NOAGE Scheduling Policy B-2

Enabling SCHED_NOAGE for Oracle Database B-2
Lightweight Timer Implementation B-3
Asynchronous Input-Output B-3
MLOCK Privilege B-4
Implementing Asynchronous Input-Output B-4
Verifying Asynchronous Input-Output B-6
Verifying That HP-UX Asynchronous Driver is Configured for Oracle Database B-6
Verifying that Oracle Database is Using Asynchronous Input-Output B-7
Asynchronous Flag in SGA B-7
Large Memory Allocations and Oracle Database Tuning B-8
Default Large Virtual Memory Page Size B-8
Tuning Recommendations B-9
Tunable Base Page Size B-9
CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration B-9
Network Information Service external naming support B-10
Activating and Setting Expanded Host Names and Node Names B-10

C Administering Oracle Database on Linux

Extended Buffer Cache Support C-1
Using hugetlbfs on SUSE Linux Enterprise Server 10 or Red Hat Enterprise Linux 4 ..... C-3
Asynchronous Input-Output Support C-3
Simultaneous Multithreading C-4
Allocating Shared Resources C-4
Database Migration from 32-Bit Linux to 64-Bit Linux C-5
Online Backup of Database With RMAN C-5
Migrating 32-Bit Linux Database to 64-Bit Linux Database C-6
Migrating 32-Bit Database to 64-Bit Database With the Same Directory Structure for Data Files C-6
Migrating 32-Bit Database to 64-Bit Database With Different Directory Structure for Data Files C-7
Migrating Data To and From ASM C-11

D Administering Oracle Database on Solaris

Intimate Shared Memory D-1

E Using Oracle ODBC Driver

Oracle ODBC Driver Certification: E-1
Features Not Supported E-2
Implementation of Data Types E-2
Limitations on Data Types E-3
Format of the Connection String for the SQLDriverConnect Function E-3
Reducing Lock Timeout in a Program E-4
Linking ODBC Applications E-5
Obtaining Information About ROWIDs E-5
ROWIDs in a WHERE Clause E-5
Enabling Result Sets E-5

Enabling EXEC Syntax E-11
Supported Functionality E-12
API Conformance E-12
Implementation of ODBC API Functions E-12
Implementation of the ODBC SQL Syntax E-13
Implementation of Data Types E-13
Unicode Support E-14
Unicode Support Within the ODBC Environment E-14
Unicode Support in ODBC API E-14
SQLGetData Performance E-15
Unicode Samples E-15
Performance and Tuning E-21
General ODBC Programming Guidelines E-21
Data Source Configuration Options E-21
DATE and TIMESTAMP Data Types E-23
Error Messages E-24

F Database Limits

Database Limits F-1

G Very Large Memory and HugePages

Very Large Memory on Linux x86 ..... G-1
Implementing VLM on 32-bit Linux G-1
Prerequisites for Implementing VLM G-2
Methods To Increase SGA Limits G-2
Hugemem Kernel G-2
Hugemem Kernel with Very Large Memory G-3
Configuring Very Large Memory for Oracle Database G-4
Restrictions Involved in Implementing Very Large Memory G-5
Overview of HugePages G-6
Tuning SGA With HugePages G-6
Configuring HugePages on RHEL/ Oracle Linux G-6
Restrictions for HugePages Configurations G-8

Index

Preface

This guide provides platform-specific information about administering and configuring Oracle Database 11*g* Release 2 (11.2) on the following platforms:

- IBM AIX on POWER Systems
- HP-UX
- Linux
- Oracle Solaris

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

Audience

This guide is intended for anyone responsible for administering and configuring Oracle Database 11*g* Release 2 (11.2). If you are configuring Oracle RAC, then refer to *Oracle Real Application Clusters Administration and Deployment Guide*.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documentation

Refer to the appropriate section for a listing of platform specific Oracle Database 11*g* documentation:

- IBM AIX on POWER Systems Documentation
- HP-UX Documentation
- Linux Documentation
- Oracle Solaris Documentation

IBM AIX on POWER Systems Documentation

- Oracle Database
  - *Oracle Database Installation Guide for IBM AIX on POWER Systems (64-Bit)*
  - *Oracle Database Installation Guide for IBM AIX on POWER Systems (64-Bit)*
  - *Oracle Database Quick Installation Guide for IBM AIX on POWER Systems (64-Bit)*
  - *Oracle Grid Infrastructure Installation Guide for AIX Based Systems*
  - *Oracle Real Application Clusters Installation Guide for Linux and UNIX*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for IBM AIX on POWER Systems (64-Bit)*
  - *Oracle Database Client Quick Installation Guide for IBM AIX on POWER Systems (64-Bit)*
- Oracle Database Examples
  - *Oracle Database Examples Installation Guide*

HP-UX Documentation

- Oracle Database
  - *Oracle Database Release Notes for HP-UX*
  - *Oracle Database Installation Guide for HP-UX*
  - *Oracle Database Quick Installation Guide for HP-UX PA-RISC (64-Bit)*
  - *Oracle Database Quick Installation Guide for HP-UX Itanium*
  - *Oracle Grid Infrastructure Installation Guide for HP-UX*
  - *Oracle Real Application Clusters Installation Guide for Linux and UNIX*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for HP-UX*
  - *Oracle Database Client Quick Installation Guide for HP-UX PA-RISC (64-Bit)*
  - *Oracle Database Client Quick Installation Guide for HP-UX Itanium*

- Oracle Database Examples
  - *Oracle Database Examples Installation Guide*

Linux Documentation

- Oracle Database
  - *Oracle Database Release Notes for Linux*
  - *Oracle Database Installation Guide for Linux*
  - *Oracle Database Quick Installation Guide for Linux x86*
  - *Oracle Database Quick Installation Guide for Linux x86-64*
  - *Oracle Grid Infrastructure Installation Guide for Linux*
  - *Oracle Real Application Clusters Installation Guide for Linux and UNIX*
  - *Oracle Enterprise Manager Grid Control Basic Installation Guide*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Linux*
  - *Oracle Database Client Quick Installation Guide for Linux x86*
  - *Oracle Database Client Quick Installation Guide for Linux x86-64*
- Oracle Database Examples
  - *Oracle Database Examples Installation Guide*

Oracle Solaris Documentation

- Oracle Database
  - *Oracle Database Release Notes for Oracle Solaris*
  - *Oracle Database Installation Guide for Oracle Solaris*
  - *Oracle Database Quick Installation Guide for Oracle Solaris on SPARC (64-Bit)*
  - *Oracle Database Quick Installation Guide for Oracle Solaris on x86-64 (64-Bit)*
  - *Oracle Grid Infrastructure Installation Guide for Solaris Operating System*
  - *Oracle Real Application Clusters Installation Guide for Solaris Operating System*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Oracle Solaris*
  - *Oracle Database Client Quick Installation Guide for Oracle Solaris on SPARC (64-Bit)*
  - *Oracle Database Client Quick Installation Guide for Oracle Solaris on x86-64 (64-Bit)*
- Oracle Database Examples
  - *Oracle Database Examples Installation Guide*

For important information that was not available when this book was released, refer to the release notes for the platform. The release notes for Oracle Database are updated regularly. You can get the most recent version from Oracle Technology Network at

`http://www.oracle.com/technology/documentation/index.html`

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.
italic	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:
	`dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \`
	`count=10000`

braces { }	Braces indicate required items: `.DEFINE {macro1}`
brackets [ ]	Brackets indicate optional items: `cvtcrt` *`termname`* `[`*`outfile`*`]`
ellipses ...	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname value1 value2 ... valueN`
italic	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line |	A vertical line indicates a choice within braces or brackets: `FILE` *`filesize`* `[K|M]`

Terminology

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

Operating System	Abbreviated Name
IBM AIX on POWER Systems	AIX
HP-UX PA-RISC (64-bit) HP-UX Itanium	HP-UX **Note:** Where the information for HP-UX is different on a particular architecture, this is noted in the text.

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

To access the platform-specific documentation on installation media, use a Web browser to open the `welcome.html` file in the top-level directory.

To access all Oracle documentation, see Oracle Technology Network Web site:

http://www.oracle.com/technology/documentation/index.html

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

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
- Operating System Accounts and Groups
- Using Trace Files

> **See Also:** The appropriate appendix in this guide for platform-specific information about administering Oracle Database

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
```

You can create a syslog audit trail to track administrative activities.

> **See Also:** "Using the Syslog Audit Trail to Audit System Administrators on UNIX Systems" in *Oracle Database Security Guide* for more information on audit trails.

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
- 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
```

Oracle Database Environment Variables

Table 1–1 describes some environment variables used with Oracle Database.

Table 1–1 Oracle Database Environment Variables

Variable	Detail	Definition
`NLS_LANG`	Function	Specifies the language, territory, and character set of the client environment. The client character set specified by `NLS_LANG` must match the character set of the terminal or terminal emulator. If required, `NLS_LANG` 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 `NLS_LANG` can be different from the database character set, in which case the character set is automatically converted. Refer to *Oracle Database Globalization Support Guide* for a list of values for this variable.
	Syntax	*`language_territory.characterset`*
	Example	`french_france.we8iso8859p15`
`ORA_NLS10`	Function	Specifies the directory where the language, territory, character set, and linguistic definition files are stored.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/nls/data`
`ORA_TZFILE`	Function	Specifies the full path and file name of the time zone file. The Oracle Database Server always uses the large time zone file (`$ORACLE_HOME/oracore/zoneinfo/timezlrg_`*`number`*`.dat`). 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 (`$ORACLE_HOME/oracore/zoneinfo/timezone_`*`number`*`.dat`). If you use the small time zone file on the client side, you must ensure that the database you access contains data only in the time zone regions recognized by the small time zone file.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/oracore/zoneinfo/timezlrg_11.dat`
`ORACLE_BASE`	Function	Specifies the base of the Oracle directory structure for Optimal Flexible Architecture compliant installations.
	Syntax	*`directory_path`*
	Example	`/u01/app/oracle`
`ORACLE_HOME`	Function	Specifies the directory containing the Oracle software.
	Syntax	*`directory_path`*

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

Variable	Detail	Definition
	Example	`$ORACLE_BASE/product/11.2.0/dbhome_1`
`ORACLE_PATH`	Function	Specifies the search path for files used by Oracle applications such as SQL*Plus. If the full path to the file is not specified, or if the file is not in the current directory, then the Oracle application uses `ORACLE_PATH` to locate the file.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/u01/app/oracle/product/11.2.0/dbhome_1/bin:.` **Note:** The period adds the current working directory to the search path.
`ORACLE_SID`	Function	Specifies the Oracle system identifier.
	Syntax	A string of numbers and letters that must begin with a letter. Oracle recommends a maximum of 8 characters for system identifiers. For more information about this environment variable, refer to *Oracle Database Installation Guide*.
	Example	SAL1
`ORACLE_TRACE`	Function	Enables the tracing of shell scripts during an installation. If it is set to `T`, then many Oracle shell scripts use the `set -x` command, which prints commands and their arguments as they are run. If it is set to any other value, or no value, then the scripts do not use the `set -x` command.
	Syntax	`T` or not `T`
	Example	`T`
`ORAENV_ASK`	Function	Controls whether the `oraenv` or `coraenv` script prompts or does not prompt for the value of the `ORACLE_SID` environment variable. If it is set to `NO`, then the scripts do not prompt for the value of the `ORACLE_SID` environment variable. If it is set to any other value, or no value, then the scripts prompt for a value for the `ORACLE_SID` environment variable.
	Syntax	`NO` or not `NO`
	Example	`NO`
`SQLPATH`	Function	Specifies the directory or list of directories that SQL*Plus searches for a `login.sql` file.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/home:/home/oracle:/u01/oracle`
`TNS_ADMIN`	Function	Specifies the directory containing the Oracle Net Services configuration files.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/network/admin`
`TWO_TASK`	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: `SQL> CONNECT USERNAME` `Enter password:` *`password`*
	Syntax	Any connect identifier.
	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.
	Example	`PRODDB_TCP`

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

Variable	Detail	Definition
NLS_OS_CHARSET	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.
	Syntax	*characterset*
	Example	WE8ISO8859P1

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.

UNIX Environment Variables

Table 1–2 describes UNIX environment variables used with Oracle Database.

Table 1–2 Environment Variables Used with Oracle Database

Variable	Detail	Definition
ADA_PATH (AIX only)	Function	Specifies the directory containing the Ada compiler.sm
	Syntax	*directory_path*
	Example	/usr/lpp/powerada
ORA_FPU_PRECISION (Linux x86 only)	Function	For all precompiled applications where calculations are done using the extended precision of the x86 Floating Point Unit, this variable must be set to EXTENDED before running the application. **Note**: Setting this variable results in non-IEEE compliant floating point results. Hence, ORA_FPU_PRECISION should not be set if you are using either BINARY_FLOAT or BINARY_DOUBLE datatypes, documented as note 246916.1 in My Oracle Support (formerly Oracle*MetaLink*) Web site: https://support.oracle.com
CLASSPATH	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.
	Syntax	Colon-separated list of directories or files: *directory1:directory2:file1:file2*
	Example	There is no default setting. CLASSPATH must include the following directories: $ORACLE_HOME/JRE/lib:$ORACLE_HOME/jlib
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. Refer to the X Window System documentation for information.
	Syntax	*hostname*:*server*[.*screen*] where *hostname* is the system name (either IP address or alias), *server* is the sequential code number for the server, and *screen* 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). **Note:** If you use a single monitor, then *screen* is optional.
	Example	135.287.222.12:0.0 bambi:0
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*

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

Variable	Detail	Definition
	Example	`/usr/lib:$ORACLE_HOME/lib`
`HOME`	Function	The home directory of the user.
	Syntax	*`directory_path`*
	Example	`/home/oracle`
`LANG` or `LC_ALL`	Function	Specifies the language and character set used by the operating system for messages and other output. Oracle tools that are programmed in Java, such as Oracle Universal Installer and Oracle Database Configuration Assistant, may also use this variable to determine the language of their user interface. Refer to the operating system documentation for more information.
`LD_OPTIONS`	Function	Specifies the default linker options. Refer to the `ld` man page for more information about this environment variable.
`LPDEST` (Solaris only)	Function	Specifies the name of the default printer.
	Syntax	*`string`*
	Example	`docprinter`
`LD_LIBRARY_PATH`	Function	Environment variable to specify the path used to search for libraries on UNIX and Linux. The environment variable may have a different name on some operating systems, such as `LIBPATH` on AIX, and `SHLIB_PATH` on HP-UX.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib`
`PATH`	Function	Used by the shell to locate executable programs; must include the `$ORACLE_HOME/bin` directory.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:$ORACLE_HOME/bin:$HOME/bin:.` **Note**: The period adds the current working directory to the search path.
`PRINTER`	Function	Specifies the name of the default printer.
	Syntax	*`string`*
	Example	`docprinter`
`TEMP`, `TMP`, and `TMPDIR`	Function	Specifies the default directories for temporary files; if set, tools that create temporary files create them in one of these directories.
	Syntax	*`directory_path`*
	Example	`/u02/oracle/tmp`

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.

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:** *Oracle Database Globalization Support Guide* and *Oracle Database Administrator's Guide* for more information about setting the database time zone

Initialization Parameters

The following sections provide information about Oracle Database initialization parameters:

- DB_BLOCK_SIZE Initialization Parameter
- ASM_DISKSTRING Initialization Parameter
- LOG_ARCHIVE_DEST_n Initialization Parameter

- DISK_ASYNCH_IO Initialization Parameter (HP-UX)

DB_BLOCK_SIZE Initialization Parameter

The DB_BLOCK_SIZE initialization parameter specifies the standard block size for the database. This block size is used for the SYSTEM tablespace and by default in other tablespaces.

The maximum value to which you can set the DB_BLOCK_SIZE is 16 KB on Linux x86. It is 32 KB on other platforms.

Note: You cannot change the value of the DB_BLOCK_SIZE initialization parameter after you create a database.

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
AIX	/dev/rhdisk*
HP-UX	/dev/rdisk/*
Solaris	/dev/rdsk/*
Linux	/dev/raw/*

LOG_ARCHIVE_DEST_*n* Initialization Parameter

The maximum value that you can set for ASYNC in the LOG_ARCHIVE_DEST_*n* initialization parameter differs on UNIX platforms as listed in the following table:

Platform	Maximum Value
HP-UX	51200
Other operating systems	102400

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_ASYNCH_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.

Operating System Accounts and Groups

This section describes the following special operating system accounts and groups that are required by Oracle Database:

- Creating Additional Operating System Accounts
- Configuring the Accounts of Oracle Users

Creating Additional Operating System Accounts

If required, create additional operating system accounts. Users must be members of the OSDBA or OSOPER groups to connect to the database with administrator privileges.

Configuring the Accounts of Oracle Users

Update the startup files of the oracle user and the operating system accounts of Oracle users, specifying the appropriate environment variables in the environment file.

For the Bourne, Bash, or Korn shell, add the environment variables to the .profile file, or the .bash_profile file for the Bash shell on Red Hat Enterprise Linux.

For the C shell, add the environment variables to the .login file.

> **Note:** You can use the oraenv or coraenv script to ensure that Oracle user accounts are updated.

Using Trace Files

This section describes the trace (or dump) that Oracle Database creates to help you diagnose and resolve operating problems.

Each server and background process writes to a trace file. When a process detects an internal error, it writes information about the error to its trace file. The file name format of a trace file is *sid_processname_unixpid*.trc, where:

- *sid* is the instance system identifier
- *processname* is a three or four-character abbreviated process name identifying the Oracle Database process that generated the file (for example, pmon, dbwr, ora, or reco)
- *unixpid* is the operating system process ID number

The following is a sample trace file name:

```
$ORACLE_BASE/diag/rdbms/mydb/mydb/trace/test_lgwr_1237.trc
```

Set the `MAX_DUMP_FILE` initialization parameter to at least 5000, to ensure that the trace file is large enough to store error information.

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, 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 the Oracle Database. SRVCTL has been enhanced to support single instance databases with Oracle Restart on standalone servers and on clusters with Oracle Clusterware.

See Also: *Oracle Database Administrator's Guide* and *Oracle Automatic Storage Management Administrator's Guide* for more information about SRVCTL commands

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 Automatic Storage Management Instances
- Stopping and Starting Oracle Restart
- Stopping and Starting Oracle Enterprise Manager Database Control
- Stopping and Starting Oracle Management Agent

Stopping and Starting Oracle Database and Automatic Storage Management Instances

This section describes how to stop and start Oracle Database and Automatic Storage Management instances.

Stopping an Oracle Database or Automatic Storage Management Instance

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

To stop an Oracle Database or Automatic Storage Management instance:

1. To identify the `SID` and Oracle home directory for the instance that should be shut down, run the following command:

   On 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 Automatic Storage Management instance on the system:

   ```
   sid:oracle_home_directory:[Y|N]
   ```

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

2. Depending on the default shell, run the `oraenv` or `coraenv` script to set the environment variables for the instance that should 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.

