Oracle8™ Administrator's Reference for LINUX

Release 8.0.5 Standard Edition

September 1998

Part No. A66585-01

Topics Including:
Optimal Flexible Architecture on Oracle8
Administering Oracle8 on LINUX
Tuning Oracle8 on LINUX
Administering SQL*Plus on LINUX
Using Oracle Precompilers and the Oracle Call Interface on LINUX
Configuring Oracle Net8
Running Oracle Data Cartridge Demos

ORACLE®

Enabling the Information Age™

Oracle8 Administrator's Reference for LINUX

Release 8.0.5

Part No. A66585-01

Primary Author: Kevin Kerr

Contributors: Kevin Adams, with Nicholas Hind, Nik Ormseth, and Lynn Robinson.

Contents

3 Tuning Oracle8 on LINUX

4 Administering SQL*Plus on Linux

5 Using Oracle Precompilers and the Oracle Call Interface on Linux

6 Configuring Oracle Net8

Index

Send Us Your Comments

Oracle8 Administrator's Reference, Release 8.0.5 for LINUX

Part No. A66585-01

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

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

If you find any errors or have other suggestions for improvement, please indicate the book title, part number, chapter, and section. You can send comments to us in the following ways:

Technical Publications Manager
Platform Technologies Division
Oracle Corporation
500 Oracle Parkway, Mailstop 1op2
Redwood Shores, CA 94065
USA

If you would like a reply, please give your name, postal or email address, and telephone number.

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

Preface

Purpose

This reference provides Linux-specific information required to successfully administer and tune the Oracle8 Server. This reference supplements information found in the Oracle8 and Oracle8 Enterprise Edition Documentation Set.

Audience

This document is intended for anyone responsible for administering the Oracle8 Server on a LINUX 2.0.34 system.

Oracle8 and Oracle8 Enterprise Edition

Unless noted otherwise, features and functionality described in this document are common to both Oracle8 and Oracle8 Enterprise Edition. For more information about Oracle8 Enterprise Edition-specific features and functionality, refer to the What's New section of the Release Notes.

Typographic Conventions

`monospace`	Monospace type indicates LINUX commands, directory names, path names, and filenames.
brackets [ ]	Words enclosed in brackets indicate key names (for example, Press [Return]). Note that brackets have a different meaning when used in command syntax.
italics	Italic type indicates a variable, including variable portions of filenames, or emphasis.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) commands, initialization parameters, or environment variables.

Because LINUX is case-sensitive, conventions in this document may differ from those used in Oracle product documentation.

Command Syntax

Command syntax appears in `monospace` font. The following conventions apply to command syntax:

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

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

Contacting Customer Support

Please copy this page and distribute it within your organization as necessary.

Oracle Support Services (OSS) can be reached at the following numbers (the hours are specified in your support contract):

- In the United States, call: **1.650.506.1500**.
- In Europe, call: **+44.1344.860160**.
- In Asia, call: **+81.3.5717.1860**.

Please prepare the following information before you call:

- Your CSI number (if applicable) or complete contact details, including any special project information.
- The release levels of the Oracle Server and associated products (for example, Oracle8 Server release 8.0.5.0, and Oracle Forms release 4.5.6.3.2).
- Operating system name and release level, including patches and packages.
- Details of error codes, numbers, and descriptions associated with the problem.
- A full description of the issue, including:
  - What happened? For example, the command used and result obtained.
  - When did it happen? For example, time of day, or after a particular command, or after an operating system or Oracle upgrade.
  - Where did it happen? For example, on a particular system, or within a particular procedure or table.
  - What is the extent of the problem? For example, is your production system unavailable, or is the impact less severe? Is the problem getting worse?

  Keep in mind what did *not* happen, as well as what did happen.
- Copies of any trace files, core dumps, or log files recorded near the time of the incident.

For installation-related problems, please have the following information available:

- Listings of the contents of the `ORACLE_HOME` directory, and any staging area, if applicable.
- Contents of the installation log files in the `$ORACLE_HOME/orainst` directory: `install.log`, `sql.log`, `make.log`, and `os.log`.

For more information, contact http://www.oracle.com/support.

Related Documentation

Advanced configuration and tuning recommended for a production database system is provided in the following manuals:

- *Oracle8 Administrator's Guide*. Use this as a starting point for tasks associated with the Oracle8 Server, such as database creation, managing database objects, and creating users.
- *Net8 Administrator's Guide*
- *Oracle8 Tuning*

Unfamiliar with the concepts or terminology associated with relational database management systems? Read Chapter 1 in the *Oracle8 Concepts* before beginning your installation.

Ordering Documentation

- In the United States, call Documentation Sales at: **1.800.252.0303**.
- In the United Kingdom, call Oracle Direct Response at: **+44.990.332200**.
- In other European countries, contact your local Oracle Support office.
- In the Asia-Pacific region, contact your Oracle sales representative.

Shipping Inquiries

- In the United States, call Client Relations at: **1.650.506.1500.**
- In the United Kingdom, call Customer Relations at: **+44.990.622300.**
- In other European countries, contact your local Oracle Support office.
- In the Asia-Pacific region, contact your Oracle sales representative.

1

Optimal Flexible Architecture on Oracle8

- Optimal Flexible Architecture (OFA)
- OFA Implemented on Oracle8 for LINUX

Optimal Flexible Architecture (OFA)

Oracle Corporation recommends the Optimal Flexible Architecture (OFA) standard for Oracle8. The OFA standard is a set of configuration guidelines for fast, reliable Oracle databases that require little maintenance.