Restarting an Oracle Database or Automatic Storage Management Instance

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

To restart an Oracle Database or Automatic Storage Management instance:

1. If required, repeat steps 1 and 2 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.

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

Stopping and Starting Oracle Enterprise Manager Database Control

This section describes how to stop and start Oracle Enterprise Manager Database Control.

Stopping Oracle Enterprise Manager Database Control

To stop Oracle Enterprise Manager Database Control:

1. Depending on the default shell, run the `oraenv` or `coraenv` script to set the environment for the database managed by the Database Control that you want to stop:

   - `coraenv` script:

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

   - `oraenv` script:

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

2. Run the following command to stop the Database Control:

```
$ $ORACLE_HOME/bin/emctl stop dbconsole
```

Starting Oracle Enterprise Manager Database Control

To start Database Control:

1. Set the `ORACLE_SID` and `ORACLE_HOME` environment variables to identify the SID and Oracle home directory for the database control that you want to start:
   - Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ ORACLE_SID=sid
     $ export ORACLE_HOME ORACLE_SID
     ```

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home
     % setenv ORACLE_SID sid
     ```

2. Run the following command to start the Database Control:

```
$ $ORACLE_HOME/bin/emctl start dbconsole
```

Stopping and Starting Oracle Management Agent

If you are using Oracle Enterprise Manager Grid Control to manage multiple Oracle products from a central location, then you must have an Oracle Management Agent installed on each host system. Typically, the Oracle Management Agent is installed in its own Oracle home directory.

This section describes how to stop and start Oracle Management Agent.

Stopping Oracle Management Agent

To stop Oracle Management Agent:

1. Run the following command to determine the Oracle home directory for Oracle Management Agent:

   ```
   $ ps -ef | grep emagent
   ```

   This command displays information about the Oracle Management Agent processes. The output of this command is similar to the following:

   ```
   94248 ?? I 0:00.18 oracle_home/agent/bin/emagent ...
   ```

2. If required, set the `ORACLE_HOME` environment variable to specify the appropriate Oracle home directory for the Oracle Management Agent:
   - Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ export ORACLE_HOME
     ```

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

3. Run the following command to stop Oracle Management Agent:

```
$ $ORACLE_HOME/agent/bin/emctl stop agent
```

Starting Oracle Management Agent

To start Oracle Management Agent:

1. If required, set the `ORACLE_HOME` environment variable to specify the appropriate Oracle home directory for Oracle Management Agent:
   - Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ export ORACLE_HOME
     ```

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

2. Run the following command to start Oracle Management Agent:

   ```
   $ $ORACLE_HOME/agent/bin/emctl start agent
   ```

Automating 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 from the system startup files.

> **See Also:** The `init` command in the operating system documentation for more information about system startup and shutdown procedures

Automating Database Startup and Shutdown on Other Operating Systems

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 Solaris:

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

   - On AIX, HP-UX, and Linux:

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

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

   ```
   SID:ORACLE_HOME:{Y|N|W}
   ```

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 that uses an Automatic Storage Management installation that is 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
AIX	`/etc`
Linux and Solaris	`/etc/init.d`
HP-UX	`/sbin/init.d`

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

> **Note:** Change the value of the `ORACLE_HOME` environment variable to an Oracle home directory for the installation. Change the value of the `ORACLE` environment variable to the user name of the owner of the database installed in the Oracle home directory (typically, `oracle`).

```
#! /bin/sh  -x
#
# Change the value of ORACLE_HOME to specify the correct Oracle home
# directory for your installation.

ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1
#
# Change the value of ORACLE to the login name of the
# oracle owner at your site.
#
ORACLE=oracle

PATH=${PATH}:$ORACLE_HOME/bin
HOST=`hostname`
PLATFORM=`uname`
export ORACLE_HOME PATH
#
if [ ! "$2" = "ORA_DB" ] ; then
   if [ "$PLATFORM" = "HP-UX" ] ; then
      remsh $HOST -l $ORACLE -n "$0 $1 ORA_DB"
```

```
      exit
   else
      rsh $HOST -l $ORACLE  $0 $1 ORA_DB
      if [ "$PLATFORM" = "Linux" ] ; then
         touch /var/lock/subsys/dbora
      fi
      exit
   fi
fi
#
case $1 in
'start')
        $ORACLE_HOME/bin/dbstart $ORACLE_HOME &
        ;;
'stop')
        $ORACLE_HOME/bin/dbshut $ORACLE_HOME &
        ;;
*)
        echo "usage: $0 {start|stop}"
        exit
        ;;
esac
#
exit
```

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
AIX	`# ln -s /etc/dbora /etc/rc.d/rc2.d/S99dbora` `# ln -s /etc/dbora /etc/rc.d/rc0.d/K01dbora`
HP-UX	`# ln -s /sbin/init.d/dbora /sbin/rc3.d/S990dbora` `# ln -s /sbin/init.d/dbora /sbin/rc0.d/K001dbora`

Linux	`# 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`

Solaris	`# ln -s /etc/init.d/dbora /etc/rc0.d/K01dbora` `# ln -s /etc/init.d/dbora /etc/rc3.d/S99dbora`

3

Configuring Oracle Database

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

- Using Configuration Assistants as Standalone Tools
- Relinking Executables

Using Configuration Assistants as Standalone Tools

Configuration assistants are usually run during an installation session, but you can also run them in standalone mode. As with Oracle Universal Installer, you can start each of the assistants noninteractively by using a response file.

This section contains the following topics:

- Using Oracle Net Configuration Assistant
- Using Oracle Database Upgrade Assistant
- Using Oracle Database Configuration Assistant
- Configuring New or Upgraded Databases

Using Oracle Net Configuration Assistant

When Oracle Net Server or Oracle Net Client is installed, Oracle Universal Installer automatically starts Oracle Net Configuration Assistant.

If you choose to perform a separate Oracle Database Client installation, then Oracle Net Configuration Assistant automatically creates a configuration that is consistent with the selections made during the installation. Oracle Universal Installer automatically runs Oracle Net Configuration Assistant to set up a net service name in the local naming file located in the `$ORACLE_HOME/network/admin` directory of the client installation.

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 Oracle Database Configuration Assistant to create a database, it automatically updates the network configuration files to include information for the new database.

Using Oracle Database Upgrade Assistant

During an Oracle Database installation, you can choose to upgrade a database from an earlier release to the current release. However, if you choose not to upgrade a database during installation or if there are multiple databases that you want to upgrade, then you can run Oracle Database Upgrade Assistant after the installation.

If you installed Oracle Database 11*g* and chose not to upgrade the database during the installation, then you must upgrade the database before mounting it.

To start Oracle Database Upgrade Assistant, run the following command:

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

For information about the command-line options available with Oracle Database Upgrade Assistant, use the `-help` or `-h` command-line arguments as follows:

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

> **See Also:** *Oracle Database Installation Guide* and *Oracle Database Upgrade Guide* for more information about upgrades

Using Oracle Database Configuration Assistant

You can use Oracle Database Configuration Assistant to:

- Create a default or customized database
- Configure an existing database to use Oracle products
- Create Automatic Storage Management disk groups
- Generate a set of shell and SQL scripts that you can inspect, modify, and run at a later time to create a database

To start Oracle Database Configuration Assistant, run the following command:

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

For information about the command-line options available with Oracle Database Configuration Assistant, use the `-help` or `-h` command-line arguments as follows:

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

Configuring New or Upgraded Databases

Oracle recommends that you run the `utlrp.sql` script after creating or upgrading a database. This script recompiles all PL/SQL modules that may be in an invalid state, including packages, procedures, and types. This is an optional step but Oracle recommends that you do it when you create the database and not at a later date.

To run the `utlrp.sql` script:

1. Switch user to `oracle`.
2. Use the `oraenv` or `coraenv` script to set the environment for the database on which you want to run the `utlrp.sql` script:
   - Bourne, Bash, or Korn shell:

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

   - C shell:

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

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

3. Run the following command to start SQL*Plus:

```
$ sqlplus "/ AS SYSDBA"
```

4. If required, run the following command to start the database:

```
SQL> STARTUP
```

5. Run the `utlrp.sql` script:

```
SQL> @?/rdbms/admin/utlrp.sql
```

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 executables that run in the Oracle home directory that you are relinking. In addition, shut down applications linked with Oracle shared libraries.

The `relink` script does not take any arguments.

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

See Also: "Accessing Oracle Database with SQL*Plus" in *Oracle Database Installation Guide for Linux* for more information on how to use the relink script with Automatic Storage Manager.

To relink 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

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about SQL*Plus

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

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

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, then set the `SYSTEM_PASS` environment variable to the SYSTEM user name and password.

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.

> **See Also:** *Oracle Database Sample Schemas* for information about installing and using Oracle Database Sample Schemas

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

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about the SQL*Plus command-line Help

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.

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
```

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

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.

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 `ed`.

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.

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.

Interrupting SQL*Plus

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

Using the SPOOL Command

The default 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
```

SQL*Plus Restrictions

This section describes the following SQL*Plus restrictions:

- Resizing Windows
- Return Codes
- Hiding the Password

Resizing Windows

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

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.

Hiding the Password

If you 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 should consider using 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_passwsord`* 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
- Adapters Utility
- Oracle Protocol Support
- Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer
- Oracle Advanced Security

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about Oracle Net Services

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 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
sqlnet.ora	$HOME/.sqlnet.ora
tnsnames.ora	$HOME/.tnsnames.ora

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 installation.

Adapters Utility

Use the adapters utility to display the transport protocols, naming methods, and Oracle Advanced Security options that Oracle Database supports on the system. To use the adapters utility, run the following commands:

```
$ cd $ORACLE_HOME/bin
$ adapters ./oracle
```

On an Oracle Database Client system, the adapters utility displays output similar to the following:

```
Oracle Net transport protocols linked with ./oracle are

         IPC
         BEQ
         TCP/IP
         SSL
         RAW

Oracle Net naming methods linked with ./oracle are:

         Local Naming (tnsnames.ora)
         Oracle Directory Naming
         Oracle Host Naming
         NIS Naming

Oracle Advanced Security options linked with ./oracle are:

         RC4 40-bit encryption
         RC4 128-bit encryption
         RC4 256-bit encryption
         DES40 40-bit encryption
         DES 56-bit encryption
         3DES 112-bit encryption
         3DES 168-bit encryption
         AES 128-bit encryption
         AES 192-bit encryption
         SHA crypto-checksumming (for FIPS)
         SHA-1 crypto-checksumming
         Kerberos v5 authentication
         RADIUS authentication
```

```
ENTRUST authentication
```

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

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

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

See Also: *Oracle Database Net Services Administrator's Guide* for more information about Oracle protocol support

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))
```

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 `/etc/services` file. Oracle recommends a value of 1521.

The following is a sample TCP/IP protocol address:

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

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. To use TCP/IP with Secure Sockets Layer, you must install Oracle Advanced Security.

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 `/etc/services` 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))
```

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.

Oracle Advanced Security

When you install Oracle Advanced Security, three `.bak` files are created: `naeet.o.bak`, `naect.o.bak`, and `naedhs.o.bak`. These files are located in the `$ORACLE_HOME/lib` directory.

6

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes how to use Oracle precompilers and the Oracle Call Interface. It contains the following sections:

- 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
- Oracle JDBC/OCI Programs with a 64-Bit Driver
- Custom Make Files
- Correcting Undefined Symbols
- Multithreaded Applications
- Using Signal Handlers
- XA Functionality

Note: To use the demonstrations described in this chapter, install the Oracle Database Examples included on the Oracle Database 11*g* Examples media.

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

- Precompiler README Files
- 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 11*g* by using the Administrator Install type.

Precompiler Configuration Files

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

Table 6–1 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++	pcscfg.cfg
Pro*COBOL	pcbcfg.cfg
Pro*FORTRAN (AIX, HP-UX, and Solaris)	pccfor.cfg
Object Type Translator	ottcfg.cfg
SQL*Module for Ada (AIX)	pmscfg.cfg

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.

Table 6–2 lists the executables for Oracle Precompilers.

Table 6–2 Executables for Oracle Precompilers

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

Precompiler README Files

The README files describe changes made to the precompiler since the last release.

Table 6–3 lists the location of the precompiler README files.

Table 6–3 Location of Precompiler README Files

Precompiler	README File
Pro*C/C++	`$ORACLE_HOME/precomp/doc/proc/readme.doc`
Pro*COBOL	`$ORACLE_HOME/precomp/doc/procob2/readme.doc`
Pro*FORTRAN	`$ORACLE_HOME/precomp/doc/pro1x/readme.txt`

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 `IRECLEN` and `ORECLEN` parameters

  The `IRECLEN` and `ORECLEN` parameters do not have maximum values.

Static and Dynamic Linking

You can statically or dynamically link Oracle libraries with precompiler and OCI or OCCI 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 main memory only once, and it can be shared by multiple application.

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
AIX (32-bit applications)	`LIBPATH`	`$ORACLE_HOME/lib`
AIX (64-bit applications)	`LIBPATH`	`$ORACLE_HOME/lib`
HP-UX (32-bit applications)	`SHLIB_PATH`	`$ORACLE_HOME/lib`
HP-UX (64-bit applications) and Linux	`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`
Solaris (32-bit applications)	`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`
Solaris (64-bit applications)	`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`

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

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
   ```

Nonthreaded 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 one of the following commands to build the OCI application:

- For 32-bit applications:

  ```
  $ make -f demo_rdbms32.mk build_nopthread EXE=oci02 OBJS=oci02.o
  ```

- For 64-bit applications:

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

Bit-Length Support for Client Applications

The following table identifies the bit lengths (31-bit, 32-bit, or 64-bit) supported for client applications:

Note: Starting with Oracle Database 11*g* Release 2 (11.2), 32-bit precompiler, 32-bit client executables, and libraries are not available in the 64-bit Oracle database installation. These are installed as part of the Oracle Database Client Installation.

Client Application Type	Supported Platforms
32-bit only	Linux x86
32-bit and 64-bit	AIX, HP-UX PA-RISC, Linux x86-64, SPARC, HP-UX Itanium and Solaris x86-64

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

Note: Starting with Oracle Database 11*g* Release 2 (11.2), 32-bit and 64-bit Oracle Database Clients are installed in separate Oracle homes.

Platform	32-Bit (or 31-Bit) Client Shared Library	64-Bit Client Shared Library
AIX	`$ORACLE_HOME/lib/libclntsh.a` `$ORACLE_HOME/lib/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.a` `$ORACLE_HOME/lib/libclntsh.so`
HP-UX PA-RISC HP-UX Itanium	`$ORACLE_HOME/lib/libclntsh.sl`	`$ORACLE_HOME/lib/libclntsh.sl`
Linux x86-64, Solaris and Solaris x86-64	`$ORACLE_HOME/lib/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.so`

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
AIX	`LIBPATH`
HP-UX (32-bit client applications)	`SHLIB_PATH`
HP-UX, Linux x86, Linux x86-64, and Solaris	`LD_LIBRARY_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	`$ORACLE_HOME/precomp/demo/proc/demo_proc32.mk`

For Information About. . .	Refer to the Following Make Files. . .
Building 32-bit OCI applications	`$ORACLE_HOME/rdbms/demo/demo_rdbms32.mk`

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

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 `$ORACLE_HOME/sqlplus/demo/demobld.sql` script to exist in the SCOTT schema with the password TIGER.

Note: You must unlock the SCOTT account and set the password before creating the demonstrations.

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

Note: On AIX systems, 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
```

```
$ make -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
```

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
AIX, HP-UX, Linux x86-64, Solaris x86-64 and Solaris	`demo_proc.mk`	`demo_proc32.mk`
Linux x86	NA	`demo_proc.mk`

See Also: The make file for more information about creating user programs

Note: On AIX systems, 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
```

Pro*COBOL Precompiler

Table 6–4 shows the naming conventions for the Pro*COBOL precompiler.

Table 6–4 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

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

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 on page 6-3 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
```

Acucorp ACUCOBOL-GT COBOL Compiler

To use the Acucorp ACUCOBOL-GT COBOL compiler, you must set the A_TERMCAP, 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: libclntsh.so:
cannot open shared object file: No such file or directory
```

A_TERMCAP and A_TERM

Set the A_TERMCAP 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_TERMCAP=/opt/COBOL/etc/a_termcap
$ A_TERM=vt100
$ export A_TERMCAP A_TERM
```

- C shell:

```
% setenv A_TERMCAP /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 AIX, the LIBPATH variable is the LD_LIBRARY_PATH variable equivalent. You must use the LIBPATH variable on AIX 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
```

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
```

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 SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT 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
```

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 (or 31-bit) user programs. The following table shows the make files that you can use to build 32-bit (or 31-bit) and 64-bit user programs with Pro*COBOL:

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, Linux x86-64, and Solaris	`demo_procob.mk`	`demo_procob_32.mk`
Linux x86	Not applicable	`demo_procob.mk`

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 `rtsora`, run the following command:

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

FORMAT Precompiler Option

The FORMAT precompiler option specifies the format of input lines for COBOL. If you specify 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.

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

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 SCOTT schema with the password TIGER.

Note: You must unlock the SCOTT 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
```

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
AIX, HP-UX, and Solaris	demo_profor.mk	demo_profor_32.mk

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
  ```

SQL*Module for ADA

Note: The information in this section applies to the AIX 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

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 ch1_drv demonstration program, the demonstration tables created by the $ORACLE_HOME/sqlplus/demo/demobld.sql script must exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT 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.

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

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

OCI and OCCI Demonstration Programs

Demonstration programs that show the features of OCI and OCCI are provided with the software through the Oracle Database 11*g* Examples media. 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 SCOTT schema with the password TIGER. 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 SCOTT 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
```

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
AIX, HP-UX, Solaris, and Linux x86-64	`demo_rdbms.mk`	`demo_rdbms32.mk`
Linux x86	NA	`demo_rdbms.mk`

See Also: The make file for more information about building user programs

Oracle JDBC/OCI Programs with a 64-Bit Driver

Note:

- The information in this section applies to AIX, HP-UX, Linux x86-64, and Solaris 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. In the `$ORACLE_HOME/jdbc/demo/samples/generic/Makefile` file, modify the JAVA and JAVAC variables 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 AIX, the `LIBPATH` variable is the `LD_LIBRARY_PATH_64` variable equivalent. You must use the `LIBPATH` variable on AIX instead of the `LD_LIBRARY_PATH_64` variable.

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 AIX, 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 AIX, Linux, and 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.

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/11.2.0/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/11.2.0
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
```

Multithreaded Applications

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

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.

Table 6–5 describes the signals that Oracle Database uses for two-task communication.

Table 6–5 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 `wait()` 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. */
                ...
        }
        ...
}
```

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 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 11*g* Examples media. You must unlock SCOTT account and set the password before creating the demonstrations.

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.

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 SCOTT
   Enter password: TIGER
   ```

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 `examp`*n*`.sql` or `sample`*n*`.sql` PL/SQL kernel demonstrations:

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

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

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 HP-UX, Linux, and Solaris:

```
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 AIX:

```
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 SCOTT;
   SQL> CONNECT SCOTT
   Enter password: TIGER
   SQL> CREATE OR REPLACE LIBRARY demolib IS
     2  'oracle_home_path/plsql/demo/extproc.so';
     3  /
   ```

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

   ```
   SQL> @extproc
   ```

PL/SQL Precompiler Demonstrations

> **Note:** The `make` commands shown in this section build the required database objects and load the sample data in the SCOTT 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
```

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 11*g* 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`. Note that 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

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 on page 8-1.

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 Input-Output for the entire system.

This section provides information about the following common tools:

- vmstat
- sar
- iostat
- swap, swapinfo, swapon, or lsps
- AIX Tools
- HP-UX Tools
- Linux Tools
- Solaris Tools

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

vmstat

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 HP-UX and Solaris:

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

- On AIX and Linux:

```
$ vmstat 5 6
```

The following is sample output of this command on HP-UX:

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

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.

The `si` and `so` columns under the `page` column indicate the number of swap-ins and swap-outs per second, respectively. Swap-ins and swap-outs should always be zero.

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.

Note: The output from the `vmstat` command differs across platforms.

See Also: Refer to the man page for information about interpreting the output

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 an HP-UX system, the following command displays a summary of Input-Output activity every ten seconds:

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

The following example shows the output of this command:

```
13:32:45 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
13:32:55       0      14     100       3      10      69       0       0
13:33:05       0      12     100       4       4       5       0       0
13:33:15       0       1     100       0       0       0       0       0
13:33:25       0       1     100       0       0       0       0       0
13:33:35       0      17     100       5       6       7       0       0
13:33:45       0       1     100       0       0       0       0       0
13:33:55       0       9     100       2       8      80       0       0
13:34:05       0      10     100       4       4       5       0       0
13:34:15       0       7     100       2       2       0       0       0
13:34:25       0       0     100       0       0     100       0       0

Average        0       7     100       2       4      41       0       0
```

The `sar` output provides a snapshot of system Input-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`

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 Solaris:

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

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.

swap, swapinfo, swapon, or lsps

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

Platform	Command
AIX	`lsps -a`
HP-UX	`swapinfo -m`
Linux	`swapon -s`
Solaris	`swap -l` and `swap -s`

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

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

AIX Tools

The following sections describe tools available on AIX systems:

- Base Operation System Tools
- Performance Toolbox
- System Management Interface Tool

> **See Also:** The AIX operating system documentation and man pages for more information about these tools

Base Operation System Tools

The AIX Base Operation System contains performance tools that are historically part of UNIX systems or are required to manage the implementation-specific features of AIX. The following table lists the most important Base Operation System tools:

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

Performance Toolbox

The AIX 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 AIX.

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 `trace` 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 for AIX Guide and Reference* for information about these tools
- *AIX 5L Performance Management Guide* for information about the syntax of some of these tools

System Management Interface Tool

The AIX 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.

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:

Tools	Function
`caliper` (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.
`gprof`	Creates an execution profile for programs.
`monitor`	Monitors the program counter and calls to certain functions.
`netfmt`	Monitors the network.
`netstat`	Reports statistics on network performance.
`nfsstat`	Displays statistics about Network File System and Remote Procedure Call activity.
`nettl`	Captures network events or packets by logging and tracing.
`prof`	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.
`profil`	Copies program counter information into a buffer.
`top`	Displays the top processes on the system and periodically updates the information.

Linux Tools

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

Solaris Tools

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

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
- Controlling Paging
- Adjusting Oracle Block Size
- Allocating Memory Resource

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
AIX	`lsps -a`
HP-UX	`swapinfo -m`
Linux	`swapon -s`
Solaris	`swap -l` and `swap -s`

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

Platform	Command
AIX	`chps` or `mkps`
HP-UX	`swapon`
Linux	`swapon -a`
Solaris	`swap -a`

Set the swap space to between two and four times the physical memory. Monitor the use of swap space, and increase it as required.

> **See Also:** The operating system documentation for more information about these commands

Controlling 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

The following table lists the important columns from the output of these commands:

Platform	Column	Function
Solaris	`vflt/s`	Indicates the number of address translation page faults. Address translation faults occur when a process refers to a valid page not in memory.
HP-UX	`at`	Indicates the number of address translation page faults. Address translation faults occur when a process refers to a valid page not in memory.
HP-UX	`re`	Indicates the number of valid pages that have been reclaimed and added to the free list by page-out activity. This value should be zero.

If 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.

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.

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, Solaris, AIX and HP-UX platforms.

On Solaris, Dynamic Intimate Shared Memory is enabled for MEMORY_TARGET or MEMORY_MAX_TARGET.

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 on page C-4.

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.

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

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.

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
S5	HP-UX and Solaris	UNIX System V file system
UFS	AIX, HP-UX, and Solaris	Unified file system, derived from BSD UNIX
VxFS	AIX, HP-UX, and Solaris	VERITAS file system
ext2/ext3	Linux	Extended file system for Linux
OCFS2	Linux	Oracle cluster file system
JFS/JFS2	AIX	Journaled file system
GPFS	AIX	General parallel file system

The suitability of a file system for an application is usually not documented. For example, even different implementations of the Unified file system 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.

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

Monitoring Disk Performance on Other 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
`bread/s, bwrit/s`	Blocks read and blocks written per second (important for file system databases)
`pread/s, pwrit/s`	Partitions read and partitions written per second (important for raw partition database systems)

An important `sar -u` column for analyzing disk performance is `%wio`, the percentage of CPU time spent waiting on blocked Input-Output.

> **Note:** Not all Linux distributions display the `%wio` column in the output of the `sar -u` command. For detailed Input-Output statistics, you can use `iostat -x` command.

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. A good default value is no more than 40 for 2 drives and no more than 60 for 4 to 8 drives.
- 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.
- If the `%wio` column value is consistently greater than 20, then the system is Input-Output bound.

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.

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
AIX	NA
HP-UX	The size of the physical memory installed on the system

Platform	Recommended Value
Linux	Minimum of the following values: ■ Half the size of the physical memory installed on the system ■ 4GB - 1 byte
Solaris	4294967295 or 4 GB minus 16 MB **Note:** If the system runs both Oracle9*i* Database and Oracle Database 11*g* instances, then you must set the value of `shm_max` to 2 GB minus 16 MB. On Solaris, this value can be greater than 4 GB to accommodate larger SGA sizes.

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
- Shared Memory on AIX

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.

- When you start the database instance, the size of the SGA is displayed next to the Total System Global Area heading.
- Run the `ipcs` command as the `oracle` user.

Shared Memory on AIX

Note: The information in this section applies only to AIX.

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 AIX 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 AIX POWER4- and POWER5-Based Systems

To turn on and reserve 10 large pages each of size 16 MB on a POWER4 or POWER 5 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 16MB size pages is configured, then this memory will be 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 AIX operating system to use 16MB 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 AIX 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.

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 AIX

This appendix contains information about administering Oracle Database on AIX. It includes the following topics:

- Memory and Paging
- Disk Input-Output Issues
- CPU Scheduling and Process Priorities
- Setting the AIXTHREAD_SCOPE Environment Variable
- Network Information Service external naming support
- Simultaneous Multithreading on AIX 5.3 and Later Versions
- Configuring IBM JSSE Provider with Oracle JDBC Thin Driver

Memory and Paging

Memory contention occurs when processes require more memory than is available. To cope with the shortage, the system pages programs and data between memory and disks.

This section contains the following topics:

- Controlling Buffer-Cache Paging Activity
- Tuning the AIX File Buffer Cache
- Allocating Sufficient Paging Space
- Controlling Paging
- Setting the Database Block Size
- Tuning the Log Archive Buffers
- Input-Output Buffers and SQL*Loader

Controlling Buffer-Cache Paging Activity

Excessive paging activity decreases performance substantially. This can become a problem with database files created on journaled file systems (JFS and JFS2). In this situation, a large number of SGA data buffers may also have analogous file system buffers containing the most frequently referenced data. The behavior of the AIX file buffer cache manager can have a significant impact on performance. It can cause an Input-Output bottleneck, resulting in lower overall system throughput.

It is possible to tune buffer-cache paging activity, but you must do it carefully and infrequently. Use the `/usr/sbin/vmo` command to tune the AIX system parameters in the following table:

Parameter	Description
`minfree`	The minimum free-list size. If the free-list space in the buffer falls lower than this size, then the system uses page stealing to replenish the free list.
`maxfree`	The maximum free-list size. If the free-list space in the buffer exceeds this size, then the system stops using page stealing to replenish the free list.
`minperm`	The minimum number of permanent buffer pages for file Input-Output.
`maxperm`	The maximum number of permanent buffer pages for file Input-Output.

See Also: *AIX 5L Performance Management Guide* for more information about AIX system parameters

Tuning the AIX File Buffer Cache

The purpose of the AIX file buffer cache is to reduce disk access frequency when journaled file systems are used. If this cache is too small, then disk usage increases and potentially saturates one or more disks. If the cache is too large, then memory is wasted.

You can configure the AIX file buffer cache by adjusting the `minperm` and `maxperm` parameters. In general, if the buffer hit ratio is low (less than 90 percent), as determined by the `sar -b` command, then increasing the `minperm` parameter value may help. If maintaining a high buffer hit ratio is not critical, then decreasing the `minperm` parameter value increases the physical memory available. Refer to the AIX documentation for more information about increasing the size of the AIX file buffer cache.

The performance gain cannot be quantified easily, because it depends on the degree of multiprogramming and the Input-Output characteristics of the workload.

Tuning the minperm and maxperm Parameters

AIX provides a mechanism for you to loosely control the ratio of page frames used for files rather than those used for computational (working or program text) segments by adjusting the `minperm` and `maxperm` values according to the following guidelines:

- If the percentage of real memory occupied by file pages falls lower than the `minperm` value, then the virtual memory manager (VMM) page-replacement algorithm takes both file and computational pages, regardless of repage rates.
- If the percentage of real memory occupied by file pages rises above the `maxperm` value, then the VMM page-replacement algorithm takes both file and computational pages.
- If the percentage of real memory occupied by file pages is between the `minperm` and `maxperm` parameter values, then the VMM normally takes only file pages. However, if the repaging rate for file pages is higher than the repaging rate for computational pages, then the computational pages are taken as well.

Use the following algorithm to calculate the default values:

- `minperm` (in pages) = ((number of page frames)-1024) * 0.2
- `maxperm` (in pages) = ((number of page frames)-1024) * 0.8

Use the following command to change the value of the `minperm` parameter to 5 percent of the total number of page frames, and the value of the `maxperm` parameter to 20 percent of the total number of page frames:

```
# /usr/sbin/vmo -o minperm percent=5 -o maxperm percent=20
```

The default values are 20 percent and 80 percent, respectively.

To optimize for quick response when opening new database connections, adjust the `minfree` parameter to maintain enough free pages in the system to load the application into memory without adding additional pages to the free list. To determine the real memory size (resident set size, working set) of a process, use the following command:

```
$ ps v process_id
```

Set the `minfree` parameter to this value or to 8 frames, whichever is larger.

If you are using Direct Input-Output, then you can set the `minperm` and `maxperm` parameters to low values. For example, 5 percent and 20 percent, respectively. This is because the AIX file buffer cache is not used for Direct Input-Output. The memory may be better used for other purposes, such as for the Oracle System Global Area.

Allocating Sufficient Paging Space

Inadequate paging space (swap space) usually causes the system to stop responding or show very slow response times. On AIX, you can dynamically add paging space on raw disk partitions. The amount of paging space you should configure depends on the amount of physical memory present and the paging space requirements of the applications. Use the `lsps` command to monitor paging space use and the `vmstat` command to monitor system paging activities. To increase the paging space, use the `smit pgsp` command.

If paging space is preallocated, then Oracle recommends that you set the paging space to a value larger than the amount of RAM. But on AIX, paging space is not allocated until required. The system uses swap space only if it runs out of real memory. If the memory is sized correctly, then there is no paging and the page space can be small. Workloads where the demand for pages does not fluctuate significantly perform well with a small paging space. Workloads likely to have peak periods of increased paging require enough paging space to handle the peak number of pages.

As a general rule, an initial setting for the paging space is half the size of RAM plus 4 GB, with an upper limit of 32 GB. Monitor the paging space use with the `lsps -a` command, and increase or decrease the paging space size accordingly. The metric percent Used in the output of `lsps -a` is typically less than 25 percent on a healthy system. A properly sized deployment requires very little paging space and an excessive amount of swapping is an indication that the RAM on the system may be undersized.

Caution: Do not undersize the paging space. If you do, then the system terminates active processes when it runs out of space. However, oversizing the paging space has little or no negative impact.

Controlling Paging

Constant and excessive paging indicates that the real memory is over-committed. In general, you should:

- Avoid constant paging unless the system is equipped with very fast expanded storage that makes paging between memory and expanded storage much faster than Oracle Database can read and write data between the SGA and disks.
- Allocate limited memory resource to where it is most beneficial to system performance. It is sometimes a recursive process of balancing the memory resource requirements and trade-offs.
- If memory is not adequate, then build a prioritized list of memory-requiring processes and elements of the system. Assign memory to where the performance gains are the greatest. A prioritized list may look like the following:
  1. OS and RDBMS kernels
  2. User and application processes
  3. Redo log buffer
  4. PGAs and shared pool
  5. Database block buffer caches

For example, suppose you query Oracle Database dynamic performance tables and views and find that both the shared pool and database buffer cache require more memory. Then, assigning the limited spare memory to the shared pool may be more beneficial than assigning it to the database block buffer caches.

The following AIX commands provide paging status and statistics:

- `vmstat -s`
- `vmstat` *`interval`* `[`*`repeats`*`]`
- `sar -r` *`interval`* `[`*`repeats`*`]`

Setting the Database Block Size

You can configure Oracle Database block size for better Input-Output throughput. On AIX, you can set the value of the `DB_BLOCK_SIZE` initialization parameter to between 2 KB and 32 KB, with a default of 4 KB. If Oracle Database is installed on a journaled file system, then the block size should be a multiple of the file system block size (4 KB on JFS, 16 K to 1 MB on GPFS). For databases on raw partitions, Oracle Database block size is a multiple of the operating system physical block size (512 bytes on AIX).

Oracle recommends smaller Oracle Database block sizes (2 KB or 4 KB) for online transaction processing or mixed workload environments and larger block sizes (8 KB, 16 KB, or 32 KB) for decision support system workload environments.

Tuning the Log Archive Buffers

By increasing the `LOG_BUFFER` size, you may be able to improve the speed of archiving the database, particularly if transactions are long or numerous. Monitor the log file Input-Output activity and system throughput to determine the optimum `LOG_BUFFER` size. Tune the `LOG_BUFFER` parameter carefully to ensure that the overall performance of normal database activity does not degrade.

Note: The `LOG_ARCHIVE_BUFFER_SIZE` parameter was obsoleted with Oracle8*i* Database.

Input-Output Buffers and SQL*Loader

For high-speed data loading, such as using the SQL*Loader direct path option in addition to loading data in parallel, the CPU spends most of its time waiting for Input-Output to complete. By increasing the number of buffers, you can maximize CPU usage, and by doing this, increase overall throughput.

The number of buffers (set by the SQL*Loader `BUFFERS` parameter) you choose depends on the amount of available memory and how much you want to maximize CPU usage.

The performance gains depend on CPU usage and the degree of parallelism that you use when loading data.

> **See Also:** *Oracle Database Utilities* for information about adjusting the file processing options string for the `BUFFERS` parameter and for information about the SQL*Loader utility

BUFFER Parameter for the Import Utility

The `BUFFER` parameter for the Import utility should be set to a large value to optimize the performance of high-speed networks when they are used. For example, if you use the IBM RS/6000 Scalable POWER parallel Systems switch, then you should set the `BUFFER` parameter to a value of at least 1 MB.

Disk Input-Output Issues

Disk Input-Output contention can result from poor memory management (with subsequent paging and swapping), or poor distribution of tablespaces and files across disks.

Ensure that the Input-Output activity is distributed evenly across multiple disk drives by using AIX utilities such as `filemon`, `sar`, `iostat`, and other performance tools to identify disks with high Input-Output activity.

This section contains the following topics:

- AIX Logical Volume Manager
- Using Journaled File Systems Compared to Raw Logical Volumes
- Using Asynchronous Input-Output
- Input-Output Slaves
- Using the DB_FILE_MULTIBLOCK_READ_COUNT Parameter
- Using Write Behind
- Tuning Sequential Read Ahead
- Tuning Disk Input-Output Pacing
- Resilvering with Oracle Database

AIX Logical Volume Manager

The AIX Logical Volume Manager can stripe data across multiple disks to reduce disk contention. The primary objective of striping is to achieve high performance when reading and writing large sequential files. Effective use of the striping features in the Logical Volume Manager enables you to spread Input-Output more evenly across disks, resulting in better overall performance.

Note: Do not add logical volumes to Automatic Storage Management disk groups. Automatic Storage Management works best when you add raw disk devices to disk groups. If you are using Automatic Storage Management, then do not use Logical Volume Manager for striping. Automatic Storage Management implements striping and mirroring.

Design a Striped Logical Volume

When you define a striped logical volume, you must specify the items listed in the following table:

Item	Recommended Settings
Drives	There must be at least two physical drives. The drives should have minimal activity when performance-critical sequential Input-Output is carried out. Sometimes, you must stripe the logical volume between two or more adapters.
Stripe unit size	Although the stripe unit size can be any power of 2 (from 2 KB to 128 KB), stripe sizes of 32 KB and 64 KB are good values for most workloads. For Oracle Database files, the stripe size must be a multiple of the database block size.
Size	The number of physical partitions allocated to the logical volume must be a multiple of the number of disk drives used.
Attributes	Cannot be mirrored. Set the `copies` attribute to a value of 1.

Other Considerations

Performance gains from effective use of the Logical Volume Manager can vary greatly, depending on the Logical Volume Manager you use and the characteristics of the workload. For decision support system workloads, you can see substantial improvement. For online transaction processing-type or mixed workloads, you can expect significant performance gains.

Using Journaled File Systems Compared to Raw Logical Volumes

Address the following considerations when deciding whether to use journaled file systems or raw logical volumes:

- File systems are continually being improved, as are various file system implementations.
- File systems require some additional configuration (AIX `minservers` and `maxservers` parameter) and add a small CPU overhead because Asynchronous Input-Output on file systems is serviced outside the kernel.
- Different vendors implement the file system layer in different ways to capitalize on the strengths of different disks. This makes it difficult to compare file systems across platforms.
- The introduction of more powerful Logical Volume Manager interfaces substantially reduces the tasks of configuring and backing up logical disks based on raw logical volumes.
- The Direct Input-Output and Concurrent Input-Output features included in IBM AIX on POWER Systems improve file system performance to a level comparable to raw logical volumes.

In earlier versions of AIX, file systems supported only buffered read and write and added extra contention because of imperfect inode locking. These two issues are solved by the JFS2 Concurrent Input-Output feature and the GPFS Direct Input-Output feature.

> **Note:** To use the Oracle RAC option, you must place data files on an Automatic Storage Management disk group or on a GPFS file system. You cannot use JFS or JFS2. Direct Input-Output is implicitly enabled when you use GPFS.

File System Options

IBM AIX on POWER Systems includes Direct Input-Output and Concurrent Input-Output support. Direct Input-Output and Concurrent Input-Output support enables database files to exist on file systems while bypassing the operating system buffer cache and removing inode locking operations that are redundant with the features provided by Oracle Database.

Where possible, Oracle recommends enabling Concurrent Input-Output or Direct Input-Output on file systems containing Oracle data files. The following table lists file systems available on AIX and the recommended setting:

File System	Option	Description
JFS	dio	Concurrent Input-Output is not available on JFS. Direct Input-Output is available, but performance is degraded compared to JFS2 with Concurrent Input-Output.
JFS large file	none	Oracle does not recommend using JFS large file for Oracle Database because its 128 KB alignment constraint prevents you from using Direct Input-Output.
JFS2	cio	Concurrent Input-Output is a better setting than Direct Input-Output on JFS2, because it provides support for multiple concurrent readers and writers on the same file. However, due to AIX restrictions on JFS2/CIO, Concurrent Input-Output is intended to be used only with Oracle data files, control files, and log files. It should be applied only to file systems that are dedicated to such a purpose. For the same reason, the Oracle home directory is not supported on a JFS2 file system mounted with the `cio` option. For example, during installation, if you inadvertently specify that the Oracle home directory is on a JFS2 file system mounted with the CIO option, then while trying to relink Oracle, you may encounter the following error: `"ld: 0711-866 INTERNAL ERROR: Output symbol table size miscalculated"`
GPFS	NA	Oracle Database silently enables Direct Input-Output on GPFS for optimum performance. GPFS Direct Input-Output already supports multiple readers and writers on multiple nodes. Therefore, Direct Input-Output and Concurrent Input-Output are the same thing on GPFS.

Considerations for JFS and JFS2

If you are placing Oracle Database logs on a JFS2 file system, then the optimal configuration is to create the file system using the `agblksize=512` option and to mount it with the `cio` option.

Before Oracle Database 11*g*, Direct Input-Output and Concurrent Input-Output could not be enabled at the file level on JFS/JFS2. Therefore, the Oracle home directory and

data files had to be placed in separate file systems for optimal performance. The Oracle home directory was placed on a file system mounted with default options, with the data files and logs on file systems mounted using the `dio` or `cio` options.

With Oracle Database 11*g*, you can enable Direct Input-Output and Concurrent Input-Output on JFS/JFS2 at the file level. You can do this by setting the `FILESYSTEMIO_OPTIONS` parameter in the server parameter file to `setall` or `directIO`. This enables Concurrent Input-Output on JFS2 and Direct Input-Output on JFS for all data file Input-Output. Because the `directIO` setting disables asynchronous Input-Output it should normally not be used. As a result of this 11*g* feature, you can place data files on the same JFS/JFS2 file system as the Oracle home directory and still use Direct Input-Output or Concurrent Input-Output for improved performance. As mentioned earlier, you should still place Oracle Database logs on a separate JFS2 file system for optimal performance.

Considerations for GPFS

If you are using GPFS, then you can use the same file system for all purposes. This includes using it for the Oracle home directory and for storing data files and logs. For optimal performance, you should use a large GPFS block size (typically, at least 512 KB). GPFS is designed for scalability, and there is no requirement to create multiple GPFS file systems as long as the amount of data fits in a single GPFS file system.

Moving from a Journaled File System to Raw Logical Volumes

To move from a journaled file system to raw devices without having to manually reload all the data, perform the following steps as the `root` user:

1. Create a raw device (preferably, in a BigVG) using the new raw logical volume device type (`-T O`), which enables putting the first Oracle block at offset zero for optimal performance:

   ```
   # mklv -T O -y new_raw_device VolumeGroup NumberOfPartitions
   ```

   > **Note:** The raw device should be larger than the existing file. In addition, you must bear in mind the size of the new raw device to prevent wasting space.

2. Set the permissions on the raw device.
3. Use `dd` to convert and copy the contents of the JFS file to the new raw device as follows:

   ```
   # dd if=old_JFS_file of=new_raw_device bs=1m
   ```

4. Rename the data file.

Moving from Raw Logical Volumes to a Journaled File System

The first Oracle block on a raw logical volume is not necessarily at offset zero. However, the first Oracle block on a file system is always at offset zero. To determine the offset and locate the first block on a raw logical volume, use the `$ORACLE_HOME/bin/offset` command. The offset can be 4096 bytes or 128 KB on AIX logical volumes or zero on AIX logical volumes created with the `mklv -T O` option.

When you have determined the offset, you can copy over data from a raw logical volume to a file system using the `dd` command and skipping the offset. The following example assumes an offset of 4096 bytes:

```
# dd if=old_raw_device bs=4k skip=1|dd of=new_file bs=256
```

You can instruct Oracle Database to use many blocks smaller than the maximum capacity of a raw logical volume. If you do this, then you must add a `count` clause to ensure that only data that contains Oracle blocks is copied. The following example assumes an offset of 4096 bytes, an Oracle block size of 8 KB, and 150000 blocks:

```
# dd if=old_raw_device bs=4k skip=1|dd bs=8k count=150000|dd of=new_file bs=256k
```

Using Asynchronous Input-Output

Oracle Database takes full advantage of asynchronous Input-Output provided by AIX, resulting in faster database access.

IBM AIX on POWER Systems support asynchronous Input-Output for database files created on file system partitions. When using asynchronous Input-Output on file systems, the kernel database processes (`aioserver`) control each request from the time a request is taken off the queue to the time it is completed. The number of `aioserver` servers determines the number of asynchronous Input-Output requests that can be processed in the system concurrently. So, it is important to tune the number of `aioserver` processes when using file systems to store Oracle Database data files.

Use one of the following commands to set the number of servers. This applies only when using asynchronous Input-Output on file systems:

- `smit aio`
- `chdev -l aio0 -a maxservers=' m ' -a minservers='n'`

See Also:

- The System Management Interface Tool (SMIT) online Help for more information about SMIT
- The man pages for more information about the `smit aio` and `chdev` commands

---

Note: On IBM AIX on POWER Systems, there are two asynchronous Input-Output subsystems available. Oracle Database 11*g* uses Legacy asynchronous Input-Output (`aio0`), even though the Oracle preinstallation script enables Legacy asynchronous Input-Output (`aio0`) and POSIX AIO (`posix_aio0`). Both asynchronous Input-Output subsystems have the same performance characteristics.

Set the minimum value to the number of servers to be started when the system is started. Set the maximum value to the number of servers that can be started in response to a large number of concurrent requests. These parameters apply to file systems only.

The default value for the minimum number of servers is 1. The default value for the maximum number of servers is 10. These values are usually too low to run Oracle Database on large systems with 4 CPUs or more, if you are not using kernelized asynchronous Input-Output. Oracle recommends that you set the parameters to the values listed in the following table:

Parameter	Value
minservers	Oracle recommends an initial value equal to the number of CPUs on the system or 10, whichever is lower.
maxservers	Starting with AIX 5L version 5.3, this parameter counts the maximum number of asynchronous Input-Output servers for each CPU. On previous versions of AIX, it was a systemwide value. If you are using General Parallel File System (GPFS), then set maxservers to worker1threads divided by the number of CPUs. This is the optimal setting. Increasing maxservers does not lead to improved Input-Output performance. If you are using JFS/JFS2, then set the initial value to 10 times the number of logical disks divided by the number of CPUs. Monitor the actual number of aioservers started during a typical workload using the pstat or ps commands. If the actual number of active aioservers equals the maxservers, then increase the maxservers value.
maxreqs	Set the initial value to 4 times the number of logical disks multiplied by the queue depth. You can determine the queue depth by running the following command: `$ lsattr -E -l hdiskxx` Typically, the queue depth is 3.

If the value of the maxservers or maxreqs parameter is set too low, then the following warning messages are repeatedly displayed:

```
Warning: lio_listio returned EAGAIN
Performance degradation may be seen.
```

You can avoid these errors by increasing the value of the maxservers parameter. To display the number of asynchronous Input-Output servers running, enter the following commands as the root user:

```
# pstat -a | grep -c aios
# ps -k | grep aioserver
```

Check the number of active asynchronous Input-Output servers periodically, and change the values of the minservers and maxservers parameters if required. The changes take place when the system is restarted.

Input-Output Slaves

Input-Output Slaves are specialized Oracle processes that perform only Input-Output. They are rarely used on AIX, because asynchronous Input-Output is the default and recommended way for Oracle to perform Input-Output operations on AIX.
Input-Output Slaves are allocated from shared memory buffers. Input-Output Slaves use the initialization parameters listed in the following table:

Parameter	Range of Values	Default Value
DISK_ASYNCH_IO	true/false	true
TAPE_ASYNCH_IO	true/false	true
BACKUP_TAPE_IO_SLAVES	true/false	false
DBWR_IO_SLAVES	0 - 999	0

Parameter	Range of Values	Default Value
DB_WRITER_PROCESSES	1-20	1

Generally, you do not adjust the parameters in the preceding table. However, on large workloads, the database writer may become a bottleneck. If it does, then increase the value of DB_WRITER_PROCESSES. As a general rule, do not increase the number of database writer processes above one for each pair of CPUs in the system or partition.

There are times when you must turn off asynchronous I/O. For example, if instructed to do so by Oracle Support for debugging. You can use the DISK_ASYNCH_IO and TAPE_ASYNCH_IO parameters to switch off asynchronous I/O for disk or tape devices. TAPE_ASYNCH_IO support is only available when the Media Manager software supports it and for Recovery Manager, if BACKUP_TAPE_IO_SLAVES is true.

Set the DBWR_IO_SLAVES parameter to greater than 0 only if the DISK_ASYNCH_IO parameter is set to false. Otherwise, the database writer process becomes a bottleneck. In this case, the optimal value on AIX for the DBWR_IO_SLAVES parameter is 4.

Using the DB_FILE_MULTIBLOCK_READ_COUNT Parameter

When using Direct Input-Output or Concurrent Input-Output with Oracle Database 11*g*, the AIX file system does not perform any read-ahead on sequential scans. For this reason the DB_FILE_MULTIBLOCK_READ_COUNT value in the server parameter file should be increased when Direct Input-Output or Concurrent Input-Output is enabled on Oracle data files. The read ahead is performed by Oracle Database as specified by the DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter.

Setting a large value for the DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter usually yields better Input-Output throughput on sequential scans. On AIX, this parameter ranges from 1 to 512, but using a value higher than 16 usually does not provide additional performance gain.

Set this parameter so that its value when multiplied by the value of the DB_BLOCK_SIZE parameter produces a number larger than the Logical Volume Manager stripe size. Such a setting causes more disks to be used.

Using Write Behind

The write behind feature enables the operating system to group write Input-Output together, up to the size of a partition. You can improve performance by doing this, because the number of Input-Output operations is reduced. The file system divides each file into 16 KB partitions to increase write performance, limit the number of dirty pages in memory, and minimize disk fragmentation. The pages of a particular partition are not written to disk until the program writes the first byte of the next 16 KB partition. To set the size of the buffer for write behind to eight 16 KB partitions, enter the following command:

```
à# /usr/sbin/vmo -o numclust=8
```

To disable write behind, enter the following command:

```
à# /usr/sbin/vmo -o numclust=0
```

Tuning Sequential Read Ahead

> **Note:** The information in this section applies only to file systems, and only when neither Direct Input-Output nor Concurrent Input-Output are used.

The VMM anticipates the need for pages of a sequential file. It observes the pattern in which a process accesses a file. When the process accesses two consecutive pages of the file, the VMM assumes that the program continues to access the file sequentially, and schedules additional sequential reads of the file. These reads overlap the program processing and make data available to the program faster. The following VMM thresholds, implemented as kernel parameters, determine the number of pages it reads ahead:

- `minpgahead`

  This parameter stores the number of pages read ahead when the VMM first detects the sequential access pattern.

- `maxpgahead`

  This parameter stores the maximum number of pages that VMM reads ahead in a sequential file.

Set the `minpgahead` and `maxpgahead` parameters to appropriate values for an application. The default values are 2 and 8 respectively. Use the `/usr/sbin/vmo` command to change these values. You can use higher values for the `maxpgahead` parameter in systems where the sequential performance of striped logical volumes is of paramount importance. To set the `minpgahead` parameter to 32 pages and the `maxpgahead` parameter to 64 pages, run the following command as the `root` user:

```
-o minpgahead=32 -o maxpgahead=64
```

Set both the `minpgahead` and `maxpgahead` parameters to a power of two. For example, 2, 4, 8, . . . 512, 1042, . . . and so on.

Tuning Disk Input-Output Pacing

Disk Input-Output pacing is an AIX mechanism that enables the system administrator to limit the number of pending Input-Output requests to a file. This prevents disk Input-Output intensive processes from saturating the CPU. Therefore, the response time of interactive and CPU-intensive processes does not deteriorate.

You can achieve disk Input-Output pacing by adjusting two system parameters: the high-water mark and the low-water mark. When a process writes to a file that already has a pending high-water mark Input-Output request, the process is put to sleep. The process wakes up when the number of outstanding Input-Output requests falls lower than or equals the low-water mark.

You can use the `smit` command to change the high and low-water marks. Determine the water marks through trial-and-error. Use caution when setting the water marks, because they affect performance. Tuning the high and low-water marks has less effect on disk Input-Output larger than 4 KB.

You can determine disk Input-Output saturation by analyzing the result of `iostat`, in particular, the percentage of `iowait` and `tm_act`. A high `iowait` percentage combined with high `tm_act` percentages on specific disks is an indication of disk

saturation. Note that a high `iowait` alone is not necessarily an indication of an Input-Output bottleneck.

Resilvering with Oracle Database

If you disable mirror write consistency for an Oracle data file allocated on a raw logical volume, then the Oracle Database fail recovery process uses resilvering to recover after a system failure. This resilvering process prevents database inconsistencies or corruption.

During fail recovery, if a data file is allocated on a logical volume with multiple copies, then the resilvering process performs a checksum on the data blocks of all the copies. It then performs one of the following:

- If the data blocks in a copy have valid checksums, then the resilvering process uses that copy to update the copies that have invalid checksums.
- If all copies have blocks with invalid checksums, then the resilvering process rebuilds the blocks using information from the redo log file. It then writes the data file to the logical volume and updates all the copies.

On AIX, the resilvering process works only for data files allocated on raw logical volumes for which minor write consistency is disabled. Resilvering is not required for data files on mirrored logical volumes with minor write consistency enabled, because minor write consistency ensures that all copies are synchronized.

If the system fails while you are upgrading an earlier release of Oracle Database that used data files on logical volumes for which minor write consistency was disabled, then run the `syncvg` command to synchronize the mirrored logical volume before starting Oracle Database. If you do not synchronize the mirrored logical volume before starting the database, then Oracle Database may read incorrect data from an logical volume copy.

Note: If a disk drive fails, then resilvering does not occur. You must run the `syncvg` command before you can reactivate the logical volume.

Caution: Oracle supports resilvering for data files only. Do not disable minor write consistency for redo log files.

CPU Scheduling and Process Priorities

The CPU is another system component for which processes may contend. Although the AIX kernel allocates CPU effectively most of the time, many processes compete for CPU cycles. If the system has multiple CPU (SMP), then there may be different levels of contention on each CPU.

The following sections provide information about CPU scheduling and process priorities:

- Changing Process Running Time Slice
- Using Processor Binding on SMP Systems

Changing Process Running Time Slice

The default value for the run-time slice of the AIX RR dispatcher is ten milliseconds (msec). Use the `schedo` command to change the time slice. A longer time slice causes a lower context switch rate if the average voluntary switch rate of the applications is lower. As a result, fewer CPU cycles are spent on context-switching for a process and the system throughput should improve.

However, a longer run-time slice can deteriorate response time, especially on a uniprocessor system. The default run-time slice is usually acceptable for most applications. When the run queue is high and most of the applications and Oracle shadow processes are capable of running a much longer duration, you may want to increase the time slice by entering the following command:

```
f # /usr/sbin/schedo -t n
```

In the preceding command, setting *n* to 0 results in a slice of 10 msec, choosing a value of 1 results in a slice of 20 msec, choosing a value of 2 results in a slice of 30 msec, and so on.

Using Processor Binding on SMP Systems

Binding certain processes to a processor can improve performance substantially on an SMP system. Processor binding is available and fully functional on AIX 5L.

However, starting with AIX 5L, specific improvements in the AIX scheduler enables Oracle Database processes to be scheduled optimally without processor binding. Therefore, Oracle no longer recommends binding processes to processors when running on AIX 5L version 5.3 or later.

Setting the AIXTHREAD_SCOPE Environment Variable

Threads in AIX can run with process-wide contention scope (M:N) or with systemwide contention scope (1:1). The `AIXTHREAD_SCOPE` environment variable controls which contention scope is used.

The default value of the `AIXTHREAD_SCOPE` environment variable is `P`, which specifies process-wide contention scope. When using process-wide contention scope, Oracle threads are mapped to a pool of kernel threads. When Oracle is waiting on an event and its thread is swapped out, it may return on a different kernel thread with a different thread ID. Oracle uses the thread ID to post waiting processes, so it is important for the thread ID to remain the same. When using systemwide contention scope, Oracle threads are mapped to kernel threads statically, one to one. For this reason, Oracle recommends that you use systemwide contention. The use of systemwide contention is especially critical for Oracle Real Application Clusters (Oracle RAC) instances.

In addition, on AIX 5L version 5.3 or later, if you set systemwide contention scope, then significantly less memory is allocated to each Oracle process.

Oracle recommends that you set the value of the `AIXTHREAD_SCOPE` environment variable to `S` in the environment script that you use to set the `ORACLE_HOME` or `ORACLE_SID` environment variables for an Oracle Database instance or an Oracle Net listener process as follows:

- Bourne, Bash, or Korn shell:

  Add the following line to the `~/.profile` or `/usr/local/bin/oraenv` script:

  ```
  AIXTHREAD_SCOPE=S; export AIXTHREAD_SCOPE
  ```

- C shell:

  Add the following line to the `~/.login` or `/usr/local/bin/coraenv` script:

  ```
  setenv AIXTHREAD_SCOPE S
  ```

Doing this enables systemwide thread scope for running all Oracle processes.

Network Information Service external naming support

Network Information Service external naming adapter is supported on AIX. To configure and use Network Information Service external naming, refer to the "Configuring External Naming Methods" section of *Oracle Database Net Services Administrator's Guide*.

Simultaneous Multithreading on AIX 5.3 and Later Versions

If Simultaneous Multithreading is enabled, and AIX 5.3 and later version operating system is used, then the `v$osstat` view reports 2 additional rows corresponding to the online logical (`NUM_LCPUS`) and virtual cpus (`NUM_VCPUS`).

If Oracle is run on AIX 5.3 and later versions without Simultaneous Multithreading, then these rows are not reported.

Configuring IBM JSSE Provider with Oracle JDBC Thin Driver

If you want to configure SSL on IBM JDK, then you may face the following issues:

- IBM JSSE does not support `SSLv2Hello` SSL protocol

  For SSL clients using Thin JDBC connectors, you must set `oracle.net.ssl_version` system property to select `TLSv1` SSL protocol or `SSLv3` SSL protocol

- IBM JSSE does not allow anonymous ciphers

  For SSL clients using anonymous ciphers, you must replace the Default Trust Manager with a Custom Trust Manager that accepts anonymous ciphers.

  See Also: IBM JSSE documentation for more information on creating and installing Custom Trust Manager

B

Administering Oracle Database on HP-UX

This appendix provides information about administering Oracle Database on HP-UX. It contains the following topics:

- HP-UX Shared Memory Segments for an Oracle Instance
- HP-UX SCHED_NOAGE Scheduling Policy
- Lightweight Timer Implementation
- Asynchronous Input-Output
- Large Memory Allocations and Oracle Database Tuning
- CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration
- Network Information Service external naming support
- Activating and Setting Expanded Host Names and Node Names

HP-UX Shared Memory Segments for an Oracle Instance

When an Oracle Database instance starts, it creates memory segments by dividing the shared memory allocated for creating the Oracle System Global Area (SGA) by the value of the HP-UX `shmmax` kernel parameter. For example, if 64 GB of shared memory is allocated for a single Oracle instance and the value of the `shmmax` parameter is 1 GB, then Oracle Database creates 64 shared memory segments for that instance.

Performance degradation can occur when an Oracle instance creates multiple shared memory segments. This is because each shared memory segment receives a unique protection key when Oracle Database creates the instance. The number of protection keys available depends on the system architecture as shown in the following table:

Architecture	Number of Protection Keys
PA-RISC	6
Itanium	14

If the Oracle instance creates more shared memory segments than the number of protection keys, then the HP-UX operating system displays protection key faults.

Oracle recommends that you set the `shmmax` parameter value to the amount of available physical memory on the system. Doing this ensures that the entire shared memory for a single Oracle instance is assigned to one shared memory segment and the instance requires only one protection key.

To display the list of active shared memory segments on the system, run the following command:

```
$ ipcs -m
```

If Oracle Database creates more segments for the instance than the number of protection keys, then increase the value of the `shmmax` kernel parameter.

> **See Also:** *Oracle Database Installation Guide* for more information about the recommended minimum kernel parameter values

HP-UX SCHED_NOAGE Scheduling Policy

On HP-UX, most processes use a time-sharing scheduling policy. Time sharing can have detrimental effects on Oracle performance by descheduling an Oracle process during critical operations, for example, when it is holding a latch. HP-UX has a modified scheduling policy, referred to as `SCHED_NOAGE`, that specifically addresses this issue. Unlike the normal time-sharing policy, a process scheduled using `SCHED_NOAGE` does not increase or decrease in priority, nor is it preempted.

This feature is suited to online transaction processing environments because online transaction processing environments can cause competition for critical resources. The use of the `SCHED_NOAGE` policy with Oracle Database can increase performance by 10 percent or more in online transaction processing environments.

The `SCHED_NOAGE` policy does not provide the same level of performance gains in decision support environments because there is less resource competition. Because each application and server environment is different, you should test and verify that the environment benefits from the `SCHED_NOAGE` policy. When using `SCHED_NOAGE`, Oracle recommends that you exercise caution in assigning highest priority to Oracle processes. Assigning highest `SCHED_NOAGE` priority to Oracle processes can exhaust CPU resources on the system, causing other user processes to stop responding.

Enabling SCHED_NOAGE for Oracle Database

To permit Oracle Database to use the `SCHED_NOAGE` scheduling policy, the OSDBA group (typically, the `dba` group) must have the `RTSCHED` and `RTPRIO` privileges to change the scheduling policy and set the priority level for Oracle processes. To give the `dba` group these privileges:

1. Log in as the `root` user.
2. Using any text editor, open the `/etc/privgroup` file, or create it if necessary.
3. Add or edit the following line, which begins with the name of the OSDBA group, specifying the privileges `RTPRIO` and `RTSCHED` that you want to grant to this group every time the system restarts:

   ```
   dba RTPRIO RTSCHED
   ```

4. Save the file, and quit the text editor.
5. Enter the following command to grant the privileges to the OSDBA group:

   ```
   # /usr/sbin/setprivgrp -f /etc/privgroup
   ```

6. Enter the following command to verify that the privileges are set correctly:

   ```
   # /usr/bin/getprivgrp dba
   ```

Add the HPUX_SCHED_NOAGE initialization parameter to the parameter file for each instance, setting the parameter to an integer value to specify process priority levels. The supported range of values is 178 to 255. Lower values represent higher priorities. The default value of HPUX_SCHED_NOAGE initialization parameter is 178 for all Oracle processes. For LMS processes, this can be changed by setting the initialization parameter _os_sched_high_priority. If the parameter _os_sched_high_priority is between 31 and 2, LMS processes run with SCHED_RR at the set priority. If the parameter value is between 178 and 255, the processes run at the set value with SCHED_NOAGE. However, LMS doesl not run at priority level less than the value of HPUX_SCHED_NOAGE.

If the HPUX_SCHED_NOAGE parameter setting is out of range, then Oracle Database automatically sets the parameter to a permissible value and continues with the SCHED_NOAGE policy with the new value. It also generates a message in the alert_*sid*.log file about the new setting. Whenever the highest priority is assigned to Oracle processes, either by the user or by automatic readjustment, Oracle Database generates a message in the alert_*sid*.log file warning about the possibility of exhaustion of CPU resources on the system. Oracle recommends that you set the parameter to assign the priority level you want for Oracle processes.

> **See Also:** The HP-UX documentation, the rtsched(1) man page, and the rtsched(2) man page for more information about priority policies and priority ranges

Lightweight Timer Implementation

With Oracle Database 11*g*, you can collect run-time statistics always if the dynamic initialization parameter STATISTICS_LEVEL is set to TYPICAL (default) or ALL. This parameter setting implicitly sets the TIMED_STATISTICS initialization parameter to true. Oracle Database on HP-UX systems uses the gethrtime() system library call to calculate elapsed times during the collection of the statistics. The use of this lightweight system library call enables you to collect run-time statistics always while running an Oracle instance, without affecting performance.

This system library call can provide a performance improvement of up to 10 percent over an Oracle release that does not use the gethrtime() system library call when the TIMED_STATISTICS initialization parameter is explicitly set to true. In addition, there is no negative impact on the online transaction processing performance of an Oracle Database while using the gethrtime() system library call to collect run-time statistics always.

Asynchronous Input-Output

The asynchronous Input-Output pseudo-driver on HP-UX enables Oracle Database to perform Input-Output to raw disk partitions using an asynchronous method, resulting in less Input-Output overhead and higher throughput. You can use the asynchronous Input-Output pseudo-driver on both HP-UX servers and workstations.

This section contains the following topics:

- MLOCK Privilege
- Implementing Asynchronous Input-Output
- Verifying Asynchronous Input-Output
- Asynchronous Flag in SGA

MLOCK Privilege

To permit Oracle Database to process asynchronous Input-Output operations, the OSDBA group (`dba`) must have the `MLOCK` privilege. To give the `dba` group the `MLOCK` privilege:

1. Log in as the `root` user.
2. Using any text editor, open the `/etc/privgroup` file, or create it if necessary.
3. Add or edit the following line, which begins with the name of the OSDBA group, specifying the privilege `MLOCK`:

   Note: You must use only one line to specify the privileges for a particular group in this file. If the file already contains a line for the `dba` group, then add the `MLOCK` privilege on the same line.

   ```
   dba RTPRIO RTSCHED MLOCK
   ```

4. Save the file, and quit the text editor.
5. Enter the following command to grant the privileges to the OSDBA group:

   ```
   # /usr/sbin/setprivgrp -f /etc/privgroup
   ```

6. Enter the following command to verify that the privileges are set correctly:

   ```
   # /usr/bin/getprivgrp dba
   ```

Implementing Asynchronous Input-Output

To use asynchronous Input-Output on HP-UX, you must use an Automatic Storage Management disk group that uses raw partitions as the storage option for database files.

> **See Also:** *Oracle Database Installation Guide* for more information about configuring Automatic Storage Management and raw logical volumes on HP-UX systems

Before you can implement asynchronous Input-Output with either storage option, you must use the System Administrator Management utility to configure the asynchronous disk driver into the HP-UX kernel.

> **Note:** In Oracle Database 11*g* Release 1, you did not have to set `DISK_ASYNCH_IO` paramter to `FALSE` on a file system. However, starting with Oracle Database 11*g* Release 2, if database uses file system for storing the database files, then ensure that you set initialization parameter `DISK_ASYNCH_IO` to `FALSE`. By default the value of `DISK_ASYNCH_IO` is `TRUE`.
>
> The `DISK_ASYNCH_IO` parameter must be set to `TRUE` only when raw partitions are used for storing database files.

To add the asynchronous disk driver and configure the kernel by using the System Administrator Management utility:

1. Run the following command as the `root` user:

```
# sam
```

2. Select the **Kernel Configuration area**.
3. Select the **Drivers area**.
4. Select the **asynchronous disk driver** (`asyncdsk`).
5. Select **Actions**, and then select **Add Driver to Kernel**.
6. Select **List**, and then select **Configurable Parameters**.
7. Select the `MAX_ASYNC_PORTS` parameter.
8. Select **Action**, and then select **Modify Configurable Parameter.**
9. Specify a new value for the parameter, using the following guidelines, and then click **OK**.

   The `MAX_ASYNC_PORTS` parameter is a configurable HP-UX kernel parameter that controls the maximum number of processes that can open the `/dev/async` file simultaneously.

   The system displays an error message when a process tries to open the `/dev/async` file after the maximum number of processes have opened the file. This error can reduce performance on systems with a large number of shadow processes or many parallel query slaves performing asynchronous Input-Output. This error is not recorded. To avoid this error, estimate the highest likely number of processes that can access the `/dev/async` file and set the `MAX_ASYNC_PORTS` parameter to this value.
10. Select **Actions**, and then select **Process a New Kernel**.
11. Select one of the following options, and then click **OK**:
    - **Move Kernel Into Place and Shutdown System/Reboot Now**
    - **Do Not Move Kernel Into Place: Do Not Shutdown/Reboot Now**

    If you choose the second option, then the new kernel, `vmunix_test`, and the `system.SAM` configuration file used to create it, are both created in the `/stand/build` directory.

To enable asynchronous Input-Output operations using the HP-UX asynchronous device driver:

1. Log in as the `root` user.
2. If `/dev/async` does not exist, use the following command to create it:

   ```
   # /sbin/mknod /dev/async c 101 0x0
   ```

   By default, the minor number is set to 0. The following table describes the various minor numbers for 8-bit that you can use to create a device file:

Minor number	Description
0x0	This is the HP-UX default value for `/dev/async`.
0x4	Enable disc device timeouts to complete with an error code rather than retrying forever. This setting is necessary for application-level disc mirroring, so as to avoid the situation where the application waits forever for a failed disc device to be repaired. Oracle RDBMS users should enable this feature when Automatic Storage Management mirroring/replication (internal redundancy) is used. SGA will be locked in memory.

Minor number	Description
0x100	Enable on-demand locking of memory pages by async driver when asyncdsk_open(2) is called. A low-overhead routine is then used to lock a page into memory during I/O operations. On-demand locking is critically important when using Oracle's Automatic Memory Management feature (the use of `MEMORY_TARGET` in the `init.ora` file to control memory usage). RDBMS deployments utilizing dynamic nPar or dynamic vPar features should also configure on-demand locking. More traditional RDBMS deployments can consider on-demand locking in light of its more obvious effects. Generally speaking, RDBMS startup will be quicker because the complete SGA is not locked into memory immediately. However, some instances will experience a slight run-time performance penalty with on-demand locking as memory pages are dynamically locked/unlocked for each I/O request.
0x104	This is a combination of 0x100 and 0x4. Both the features are enabled.

3. Enter the following command to verify that the `/dev/async` device file exists and has the major number 101:

   ```
   # ls -l /dev/async
   ```

   The output of this command should look similar to the following:

   ```
   crw------- 1 oracle   dba     101 0x000000 Oct 28 10:32  /dev/async
   ```

4. If required, give the device file the operating system owner and permissions consistent with those of the Oracle software owner and OSDBA group.

   If the Oracle software owner is `oracle` and the OSDBA group is `dba`, then run the following commands:

   ```
   # /usr/bin/chown oracle:dba /dev/async
   # /usr/bin/chmod 660 /dev/async
   ```

Verifying Asynchronous Input-Output

To verify asynchronous Input-Output, first verify that the HP-UX asynchronous driver is configured for Oracle Database, then verify that Oracle Database is executing asynchronous Input-Output through the HP-UX device driver:

- Verifying That HP-UX Asynchronous Driver is Configured for Oracle Database
- Verifying that Oracle Database is Using Asynchronous Input-Output

Verifying That HP-UX Asynchronous Driver is Configured for Oracle Database

To verify that the HP-UX asynchronous driver is configured properly for Oracle Database:

1. Start Oracle Database with a few parallel query slave processes.
2. Start the GlancePlus/UX utility as follows:

   ```
   $ gpm
   ```

3. In the main window, click **Reports** and then click **Process List**.
4. In the Process List window, select one parallel query slave process, select **Reports**, and then select **Process Open Files**.

   The list of files currently opened by the parallel query slave process is displayed.

5. In the list of open files, check for the `/dev/async` file or the 101 0x104000 mode.

   If either is in the list, then the `/dev/async` file has been opened by the parallel query slave process, and the HP-UX asynchronous device driver is configured properly to enable Oracle processes to run asynchronous Input-Output. Make a note of the file descriptor number for the `/dev/async` file.

Verifying that Oracle Database is Using Asynchronous Input-Output

To verify that Oracle Database is using asynchronous Input-Output through the HP-UX asynchronous device driver:

1. Attach the HP-UX `tusc` utility to the same Oracle parallel query slave that you selected in GlancePlus in the preceding procedure.
2. Run an Input-Output bound query in the environment.
3. Check the pattern of read and write calls in the `tusc` output.

   You can do this, for example, by entering the following command, where *`pid`* is the process ID of a parallel query slave supposed to process asynchronous Input-Output:

   ```
   $ tusc -p pid > tusc.output
   ```

4. After running the query, press **Ctrl+c** to disconnect from the process, and then open the `tusc.output` file.

   The following example shows a sample `tusc.output` file:

   ```
   ( Attached to process 2052: "ora_p000_tpch" [ 64-bit ])
   ..................
   ......................
   [2052] read(9, "80\0\001\013  \b\0\0\0\0\0\0\0\0".., 388) .. = 28
   [2052] write(9, "\0\0\00e\0\0\0\080\0\001\013Ð \0".., 48) .. = 48
   [2052] read(9, "80\0\001\013¢ 18\0\0\0\0\0\0\0\0".., 388) .. = 28
   [2052] write(9, "\0\0\00e\0\0\0\080\0\001\01bd4\0".., 48) .. = 48
   ```

   If the `DISK_ASYNCH_IO` initialization parameter is not explicitly set to `false` (set to `true` by default), then the `tusc.output` file shows a pattern of asynchronous read/write calls of the same file descriptor (9 in the preceding example) back to back.

   Map the file descriptor number in the `tusc.output` file to that used by `/dev/async` file in GlancePlus. They should match for the particular parallel query slave process. This verifies that Input-Output through the HP-UX asynchronous driver is asynchronous. With synchronous Input-Output, or if the `DISK_ASYNCH_IO` initialization parameter is explicitly set to `FALSE`, you do not see the asynchronous read/write pattern described previously. Instead, you see calls to `lseek` or `pread/pwrite`. You also see many different file descriptors (the first argument to read/write) instead of just a single file descriptor.

Asynchronous Flag in SGA

Oracle Database on HP-UX uses a nonblocking polling facility provided by the HP-UX asynchronous driver to check the status of Input-Output operations. This polling is performed by checking a flag that is updated by the asynchronous driver based on the status of the Input-Output operations submitted. HP-UX requires that this flag be in shared memory.

Oracle Database configures an asynchronous flag in the SGA for each Oracle process. Oracle Database on HP-UX has a true asynchronous Input-Output mechanism where

Input-Output requests can be issued even though some previously issued Input-Output operations are not complete. This helps to enhance performance and ensures good scalability of parallel Input-Output processes.

Releases of Oracle Database earlier than release 8.1.7 were able to run Input-Output operations only from shared memory by using the HP-UX asynchronous driver. Oracle Database 11*g* runs Input-Output operations from both shared memory and process-private regions using the new HP-UX asynchronous driver. However, Input-Output operations through the asynchronous driver are not asynchronous in nature. This is because Oracle Database must perform a blocking wait to check the status of Input-Output operations submitted to the asynchronous driver. This causes some Oracle processes, such as the database writer process, to essentially process synchronous Input-Output.

Large Memory Allocations and Oracle Database Tuning

Applications running on Oracle Database 11*g* can use significantly more memory than applications running on earlier releases. This is because Oracle Database 11*g* changes the default setting for virtual memory data pages from D (4KB) to L (4 GB) on HP-UX systems.

This section contains the following topics:

- Default Large Virtual Memory Page Size
- Tuning Recommendations
- Tunable Base Page Size

Default Large Virtual Memory Page Size

By default, Oracle Database uses the largest virtual memory page size setting available on HP-UX for allocating process-private memory. It is defined by the value `L` (largest.) This value is set as one of the `LARGE_PAGE_FLAGS` options when linking an Oracle executable.

When the virtual memory page size is set to L, HP-UX allocates the available process-private memory to pages of 1 MB, 4 MB, 16 MB and so on, until it reaches the 1 GB limit, or until it reaches the total amount of allocated memory. If you allocate enough memory to the Oracle PGA for the operating system to be able to allocate memory in larger data page size units, then the operating system allocates the maximum page size at once. For example, if you allocate 48 MB for the Oracle PGA, then the system can have either 3 pages each of 16 MB, or a series of pages in unit sizes with the smaller multiples. For example, four 1 MB pages, three 4 MB pages, and two 16 MB pages. If you allocate 64 MB to the PGA, then the operating system allocates one page of 64 MB, as the data page unit size matches the available memory.

In general, large memory pages yield better application performance by reducing the number of virtual memory translation faults that must be handled by the operating system, freeing more CPU resources for the application. Large pages help to reduce the total number of data pages required to allocate the process-private memory. This reduction decreases the chances of translation lookaside buffer misses at the processor level.

However, if applications are constrained in memory and tend to run a very large number of processes, then this drastic page size increase may lead processes to indicate large memory allocations, followed by an `Out of memory` error message. If this happens, then you must lower the page size to a value between the D (default) size of 4 KB and the `L` (largest) size of 4 GB.

With the lowest page size setting (4 KB), CPU utilization can be 20 percent higher than that with a larger page size setting. With the highest setting of `L`, the memory utilization can be 50 percent higher than that with a 4 MB setting. In cases where the system shows memory constraints, Oracle recommends that you set the page size to match the requirements of the particular application, within the constraints of available memory resources.

For example, an application that has problems with the `L` setting may show reasonable performance with a 4 MB virtual memory page setting.

Tuning Recommendations

To address tuning for the increased memory allocation required for persistent private SQL areas and large virtual memory page sizes, Oracle recommends that you decrease the virtual memory data page size for Oracle Database as required. Use the following command to alter the page size setting:

```
# /usr/bin/chatr +pd newsize $ORACLE_HOME/bin/oracle
```

In the preceding example, *`newsize`* represents the new value of the virtual memory page size.

Display the new setting using the `chatr` command as follows:

```
# /usr/bin/chatr $ORACLE_HOME/bin/oracle
```

Tunable Base Page Size

A large base page size enables efficient memory management. The default value for `base_pagesize` is 4 KB. The new feature introduced with HP-UX 11.31 enables to adjust the size of the base page, by invoking `kctune` (1 M) to change the tunable `base_pagesize` and then restart the computer.

Note: The base page size feature is not supported on HP-UX PA RISC operating system.

CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration

HP-UX 11*i* supports dynamic run-time reconfiguration of processor sets and dynamic reassignment of workload between processor sets by valid users.

HP-UX Virtual Partitions enable users to configure their systems in multiple logical partitions where each partition is assigned its own set of processors, memory, and Input-Output resources, and can run a separate instance of the HP-UX operating system. HP-UX processor sets integrated with vPars support dynamic processor migration from one virtual partition to another without requiring a restart of any virtual partition. This helps to provide efficient resource partitioning between applications to minimize interference and guarantees necessary resource allocation to each application running on the HP-UX server.

See Also: Refer to the "System and Database Changes" section in the *Oracle Database Concepts* guide for more information on dynamic resource provisioning

Network Information Service external naming support

Network Information Service external naming adapter is supported on HP-UX. To configure and use Network Information Service external naming, refer to the "Configuring External Naming Methods" section of *Oracle Database Net Services Administrator's Guide*.

Activating and Setting Expanded Host Names and Node Names

The system node names and host names have default length limits of 8 and 64 bytes. The system administrator can configure the system to expand both these limits to 255 bytes.

A dynamic kernel tunable parameter `expanded_node_host_names`, must be turned on to allow larger names to be set.

To turn on the kernel parameter, run the following command:

```
kctune expanded_node_host_names=1
```

To turn off the kernel parameter, run the following command:

```
kctune expanded_node_host_names=0
```

C

Administering Oracle Database on Linux

This appendix contains information about administering Oracle Database on Linux. From Oracle

It contains the following topics:

- Extended Buffer Cache Support
- Using hugetlbfs on SUSE Linux Enterprise Server 10 or Red Hat Enterprise Linux 4
- Asynchronous Input-Output Support
- Simultaneous Multithreading
- Allocating Shared Resources
- Database Migration from 32-Bit Linux to 64-Bit Linux

Note: Starting with Oracle Database 11*g* release 2 (11.2), Linux x86-64 media does not contain Linux x86 binaries. You must use Linux x86 media to install 32-bit Oracle home.

Extended Buffer Cache Support

Note: This section applies to Linux x86 only.

Oracle Database can allocate and use more than 4 GB of memory for the database buffer cache. This section describes the limitations and requirements of the extended buffer cache feature on Linux x86 systems.

See Also: Very Large Memory on Linux x86 for more information.

See Also: *Oracle Database Concepts* for more information about the extended buffer cache feature.

In-Memory File System

To use the extended buffer cache feature, create an in-memory file system on the `/dev/shm` mount point equal in size or larger than the amount of memory that you intend to use for the database buffer cache. For example, to create an 8 GB file system on the `/dev/shm` mount point:

1. Run the following command as the `root` user:

```
# mount -t tmpfs shmfs -o size=8g /dev/shm
```

2. To 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:

```
shmfs /dev/shm tmpfs size=8g 0 0
```

When Oracle Database starts with the extended buffer cache feature enabled, it creates a file in the `/dev/shm` directory that corresponds to the Oracle buffer cache.

> **Note:** If an in-memory file system is already mounted on the `/dev/shm` mount point, then ensure that its size equals or is larger than the amount of memory that is used for the database buffer cache.

USE_INDIRECT_DATA_BUFFERS Initialization Parameter

To enable the extended buffer cache feature, set the `USE_INDIRECT_DATA_BUFFERS` initialization parameter to TRUE in the parameter file. This enables Oracle Database to specify a larger buffer cache.

Dynamic Cache Parameters

If the extended cache feature is enabled, then you must use the `DB_BLOCK_BUFFERS` parameter to specify the database cache size.

Do not use the following dynamic cache parameters while the extended buffer cache feature is enabled:

- `DB_CACHE_SIZE`
- `DB_2K_CACHE_SIZE`
- `DB_4K_CACHE_SIZE`
- `DB_8K_CACHE_SIZE`
- `DB_16K_CACHE_SIZE`

Limitations

The following limitations apply to the extended buffer cache feature:

- You cannot create or use tablespaces with nondefault block sizes. You can create tablespaces using only the block size specified by the `DB_BLOCK_SIZE` parameter.
- You cannot change the size of the buffer cache while the instance is running.

> **See Also:** *Oracle Database SQL Language Reference* for more information about the default block size used by the `CREATE TABLESPACE` command.

Note: The default VLM window size is 512 MB. This memory size is allocated to the address space of the process. To increase or decrease this value, set the `VLM_WINDOW_SIZE` environment variable to the new size in bytes. For example, to set the `VLM_WINDOW_SIZE` to 256 MB, run the following command:

```
$ export VLM_WINDOW_SIZE=268435456
```

The value that you specify for the `VLM_WINDOW_SIZE` environment variable must be a multiple of 64 KB.

Using hugetlbfs on SUSE Linux Enterprise Server 10 or Red Hat Enterprise Linux 4

To enable Oracle Database to use large pages (sometimes called huge pages) on SUSE Linux Enterprise Server 10, or Red Hat Enterprise Linux 4, 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 a enough 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 GB / 2 MB = 819.2).

See Also: Overview of HugePages for more information on tuning SGA with hugepages, configuring hugepages, and hugepages configuration restrictions.

Asynchronous Input-Output Support

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_ASYNCH_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 dNFS, asynchronous Input-Output is enabled by default. There is no need to set FILESYSTEMIO_OPTIONS to enable asynchronous Input-Output in these environments.

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`).

Allocating Shared Resources

To use the `MEMORY_TARGET` or `MEMORY_MAX_TARGET` feature, the following kernel parameters must be potentially 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 4GB 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:

    ```
    # shmfs /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.

Database Migration from 32-Bit Linux to 64-Bit Linux

To migrate an Oracle Database 11*g* Release 2 (11.2) for 32-bit Linux to an Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux, you must perform the following steps:

- Online Backup of Database With RMAN
- Migrating 32-Bit Linux Database to 64-Bit Linux Database
- Migrating Data To and From ASM

Online Backup of Database With RMAN

Online backup enables to take a backup of the database without having to shutdown the database. To achieve this, perform the following steps:

1. Connect to the database instance as SYSDBA:

```
SQL> CONNECT / AS SYSDBA;
```

2. Run the following commands to ensure that the database is in ARCHIVELOG mode:

```
SQL> SHUTDOWN IMMEDIATE
Database closed
Database dismounted
Oracle instance shutdown

SQL> STARTUP MOUNT
Oracle instance started

Total System Global Area  272629760 bytes
Fixed Size                   788472 bytes
Variable Size             103806984 bytes
Database Buffers          167772160 bytes
Redo Buffers                 262144 bytes
Database mounted

SQL>ALTER DATABASE ARCHIVELOG;
Database altered

SQL> ALTER DATABASE OPEN;
Database altered
```

3. Run the following command to start RMAN, which is located under $ORACLE_HOME/bin directory:

```
RMAN>connect target
```

4. To backup the 32-bit database and all the archived redo log files, run the following command:

```
RMAN>backup database plus archivelog delete input;
```

Note: Archive redo logs are very important to recover the database. Oracle recommends that you back them up along with your database. You can backup the archive redo logs from time to time by issuing the following command:

```
RMAN>backup archivelog all delete input;
```

Migrating 32-Bit Linux Database to 64-Bit Linux Database

This section covers the following topics:

- Migrating 32-Bit Database to 64-Bit Database With the Same Directory Structure for Data Files
- Migrating 32-Bit Database to 64-Bit Database With Different Directory Structure for Data Files

Migrating 32-Bit Database to 64-Bit Database With the Same Directory Structure for Data Files

If the control file, data file, redo log files have the same structure on the target computer as in the source computer, then perform the following steps:

1. Perform an online backup of the database before starting the migration process. Refer to Online Backup of Database With RMAN on page C-5 for more information.
2. Install Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux in a new Oracle Database home. It is recommended that you use the same version of Oracle Database home as on the 32-bit computer.

   See Also: *Oracle Database Installation Guide for Linux*

3. Copy `init.ora`, data files, control file, and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer.
4. Edit `init.ora` file to include the following changes:
   - Update the memory requirements included in the file.
   - Edit the control file location if necessary.

   Note: Oracle recommends that you double the values of shared pool, java pool, `sga_target` and large pool listed in the `init.ora` file.

5. Connect to the database instance as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA;
   ```

6. Set the system to spool results to a log file for later verification of success. For example:

   ```
   SQL> SPOOL /tmp/utlirp.log
   ```

7. Start the 64-bit Oracle Database as follows:

   ```
   SQL> STARTUP UPGRADE pfile=init.ora;
   ```

Note: *Oracle Database Upgrade Guide* for more information on changing from 32-bit to 64-bit

8. Run the following command on the 64-bit Oracle Database, to invalidate all the PL/SQL modules:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlirp.sql
```

9. Shut down the 64-bit Oracle Database:

```
SQL>shutdown immediate;
```

10. Start the 64-bit Oracle Database:

```
SQL> STARTUP pfile=init.ora;
```

11. Revalidate all the existing PL/SQL modules in the format required by the 64-bit Oracle Database:

```
SQL>$ORACLE_HOME/rdbms/admin/utlrp.sql;
```

Migrating 32-Bit Database to 64-Bit Database With Different Directory Structure for Data Files

If the control file, data file, redo log files have different structure on the target computer as compared to the source computer, then perform the following steps:

1. Perform an online backup of the database before starting the migration process. Refer to Online Backup of Database With RMAN on page C-5 for more information.
2. Install Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux in a new Oracle Database home. It is recommended that you use the same version of Oracle Database home as on the 32-bit computer.

   See Also: *Oracle Database Installation Guide for Linux*

3. Edit `init.ora` file on the 64-bit computer to include the following changes:
   - Update the memory requirements included in the file.
   - The `init.ora` file still contains the 32-bit control file path. You must manually update `control_files` parameter value to include the 64-bit control file location.

   Note: Oracle recommends that you double the values of shared pool, java pool, `sga_target` and large pool listed in the `init.ora` file.

4. If the 64-bit target computer contains a different structure for data files, then you must re-create the control file or mount database on 64-bit computer. Refer to Re-aligning Data File Path and Name on page C-8 for more information.

   Note: Oracle recommends not to use the `RESETLOGS` option to re-create control files.

5. Set the system to spool results to a log file for later verification of success. For example:

```
SQL> SPOOL /tmp/utlirp.log
```

6. Run the following command on the 64-bit Oracle Database, to invalidate all the PL/SQL modules:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlirp.sql
```

7. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF;
```

8. Shut down the 64-bit Oracle Database:

```
SQL>shutdown immediate;
```

9. Start the 64-bit Oracle Database:

```
SQL> STARTUP pfile=init.ora;
```

10. Revalidate all the existing PL/SQL modules in the format required by the 64-bit Oracle Database:

```
SQL>$ORACLE_HOME/rdbms/admin/utlrp.sql;
```

Re-aligning Data File Path and Name

The following are some methods to realign the data file names and path to point to the correct location:

- Re-creating Control File
- Mounting Database on a 64-Bit Computer

Re-creating Control File

Perform the following steps to re-create the control file:

1. Run the following command to backup the control file to trace. The trace file is located under the `diagnostic_dest` directory on the 32-bit Linux computer. The following command generates a trace file which contains the necessary sql to re-create the control file:

```
SQL> alter database backup controlfile to trace;
```

> **Note:** Ensure that you open the Oracle Database in the `UPGRADE` mode after the control file is created.

2. Rename the trace file generated into `.sql` format on the 32-bit Linux computer. For example:

```
$ cp trace.ora control.sql
```

The contents of the control file are as follows, for example:

- Re-creating control files with `NORESETLOGS` option.

  32-bit control file with `NORESETLOGS` option:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" NORESETLOGS  NOARCHIVELOG
```

```
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l32/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l32/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l32/oracle/dbs/t_db1.f'
     '/ade/aime_l32/oracle/dbs/t_ax1.f'
     '/ade/aime_l32/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE;
ALTER DATABASE OPEN UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l32/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

Now, let us consider the modified 64-bit control file:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" NORESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l64/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l64/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l64/oracle/dbs/t_db1.f'
     '/ade/aime_l64/oracle/dbs/t_ax1.f'
     '/ade/aime_l64/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE;
ALTER DATABASE OPEN UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l64/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

- Re-creating control files with `RESETLOGS` option:

  32-bit control file with `RESETLOGS` option:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l32/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l32/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l32/oracle/dbs/t_db1.f'
     '/ade/aime_l32/oracle/dbs/t_ax1.f'
     '/ade/aime_l32/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE USING BACKUP CONTROLFILE;
ALTER DATABASE OPEN RESETLOGS UPGRADE;
```

```
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l32/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

Now, let us consider the modified 64-bit control file:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l64/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l64/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l64/oracle/dbs/t_db1.f'
     '/ade/aime_l64/oracle/dbs/t_ax1.f'
     '/ade/aime_l64/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE USING BACKUP CONTROLFILE;
ALTER DATABASE OPEN RESETLOGS UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l64/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

3. Based on the method employed to realign the file paths to point to the correct locations, you must copy the necessary files from the source 32-bit Linux computer to the target 64-bit Linux computer:
   - `NORESETLOGS` option: Copy `init.ora`, data files, re-created control files (*`control.sql`*), and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
   - `RESETLOGS` option: Copy `init.ora`, data files, and re-created control files (*`control.sql`*) from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
4. Connect to the database instance as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA;
   ```

5. To change from 32-bit to 64-bit, run the following command from the Linux 64-bit Oracle Database home:

   ```
   sql>set echo on
   sql>@control.sql
   ```

Mounting Database on a 64-Bit Computer

Perform the following steps to mount the database on a 64-bit computer:

1. Copy `init.ora`, data files, control file and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
2. Connect to the database instance as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA;
   ```

3. Start the 64-bit Oracle Database as follows:

   ```
   SQL> STARTUP mount pfile=init.ora;
   ```

4. Update all the 32-bit data file locations to include the 64-bit data file locations. For example:

```
sql> alter database rename file '/ade/aime_l32/oracle/dbs/t_db1.f' to
'/ade/aime_l64/oracle/dbs/t_db1.f'
sql> Database altered.
```

5. Update all the 32-bit log file locations to include the 64-bit log file locations. For example:

```
sql> alter database rename file '/ade/aime_l32/oracle/dbs/t_log.f' to
'/ade/aime_l64/oracle/dbs/t_log.f'
sql> Database altered.
```

6. To change from 32-bit to 64-bit, run the following command from the Linux 64-bit Oracle Database home:

```
sql> ALTER DATABASE OPEN UPGRADE;
```

Note: Refer to *Oracle Database Upgrade Guide* for more information on changing from 32-bit to 64-bit

Migrating Data To and From ASM

To take advantage of Automatic Storage Management (ASM), you can migrate an Oracle 11*g* Release 2 (11.2) database into and out of an ASM disk group using Recovery Manager (RMAN). This migration is performed using RMAN even if you are not using RMAN for your primary backup and recovery strategy.

See Also: Chapter 8, "Performing ASM Data Migration with RMAN", in *Oracle Automatic Storage Management Administrator's Guide* for more information on migrating databases.

D

Administering Oracle Database on Solaris

This appendix contains information about administering Oracle Database on Solaris.

Intimate Shared Memory

On Solaris systems, Oracle Database uses Intimate Shared Memory for shared memory segments because it shares virtual memory resources between Oracle processes. Intimate Shared Memory causes the physical memory for the entire shared memory segment to be locked automatically.

On Solaris 10 systems, dynamic/pageable Intimate Shared Memory 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.

Oracle Database automatically selects Intimate Shared Memory or Dynamic Intimate Shared Memory based on the following criteria:

- Oracle Database uses Dynamic Intimate Shared Memory 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 also uses `DISM` if it is available on the system and `MEMORY_TARGET` or `MEMORY_MAX_TARGET` is set by the user.
- Oracle Database uses Intimate Shared Memory 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 Intimate Shared Memory or Dynamic Intimate Shared Memory, 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 Dynamic Intimate Shared Memory, 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.

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.

Note:

- Do not set the `LOCK_SGA` parameter to `TRUE` in the server parameter file. If you do, then Oracle Database 11*g* cannot start.
- The process name for the `oradism` utility is `ora_dism_sid`, where `sid` is the system identifier. When using Dynamic Intimate Shared Memory, 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.

E

Using Oracle ODBC Driver

This appendix provides information related to using Oracle ODBC Driver. It contains the following sections:

- Oracle ODBC Driver Certification:
- 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

Oracle ODBC Driver Certification:

On Unix platforms, Oracle ODBC driver are certified on the following operating systems using Open Source Driver Manager from unixODBC.org:

- Linux x86-32 32 bit 2.2.14
- Linux x86-64 32,64 bit 2.2.14
- Oracle Solaris on SPARC 32,64 bit 2.2.14
- AIX 5L 5.2 32,64 bit 2.2.12
- Solaris x64 64 bit 2.2.14
- Linux IA64 64 bit 2.2.14
- Linux on Power 32,64 bit 2.2.14
- HP Itanium 32,64 bit 2.2.14

Features Not Supported

Oracle ODBC Driver does not support the following 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	

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" on page E-23 for information about the DATE and TIMESTAMP data types related to performance and tuning

Floating Point Data Types

When connected to an Oracle Database 11*g* Release 2 (11.2) 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.

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.
SQL_LONGVARCHAR and SQL_WLONGVARCHAR	The Oracle limit for SQL_LONGVARCHAR data, where the column type is LONG, is 2,147,483,647 bytes. The Oracle limit for SQL_LONGVARCHAR data, where the column type is CLOB, is 4 gigabytes. The limiting factor is the client workstation memory.
SQL_LONGVARCHAR and SQL_LONGVARBINARY	Oracle Database permits only a single long data column in each table. The long data types are SQL_LONGVARCHAR (LONG) and SQL_LONGVARBINARY (LONG RAW). Oracle recommends that you use CLOB and BLOB columns instead. There is no restriction on the number of CLOB and BLOB columns in a table.

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
DSN	ODBC data source name	User-supplied name This is a mandatory keyword.
DBQ	TNS service name	User-supplied name This is a mandatory keyword.
UID	User ID or user name	User-supplied name This is a mandatory keyword.
PWD	Password	User-supplied name Specify PWD=; for an empty password. This is a mandatory keyword.
DBA	Database attribute	W implies write access R implies read-only access
APA	Applications attributes	T implies that thread safety is to be enabled F implies that thread safety is to be disabled
RST	Result sets	T implies that result sets are to be enabled. F implies that result sets are to be disabled.
QTO	Query timeout option	T implies that query timeout is to be enabled. F implies that query timeout is to be disabled.
CSR	Close cursor	T implies that close cursor is to be enabled. F implies that close cursor is to be disabled.

Keyword	Meaning	Value
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 SQL_WCHAR support	T implies Force SQL_WCHAR is to be enabled. F implies Force SQL_WCHAR 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.
MDI	Set metadata ID default	T implies that the default value of SQL_ATTR_METADATA_ID is SQL_TRUE. F implies that the default value of SQL_ATTR_METADATA_ID is SQL_FALSE.
DPM	Disable SQLDescribeParam	T implies that SQLDescribeParam is to be disabled. F implies that SQLDescribeParam is to be enabled.
BTD	Bind TIMESTAMP as DATE	T implies that SQL_TIMESTAMP is to be bound as Oracle DATE. F implies that SQL_TIMESTAMP is to be bound as Oracle TIMESTAMP.
NUM	Numeric settings	NLS implies that the Globalization Support numeric settings are to be used (to determine the decimal and group separator).

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.

Linking ODBC Applications

When you link the program, you must link it with the Driver Manager library, `libodbc.so`.

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.

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.

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 ="scott";
char *DefPassWord ="tiger";
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);
}
```

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.

Supported Functionality

This sections 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

API Conformance

Oracle ODBC Driver release 10.2.0.1.0 and higher supports all Core, Level 2, and Level 1 functions.

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)
```

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.

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`.

Unicode Support

This section provide information about Unicode support. It contains the following topics:

- Unicode Support Within the ODBC Environment
- Unicode Support in ODBC API
- SQLGetData Performance
- Unicode Samples

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.

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 must 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`

SQL Data Types	ODBC SQL Data Types
NCLOB	SQL_WLONGVARCHAR

SQLGetData Performance

The SQLGetData function allows an ODBC application to specify the data type to receive a column as 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.

Unicode Samples

Because Oracle ODBC Driver itself was implemented using TCHAR macros, it is recommended that ODBC application programs use TCHAR in order 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 E–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("scott"), SQL_NTS, _T("tiger"),
SQL_NTS);
.
.
.
if (conHnd)
  SQLFreeConnect(conHnd);
if (envHnd)
  SQLFreeEnv(envHnd);
```

Example E–2 Simple Retrieval

The following example retrieves the employee names and the job tiles 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 E–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 E–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 E–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 = lstrlen(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 (temination)
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"));
  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"));
}
```

Performance and Tuning

This section contains the following topics:

- General ODBC Programming Guidelines
- Data Source Configuration Options
- DATE and TIMESTAMP Data Types

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.

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 on 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" on page E-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 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.

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.

F

Database Limits

This appendix describes database limits.

Database Limits

Table F–1 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 F–1 CREATE CONTROLFILE and CREATE DATABASE Parameters

Parameter	Default	Maximum Value
`MAXLOGFILES`	16	255
`MAXLOGMEMBERS`	2	5
`MAXLOGHISTORY`	100	65534
`MAXDATAFILES`	30	65534
`MAXINSTANCES`	1	1055

Table F–2 lists the Oracle Database file size limits in bytes.

Table F–2 File Size Limits

File Type	Platform	File size limit
Data files	Any	4,194,303 multiplied by the value of the `DB_BLOCK_SIZE` parameter
Import/Export files and SQL*Loader files	AIX, HP-UX, Linux, and Solaris: 32-bit with 32-bit files	2,147,483,647 bytes
	AIX, HP-UX, Linux, and Solaris: 64-bit files	Unlimited **Note**: File size is limited by limits allowed by the operating system on different file systems.
Control files	HP-UX, Linux, AIX, and Solaris	25000 control file blocks with a block size of 4096 bytes.

G

Very Large Memory and HugePages

This chapter guides Linux system administrators to configure Very Large Memory configurations and HugePages on Red Hat Enterprise Linux/ Oracle Linux.

This chapter contains the following sections:

- Very Large Memory on Linux x86
- Overview of HugePages

Very Large Memory on Linux x86

Very Large Memory (VLM) configurations allow a 32-bit Oracle Database to access more than 4GB RAM that is traditionally available to Linux applications. The Oracle VLM option for 32-bit creates a large database buffer cache using an in-memory file system (`/dev/shm`). Other parts of the SGA are allocated from regular memory. VLM configurations improve database performance by caching more database buffers in memory, which significantly reduces the disk I/O compared to configurations without VLM. This chapter shows how to increase the SGA memory using VLM on a 32-bit computer.

> **Note:** The contents documented in this section apply only to 32-bit Linux operating system. With a 64-bit architecture, VLM support is available natively. All 64-bit Linux operating systems use the physical memory directly, as the maximum available virtual address space is 16 EB (exabyte = 2^60 bytes.)

This section includes the following topics:

- Implementing VLM on 32-bit Linux
- Prerequisites for Implementing VLM
- Methods To Increase SGA Limits
- Configuring Very Large Memory for Oracle Database
- Restrictions Involved in Implementing Very Large Memory

Implementing VLM on 32-bit Linux

With 32-bit architectures, VLM is accessed through a VLM window of a specific size. The VLM window is a data structure in the process address space that provides access to the whole virtual address space from a window of a specific size. On 32-bit Linux, you must set the parameter `USE_INDIRECT_DATA_BUFFERS=TRUE`, and mount a

`shmfs` or `tmpfs` or `ramfs` type of in-memory filesystem over `/dev/shm` to increase the usable address space.

Prerequisites for Implementing VLM

The following are some of the prerequisites for implementing VLM on a 32-bit operating system:

- The computer on which Oracle Database is installed must have more than 4GB of memory.
- The computer must be configured to use a kernel with PAE support upon startup.
- The `USE_INDIRECT_DATA_BUFFERS=TRUE` must be present in the initialization parameter file for the database instance that uses VLM support.
- Initialization parameters `DB_BLOCK_BUFFERS` and `DB_BLOCK_SIZE` must be set to values you have chosen for the Oracle Database.

Methods To Increase SGA Limits

In a typical 32-bit Linux kernel, one can create an SGA of up to 1.7GB size. Using a Linux Hugemem kernel enables the creation of an SGA of upto 3.6GB size. To go beyond 3.6GB on a 32-bit kernel, you must use the VLM feature.

The following are the methods to increase SGA limits on a 32-bit computer:

- Hugemem Kernel
- Hugemem Kernel with Very Large Memory

Hugemem Kernel

Red Hat Enterprise Linux 4 and Oracle Linux 4 include a new kernel known as the Hugemem kernel. The Hugemem kernel feature is also called a 4GB-4GB Split Kernel as it supports a 4GB per process user space (versus 3GB for the other kernels), and a 4GB direct kernel space. Using this kernel enables RHEL 4/Oracle Linux 4 to run on systems with up to 64GB of main memory. The Hugemem kernel is required to use all the memory in system configurations containing more than 16GB of memory. The Hugemem kernel can run configurations with less memory.

A classic 32-bit 4GB virtual address space is split 3GB for user processes and 1GB for the kernel. The new scheme (4GB/4GB) permits 4GB of virtual address space for the kernel and almost 4GB for each user process. Due to this scheme with hugemem kernel, 3.6GB of SGA can be created without using the indirect data buffer method.

> **Note:** Red Hat Enterprise Linux 5/ Oracle Linux 5 on 32-bit does not have the hugemem kernel. It supports only the 3GB user process/ 1GB kernel split. It has a PAE kernel that supports systems with more than 4GB of RAM and reliably upto 16GB. Since this has a 3GB/1GB kernel split, the system may run out of lowmem if the system's load consumes lots of lowmem. There is no equivalent kernel for hugemem in Enterprise Linux 5 and one is recommended to either use Enterprise Linux 4 with hugemem or go for 64-bit.

The Hugemem kernel on large computers ensures better stability as compared to the performance overhead of address space switching.

Run the following command to determine if you are using the `Hugemem` kernel:

```
$ uname -r
2.6.9-5.0.3.ELhugemem
```

Hugemem Kernel with Very Large Memory

If you use only Hugemem kernels on 32-bit systems, then the SGA size can be increased but not significantly. Refer to section "Hugemem Kernel" on page 2, for more information.

> **Note:** Red Hat Enterprise Linux 5/ Oracle Linux 5 does not support the hugemem kernel. It supports a PAE kernel that can be used to implement Very Large Memory (VLM) as long as the physical memory does not exceed 16GB.

This section shows how the SGA can be significantly increased by using Hugemem kernel with VLM on 32-bit systems.

The SGA can be increased to about 62GB (depending on block size) on a 32-bit system with 64GB RAM. A processor feature called Page Address Extension (PAE) permits you to physically address 64GB of RAM. Since PAE does not enable a process or program to either address more than 4GB directly, or have a virtual address space larger than 4GB, a process cannot attach to shared memory directly. To address this issue, a shared memory filesystem (memory-based filesystem) must be created which can be as large as the maximum allowable virtual memory supported by the kernel. With a shared memory filesystem processes can dynamically attach to regions of the filesystem allowing applications like Oracle to have virtually a much larger shared memory on 32-bit operating systems. This is not an issue on 64-bit operating systems.

VLM moves the database buffer cache part of the SGA from the System V shared memory to the shared memory filesystem. It is still considered one large SGA but it consists now of two different operating system shared memory entities. VLM uses 512MB of the non-buffer cache SGA to manage VLM. This memory area is needed for mapping the indirect data buffers (shared memory filesystem buffers) into the process address space since a process cannot attach to more than 4GB directly on a 32-bit system.

> **Note:** `USE_INDIRECT_DATA_BUFFERS=TRUE` must be present in the initialization parameter file for the database instance that use Very Large Memory support. If this parameter is not set, then Oracle Database 11*g* Release 2 (11.2) or later behaves in the same way as previous releases.
>
> You must also manually set the initialization parameters `DB_BLOCK_BUFFERS` and `SHARED_POOL_SIZE` to values you have chosen for an Oracle Database. Automatic Memory Management (AMM) cannot be used. The initialization parameter `DB_BLOCK_SIZE` sets the block size and in combination with `DB_BLOCK_BUFFERS` determines the buffer cache size for an instance

For example, if the non-buffer cache SGA is 2.5GB, then you will only have 2GB of non-buffer cache SGA for shared pool, large pool, and redo log buffer since 512MB is used for managing VLM. It is not recommended to use VLM if buffer cache size is less than 512MB.

In RHEL 4/ Oracle Linux 4 there are two different memory file systems that can be used for VLM:

- `tmpfs or shmfs`: mount a `shmfs` with a certain size to `/dev/shm`, and set the correct permissions. For `tmpfs` you do not need to specify a size. `Tmpfs or shmfs` allocated memory is pageable.

  For example:

  ```
  Example Mount shmfs:
  # mount -t shm shmfs -o size=20g /dev/shm

  Edit /etc/fstab:
  shmfs /dev/shm shm size=20g 0 0

  OR

  Example Mount tmpfs:
  # mount –t tmpfs tmpfs /dev/shm

  Edit /etc/fstab:
  none /dev/shm tmpfs defaults 0 0
  ```

- `ramfs`: `ramfs` is similar to `shmfs`, except that pages are not pageable or swappable. This approach provides the commonly desired effect. `ramfs` is created by:

  ```
  umount /dev/shm
  mount -t ramfs ramfs /dev/shm
  ```

Configuring Very Large Memory for Oracle Database

Complete the following procedure to configure Very Large Memory on Red Hat Enterprise Linux 4/ Oracle Linux 4 using `ramfs`:

1. Log in as a `root` user:

   ```
   sudo -sh
   Password:
   ```

2. Edit the `/etc/rc.local` file and add the following entries to it to configure the computer to mount `ramfs` over the `/dev/shm` directory, whenever you start the computer:

   ```
   umount /dev/shm
   mount -t ramfs ramfs /dev/shm
   chown oracle:oinstall /dev/shm
   ```

   In the preceding commands, `oracle` is the owner of Oracle software files and `oinstall` is the group for Oracle owner account. If the new configuration disables `/etc/rc.local` file or you start an instance of Oracle database using a Linux service script present under the `/etc/init.d` file, then you can add those entries in the service script too.

   Note, this configuration will make `ramfs` ready even before your system autostarts crucial Oracle Database instances. The commands can also be included in your startup scripts. It is important that you test the commands extensively by repeated reboot action, after you complete configuring the computer using the following steps:

3. Restart the server.

4. Log in as a `root` user.
5. Run the following command to check if the `/dev/shm` directory is mounted with the ramfs type:

```
/dev/shm directory is mounted with the ramfs type:

# mount | grep shm
ramfs on /dev/shm type ramfs (rw)
```

6. Run the following command to check the permissions on the `/dev/shm` directory:

```
# ls -ld /dev/shm
drwxr-xr-x  3 oracle oinstall 0 Jan 13 12:12 /dev/shm
```

7. Edit the `/etc/security/limits.conf` file and add the following entries to it to increase the max locked memory limit:

```
soft    memlock        3145728
hard    memlock        3145728
```

8. Switch to the `oracle` user:

```
# sudo - oracle
Password:
```

9. Run the following command to check the max locked memory limit:

```
$ ulimit -l
3145728
```

10. Complete the following procedure to configure instance parameters for Very Large Memory:
    a. Replace the `DB_CACHE_SIZE`, `DB_xK_CACHE_SIZE`, `sga_target`, and `memory_target` parameters with `DB_BLOCK_BUFFERS` parameter.
    b. Add the `USE_INDIRECT_DATA_BUFFERS=TRUE` parameter.
    c. Configure SGA size according to the SGA requirements.
    d. Remove `SGA_TARGET`, `MEMORY_TARGET`, or `MEMORY_MAX_TARGET` parameters, if set.
11. Start the database instance.
12. Run the following commands to check the memory allocation:

```
$ ls -l /dev/shm
$ ipcs -m
```

See Also: "Configuring HugePages on RHEL/ Oracle Linux" on page 6 section for more information about HugePages.

Restrictions Involved in Implementing Very Large Memory

Following are the limitations of running a computer in the Very Large Memory mode:

- You cannot use Automatic Memory Management (AMM) while implementing VLM using `ramfs`, because AMM works on dynamic SGA tuning. With AMM swapping is possible. For example, you can unmap the unused SGA space and map it to PGA. Dynamic SGA and multiple block size are not supported with Very Large Memory because `ramfs` is not swappable. To enable Very Large Memory, you must ensure that you set the value of `MEMORY_TARGET` to zero.

- VLM can be implemented only if Database Buffer Cache size is greater than 512MB.

Overview of HugePages

HugePages is a feature integrated into the Linux kernel 2.6. It is a method to have larger page size that is useful for working with very large memory. HugePages is useful for both 32-bit and 64-bit configurations. HugePage sizes vary from 2MB to 256MB, 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.

This section includes the following topics:

- Tuning SGA With HugePages
- Configuring HugePages on RHEL/ Oracle Linux
- Restrictions for HugePages Configurations

Tuning SGA With HugePages

Without HugePages, the operating system keeps each 4KB of memory as a page, and when it is allocated to the SGA, then the lifecycle of that page (dirty, free, mapped to a process, and so on) is kept up to date by the operating system kernel.

With HugePages, the operating system page table (virtual memory to physical memory mapping) is smaller, since each page table entry is pointing to pages from 2MB to 256MB. Also, the kernel has fewer pages whose lifecyle must be monitored.

Note: 2MB size of HugePages is available with Linux x86-64 and Linux x86. 256 MB size of HugePages is available only with Linux Itanium.

For example, if you use HugePages with Itanium hardware, and you want to map 256MB 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 MB * 1024KB/4KB = 65536 PTEs.

The following are the advantages of using HugePages:

- Increased performance through increased TLB hits.
- Pages are locked in memory and are never swapped out which guarantees that shared memory like SGA remains in RAM.
- Contiguous pages are preallocated and cannot be used for anything else but for System V shared memory (e.g. SGA)
- Less bookkeeping work for the kernel for that part of virtual memory due to larger page sizes

Configuring HugePages on RHEL/ Oracle Linux

Complete the following steps to configure HugePages on the computer:

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

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

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

   Note: Following is an example that may require modifications.

   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') 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
```

3. 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.

4. Set the following kernel parameter:

```
# sysctl -w vm.nr_hugepages=value_displayed_in_step_3
```

5. To make the value of the parameter available for every time you restart the computer, edit the `/etc/sysctl.conf` file and add the following entry:

   ```
   vm.nr_hugepages=value_displayed_in_step_3
   ```

6. Run the following command to check the available `hugepages`:

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

7. Restart the instance.
8. Run the following command to check the available `hugepages` (1 or 2 pages free):

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

> **Note:** If the setting of the `nr_hugepages` parameter is not effective, you must restart the server.

Restrictions for HugePages Configurations

Following are the limitations of using HugePages:

- The Automatic Memory Management (AMM) and HugePages are not compatible. With AMM the entire SGA memory is allocated by creating files under `/dev/shm`. When Oracle Database allocates SGA that way HugePages are not reserved. You must disable AMM on Oracle Database 11*g* to use HugePages.
- If you are using VLM in a 32-bit environment, then you cannot use HugePages for the Database Buffer cache. HugePages can be used for other parts of SGA like `shared_pool`, `large_pool`, and so on. Memory allocation for VLM (buffer cache) is done using shared memory file systems (`ramfs/tmpfs/shmfs`). HugePages does not get reserved or used by the memory file systems.
- HugePages are not subject to allocation or release after system startup, unless a system administrator changes the HugePages configuration by modifying the number of pages available, or the pool size. If the space required is not reserved in memory during system startup, then HugePages allocation fails.

Index

Symbols

@ abbreviation, 1-1

A

A_TERM environment variable, 6-10
A_TERMCAP environment variable, 6-10
ADA_PATH environment variable, 1-4
adapters utility, 5-2
administering command line SQL, 4-1
administrators
  operating system accounts, 1-8
AIX tools
  Base Operation System tools, 8-4
  Performance Tool Box Agent, 8-5
  Performance Tool Box Manager, 8-5
  SMIT, 8-5
  System Management Interface tools, 8-5
AIXTHREAD_SCOPE environment variable, A-14
archive buffers
  tuning, A-4
ASM_DISKSTRING initialization parameter, 1-7
assistants
  Oracle Database Configuration Assistant, 3-2
  Oracle Database Upgrade Assistant, 3-2
  Oracle Net Configuration Assistant, 3-1
asynchronous flag in SGA, B-7
asynchronous input/output, B-3, B-4
  verifying, B-6
Automatic Storage Management
  restarting, 2-3
  stopping, 2-1
Automatic Storage Management, using, 8-9
automating
  shutdown, 2-5
  startup, 2-5

B

bit-length support, 6-4
block size, A-4
  adjusting, 8-8
buffer cache
  limitations, C-2
  paging activity, A-1
  support, C-1
  tuning, A-2
  tuning size, 8-14
buffer manager, 8-7
BUFFER parameter, A-5

C

cache size, 8-14
catching routine, 6-19
  example, 6-19
CLASSPATH environment variable, 1-4
client shared libraries, 6-3
client static libraries, 6-3
COBDIR environment variable, 6-9
commands
  iostat, 8-3
  lsps, 8-3
  sar, 8-2
  SPOOL, 4-4
  swap, 8-3
  swapinfo, 8-3
  swapon, 8-3
  vmo, A-2
  vmstat, 8-2
common environment
  setting, 1-5
concurrent input/output, A-7
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
configuring
  accounts of Oracle users, 1-8
  Oracle Database, 3-2
coraenv file, 1-5
CPU scheduling, A-13
CPU_COUNT initialization parameter, B-9
CREATE CONTROLFILE parameter, F-1
CREATE DATABASE parameter, F-1

D

database
  block size, A-4
database block size
  setting, A-4
Database Control
  See Oracle Enterprise Manager Database Control
database limits, F-1
DB_BLOCK_SIZE initialization parameter, 1-7, 8-12
DB_CACHE_SIZE initialization parameter, 8-12
DB_FILE_MULTIBLOCK_READ_COUNT
    parameter, A-11
DBAs
  See administrators
dbhome file, 1-6
debugger programs, 6-3
demo_proc32.mk file, 6-7
demo_procob.mk file, 6-11
demonstration programs
  for Pro*COBOL, 6-11
  Oracle JDBC/Oracle Call Interface, 6-17
  Pro*C/C++, 6-6
  Pro*FORTRAN, 6-13
  SQL*Module for Ada, 6-14
demonstrations
  PL/SQL, 7-1
  precompiler, 7-4
  SQL*Loader, 7-1
direct input/output, A-7
disk input/output
  file system type, 8-10
  input/output
    slaves, A-10
  pacing, A-12
  tuning, 8-9
disks
  monitoring performance, 8-10
DISM, D-1
DISPLAY environment variable, 1-4
dynamic cache parameters, C-2
dynamic linking
  Oracle libraries and precompilers, 6-3

E

environment variables, 6-9
  A_TERM, 6-10
  A_TERMCAP, 6-10
  ADA_PATH, 1-4
  AIXTHREAD_SCOPE, A-14
  all, 1-2
  CLASSPATH, 1-4
  COBDIR, 6-9
  DISPLAY, 1-4
  for Pro*COBOL, 6-8
  HOME, 1-5
  LANG, 1-5
  LANGUAGE, 1-5
  LD_LIBRARY_PATH, 1-5, 6-9, 6-10
  LD_OPTIONS, 1-5
  LIBPATH, 6-9
  LPDEST, 1-5
  MicroFocus COBOL compiler, 6-9
  ORA_TZFILE, 1-2
  ORACLE_BASE, 1-2
  ORACLE_HOME, 1-2
  ORACLE_PATH, 1-3
  ORACLE_SID, 1-1, 1-3
  ORACLE_TRACE, 1-3
  ORAENV_ASK, 1-3
  PATH, 1-5, 4-3, 6-9, 6-10
  PRINTER, 1-5
  SHLIB_PATH, 6-9
  SQLPATH, 1-3
  TMPDIR, 1-5
  TNS_ADMIN, 5-1
  TWO_TASK, 1-3
executables
  precompiler, 6-2
  precompilers, 6-2
  relinking, 3-3
Extended file system, 8-10

F

file buffer cache
  tuning on AIX, A-2
file systems
  ext2/ext3, 8-10
  GPFS, 8-10
  JFS, 8-10
  OCFS2, 8-10
  S5, 8-10
  UFS, 8-10
  VxFS, 8-10
files
  alert, 1-8
  coraenv, 1-5
  dbhome, 1-6
  demo_procob.mk, 6-11
  glogin.sql, 4-1
  ins_precomp.mk, 6-2
  login.sql, 4-1
  Oracle Net Services configuration, 5-1
  oraenv, 1-5
  ottcfg.cfg, 6-2
  pcbcfg.cfg, 6-2
  pccfor.cfg, 6-2
  pcscfg.cfg, 6-2
  pmscfg.cfg, 6-2
  privgroup, B-2, B-4
  README, 6-3
  root.sh, 1-6
  services, 5-4
  trace, 1-8
FORMAT precompiler, 6-11
  Pro*COBOL, 6-12

G

getprivgrp command, B-2, B-4

glogin.sql file, 4-1
GPFS
  considerations for using, A-8

H

HOME environment variable, 1-5
HP-UX dynamic processor reconfiguration, B-9
HP-UX tools, 8-6
  Glance/UX, 8-5
hugetlbfs
  on SUSE, C-3

I

implementing asynchronous input/output, B-4
Import utility, A-5
initialization parameters, 1-6
  ASM_DISKSTRING, 1-7
  CPU_COUNT, B-9
  DB_BLOCK_SIZE, 1-7, 8-12
  DB_CACHE_SIZE, 8-12
  JAVA_POOL_SIZE, 8-12
  LARGE_POOL_SIZE, 8-12
  LOG_ARCHIVE_DEST_n, 1-7
  LOG_BUFFERS, 8-12
  SHARED_POOL_SIZE, 8-12
in-memory file system, C-1
input/output
  asynchronous, A-9, B-3
  slaves, A-10
  tuning, 8-9
ins_precomp.mk file, 6-2
installing
  SQL*Plus command line Help, 4-2
intimate shared memory, D-1
I/O buffers and SQL*Loader, A-5
iostat command, 8-3
IPC protocol, 5-3
ireclen, 6-3
ISM, D-1

J

JAVA_POOL_SIZE initialization parameters, 8-12
JFS2 considerations, A-7
JFSconsiderations, A-7
Journaled file system, 8-10, A-6, A-8

L

LANG environment variable, 1-5
LANGUAGE environment variable, 1-5
LARGE_POOL_SIZE initialization parameters, 8-12
LD_LIBRARY_PATH environment variable, 1-5, 6-9, 6-10
LD_OPTIONS environment variable, 1-5
LIBPATH environment variable, 6-9
libraries
  client shared and static, 6-3
lightweight timer implementation, B-3
Linux tools, 8-6
listener
  setting up for TCP/IP or TCP/IP with Secure Sockets Layer, 5-4
LOG_ARCHIVE_DEST_n initialization parameter, 1-7
LOG_BUFFERS initialization parameters, 8-12
Logical Volume Manager, A-5
login.sql file, 4-1
LPDEST environment variable, 1-5
lsps command, 8-3

M

make files
  custom, 6-17
  demo_proc32.mk, 6-7
  demo_procob.mk, 6-11
  ins_precomp.mk, 6-2
MAXDATAFILES parameter, F-1
maxfree parameter, A-2
MAXINSTANCES parameter, F-1
MAXLOGFILES parameter, F-1
MAXLOGHISTORY parameter, F-1
MAXLOGMEMBERS parameter, F-1
maxperm parameter, A-2
memory
  contention, A-1
  control paging, 8-8
  swap space, 8-7
  tuning, 8-7
MicroFocus COBOL compiler, 6-9
migrating, 3-2
minfree parameter, A-2
minor numbers of 8-bit, B-5
minperm parameter, A-2
MLOCK privilege, B-4
mpstat command, 8-6
multiple signal handlers, 6-19
multithreaded applications, 6-19

O

operating system buffer cache, tuning, 8-14
operating system commands
  getprivgrp, B-2
  running, 4-4
  setprivgrp, B-2
operating system groups
  OSDBA, 1-8
  OSOPER, 1-8
operating system tools
  for AIX, 8-4
  iostat, 8-3
  lsps, 8-3
  sar, 8-2
  swap, 8-3
  swapinfo, 8-3
  swapon, 8-3
  vmstat, 8-2
ORA_NLS10 environment variable, 1-2

ORA_TZFILE environment variable, 1-2
Oracle block size, adjusting, 8-8
Oracle buffer manager, 8-7
Oracle C++ Call Interface, 6-16
  demonstration programs, 6-16
  user programs, 6-16
Oracle Call Interface, 6-16
  demonstration programs, 6-16
  user programs, 6-16
Oracle Cluster Services Synchronization Daemon
  starting, 2-3
  stopping, 2-3
Oracle Database, 3-2
  restarting, 2-3
Oracle Database Configuration Assistant
  configuring, 3-2
Oracle Database environment variables
  Oracle Database variables, 1-2
Oracle Database process
  stopping, 2-1
Oracle Database Sample Schemas
Oracle Database tuning and large memory
    allocations, B-8
Oracle Database Upgrade Assistant, 3-2
Oracle Enterprise Manager Database Control
  starting, 2-4
  stopping, 2-3
Oracle environment variables
  ORA_NLS10, 1-2
  ORACLE_BASE, 1-2
  ORACLE_HOME, 1-2
  ORACLE_SID, 1-3
Oracle JDBC/Oracle Call Interface
  demonstration programs, 6-17
Oracle Management Agent
  starting, 2-5
  stopping, 2-4
Oracle Net Configuration Assistant
  using, 3-1
Oracle Net Services
  configuration files, 5-1
  IPC protocol, 5-3
  Oracle advanced security, 5-5
  protocol support, 5-3
  protocols, 5-3
  Secure Sockets Layer protocol, 5-4
  TCP/IP protocol, 5-3
Oracle ODBC Driver, E-1
Oracle Protocol Support
  IPC protocol, 5-3
  TCP/IP protocol, 5-3
  TCP/IP with Secure Sockets Layer protocol, 5-4
Oracle user accounts
  configuring, 1-8
ORACLE_BASE environment variable, 1-2
ORACLE_HOME environment variable, 1-2
ORACLE_PATH environment variable, 1-3
ORACLE_SID environment variable, 1-1, 1-3
ORACLE_TRACE environment variable, 1-3
oradism command, D-1
oraenv file, 1-5
ORAENV_ASK environment variable, 1-3
oreclen, 6-3
OSDBA group, 1-8
OSOPER group, 1-8
ottcfg.cfg file, 6-2

P

page-out activity, 8-8
paging
  allocating sufficient, A-3
  controlling, A-3
  tuning, 8-7
parameters
  BUFFER, A-5
  CREATE CONTROLFILE, F-1
  CREATE DATABASE, F-1
  DB_FILE_MULTIBLOCK_READ_COUNT, A-11
  dynamic cache, C-2
  MAXDATAFILES, F-1
  maxfree, A-2
  MAXLOGFILES, F-1
  MAXLOGHISTORY, F-1
  MAXLOGMEMBERS, F-1
  maxperm, A-2
  minfree, A-2
  minperm, A-2
  SCHED_NOAGE, B-2
  SGA_MAX_SIZE, D-1
  shm_max, 8-11
  shm_seg, 8-12
  shmmax, 8-11
  shmseg, 8-12
  USE_INDIRECT_DATA_BUFFERS, C-2
PATH environment variable, 1-5, 4-3, 6-9, 6-10
pcbcfg.cfg file, 6-2
pccfor.cfg file, 6-2
pcscfg.cfg file, 6-2
Performance Tool Box Agent, 8-5
performance tuning tools, 8-6
PL/SQL demonstrations, 7-1
PL/SQL kernel demonstrations, 7-2
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, 6-8
  README files, 6-3
  running demonstrations, 7-4
  signals, 6-19
  uppercase to lowercase conversion, 6-3
  value of ireclen and oreclen, 6-3
  vendor debugger programs, 6-3
PRINTER environment variable, 1-5

privgroup file, B-2, B-4
Pro*C/C++
  demonstration programs, 6-6
  make files, 6-6
  signals, 6-19
  user programs, 6-7
Pro*C/C++ precompiler, 6-6
Pro*COBOL
  demonstration programs, 6-11
  environment variables, 6-8
  FORMAT precompiler, 6-11, 6-12
  naming differences, 6-8
  Oracle Runtime system, 6-10
  user programs, 6-11
Pro*FORTRAN demonstration programs, 6-13
processor binding on SMP systems
  using, A-14
PRODUCT_USER_PROFILE table, 4-2
Programmer's Analysis Kit, 8-6
protocols, 5-3

R

raw devices
  buffer cache size, 8-14
raw logical volumes, A-6, A-8
relinking executables, 3-3
removing
  SQL*Plus command line Help, 4-3
resilvering, with Oracle Database, A-13
restarting
  Automatic Storage Management, 2-3
  Oracle Database, 2-3
restrictions, SQL*Plus, 4-4
  passwords, 4-5
  resizing windows, 4-5
  return codes, 4-5
root.sh script, 1-6

S

Sample Schemas
  See Oracle Database Sample Schemas
sar command, 8-2, 8-8
SCHED_NOAGE
  enabling, B-2
  scheduling policy, B-2
scripts
  root.sh, 1-6
sequential read ahead
  tuning, A-12
services file, 5-4
setprivgrp command, B-2, B-4
SGA, 8-11
  determining the size of, 8-12
SGA_MAX_SIZE parameter, D-1
shared memory segments, B-1
shared memory, on AIX, 8-13
SHARED_POOL_SIZE initialization
    parameters, 8-12
SHLIB_PATH environment variable, 6-9
shm_max parameter, 8-11
shm_seg parameter, 8-12
shmmax parameter, 8-11
shmseg parameter, 8-12
shutdown
  automating, 2-5
SIGCLD signal, 6-19
SIGCONT signal, 6-19
SIGINT signal, 6-19
SIGIO signal, 6-19
signal handlers, 6-19
signal routine, 6-19
  example, 6-19
signals
  SIGCLD, 6-19
  SIGCONT, 6-19
  SIGINT, 6-19
  SIGIO, 6-19
  SIGPIPE, 6-19
  SIGTERM, 6-19
  SIGURG, 6-19
SIGPIPE signal, 6-19
SIGTERM signal, 6-19
SIGURG signal, 6-19
Solaris tools, 8-6
SPOOL command
  SQL*Plus, 4-4
SQL*Loader, A-5
SQL*Loader demonstrations, 7-1
SQL*Module for Ada, 6-14
  demonstration programs, 6-14
  user programs, 6-15
SQL*Plus
  command line Help, 4-2
  default editor, 4-3
  editor, 4-3
  interrupting, 4-4
  PRODUCT_USER_PROFILE table, 4-2
  restrictions, 4-4
  running operating system commands, 4-4
  site profile, 4-1
  SPOOL command, 4-4
  system editor, 4-3
  user profile, 4-1
  using command-line SQL*Plus, 4-3
SQL*Plus command line Help
  installing, 4-2
  removing, 4-3
SQL*Plus, interrupting, 4-4
SQLPATH environment variable, 1-3
starting
  Oracle Cluster Services Synchronization
      Daemon, 2-3
  Oracle Enterprise Manager Database Control, 2-4
  Oracle Management Agent, 2-5
startup
  automating, 2-5
static linking
  Oracle libraries and precompilers, 6-3
stopping

Oracle Cluster Services Synchronization Daemon, 2-3
Oracle Enterprise Manager Database Control, 2-3
Oracle Management Agent, 2-4
striped logical volume
designing, A-6
swap command, 8-3
swap space, 8-7
tuning, 8-7
swap space allocation, 8-7
swapinfo command, 8-3
swapon command, 8-3
symfind utility, 6-18
SYSDATE, 1-6
system editor
SQL*Plus, 4-3
system time, 1-6

T

tables
PRODUCT_USER_PROFILE, 4-2
TCP/IP protocol, 5-3
TCP/IP with Secure Sockets Layer protocol, 5-4
thread support, 6-19
TMPDIR environment variable, 1-5
trace alert, 1-8
trace files, 1-8
tuning, 8-7
disk input/output, 8-9
input/output bottlenecks, 8-9
large memory allocations, B-8
memory management, 8-7
recommendations, B-9
sequential read ahead, A-12
tuning tools
Glance/UX utility, 8-5
iostat command, 8-3
lsps command, 8-3
mpstat, 8-6
Performance Tool Box Agent, 8-5
Performance Tool Box Manager, 8-5
Programmer's Analysis Kit, 8-6
sar command, 8-2
swap command, 8-3
swapinfo command, 8-3
swapon command, 8-3
vmstat command, 8-2
TWO_TASK environment variable, 1-3

U

undefined symbols, 6-18
unified file systems, 8-10
UNIX System V file system, 8-10
upgraded databases
configuring, 3-2
upgrading, 3-2
USE_INDIRECT_DATA_BUFFERS parameter, C-2
user interrupt handler, 6-19
user profile
SQL*Plus, 4-1
user programs
for Pro*C/C++, 6-7
Oracle C++ Call Interface, 6-16
Oracle Call Interface, 6-16
Pro*C/C++, 6-7
Pro*COBOL, 6-11
SQL*Module for Ada, 6-15
using command-line SQL*Plus, 4-3
utilities
adapters, 5-2
Import, A-5
symfind, 6-18
UTLRP.SQL
recompiling invalid SQL modules, 3-2

V

Veritas file system, 8-10
virtual memory data pages
tuning Oracle Database, B-8
Virtual Memory Manager, A-12
virtual memory page size, default, B-8
vmo command, A-2
vmstat command, 8-2

X

XA functionality, 6-20
X/Open Distributed Transaction Processing XA interface, 6-20