OFA is designed to:

- organize large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- facilitate routine administrative tasks like software and data backup functions, which are often vulnerable to data corruption
- alleviate switching among multiple Oracle databases
- adequately manage and administer database growth
- help eliminate fragmentation of free space in the data dictionary, isolate other fragmentation, and minimize resource contention

Characteristics of OFA-Compliant Database

An OFA-compliant database provides the following benefits:

File System Organization

The file system is organized to allow easy administration and accommodate scalability for:

- adding data into existing databases
- adding users
- creating databases
- adding hardware

Distributed I/O Loads

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

Hardware Support

Hardware costs are minimized only when it does not conflict with operational considerations.

Safeguards Against Drive Failures

By spreading applications across more than one drive, drive failures impact as few applications as possible.

Distribution of Home Directories

The following items can be distributed across more than one disk drive:

- the collection of home directories
- the contents of an individual home directory

Integrity of Login Home Directories

It is possible to add, move, or delete login home directories without having to revise programs that refer to them.

Independence of LINUX Directory Subtrees

Categories of files are separated into independent LINUX directory subtrees so that files in one category are minimally affected by operations on files in other categories.

Supports Concurrent Execution of Application Software

It is possible to execute multiple versions of applications software simultaneously, allowing the user to test and use a new release of an application before abandoning the previous version. Transferring to a new version after an upgrade is simple for the administrator and transparent for the user.

Distinguishes Administrative Information for each Database

The ability to separate administrative information about one database from that of another, ensures a reasonable structure for the organization and storage of administrative data.

Uses Consistent Database File Naming

Database files are named so that:

- database files are easily distinguishable from all other files
- files of one database are easily distinguishable from files of another database
- control files, redo log files, and data files are identifiable as such
- the association of data file to tablespace is clearly indicated

Separation of Tablespace Contents

Tablespace contents are separated to:

- minimize tablespace free space fragmentation
- minimize I/O request contention
- maximize administrative flexibility

Tuning of I/O Loads across all Drives

I/O loads are tuned across all drives, including drives storing Oracle data in raw devices.

Additional Benefits of OFA for Parallel Server

For Oracle Parallel Server Installations:

- administrative data is stored in a central place, accessible to all database administrators
- administrative data for an instance is associated with the instance by the file name

OFA Implemented on Oracle8 for LINUX

A careful naming strategy for database files eliminates data administration problems. The OFA rules provided here correspond to the original OFA recommendations published in *The OFA Standard: Oracle8 for Open Systems*.

Naming Mount Points

Mount Point Syntax

Name all mount points using the syntax */pm*, where *p* is a string constant and *m* is a unique fixed-length key (typically a two-digit number) used to distinguish each mount point. For example: `/u01` and `/u02`, or `/disk01` and `/disk02`.

Naming Mount Points for Very Large Databases (VLDBs)

If each disk drive contains database files from one application and there are enough drives for each database to ensure no I/O bottleneck, then use the syntax */q/dm* for naming mount points, as explained in Table 1–1.

Table 1–1 Syntax for Naming Mount Points

q	a string denoting that Oracle data is stored here

Table 1–1 Syntax for Naming Mount Points

dm	the value of the initialization parameter DB_NAME (synonymous with the instance *sid* for single-instance databases)

For example, mount points named `/u01/oradata/test01` and `/u01/oradata/test02` allocate two drives for the Oracle test database.

Naming Directories

Home Directory Syntax

Name home directories using the syntax */pm/h/u*, as explained in Table 1–2.

Table 1–2 Syntax for Naming Home Directories

pm	a mount point name
h	a standard directory name
u	the name of the owner of the directory

For example, `/u01/app/oracle` is the Oracle server software owner home directory (also referred to as `ORACLE_BASE` and defaulted by the installer) and `/u01/app/applmgr` is an Oracle applications software owner home directory.

Placing home directories at the same level in the LINUX file system is advantageous for the following reason: it allows the collection of applications owner login home directories on different mount points, to be referred to with the single pattern matching string, `/*/app/*`.

Referring to Pathnames

Refer to explicit pathnames only in files designed specifically to store them, such as `/etc/passwd` and the Oracle `oratab` file. Refer to group memberships only in the `/etc/group` file.

Software Directories

In order to help fulfill the OFA requirement that it be possible to simultaneously execute multiple versions of application software, store each version of the Oracle8 Server software in a directory matching the pattern */pm/h/*`product`*/v*, as explained in Table 1–3.

Table 1–3 Syntax for Naming Oracle8 Server Software Directories

h	a standard directory name

Table 1–3 Syntax for Naming Oracle8 Server Software Directories

v	the version of the software

For example: `/u01/app/oracle/product/8.0.5` indicates the start of the directory structure where the Oracle8 Server files are located. Set the `ORACLE_HOME` environment variable to this directory.

Naming Files

Administration Files

To facilitate the organization of administrative data, it is recommended that you store database-specific administration files in subdirectories according to *h*/`admin`/*d*/*a*/, where *h* is the Oracle software owner's home directory, *d* is the database name (DB_NAME), and *a* is a subdirectory for each of the following database administration files described in Table 1–4:

Table 1–4 Subdirectories for Database Administration Files

`adhoc`	ad hoc SQL scripts for a given database
`arch`	archived redo log files
`adump`	audit files (Set AUDIT_FILE_DEST in `config`*db_name*`.ora` to point here. This subdirectory should be cleaned out periodically).
`bdump`	background process trace files
`cdump`	core dump files
`create`	programs used to create the database
`exp`	database export files
`logbook`	files recording the status and history of the database
`pfile`	instance parameter files
`udump`	user SQL trace files

As an example, the subdirectory `adhoc` would have the following pathname, `/u01/app/oracle/admin/sab/adhoc/`

Database Files

The following naming convention for database files ensures that they are easily identifiable:

- for control files, use */pm/q/d/*`control.ctl`
- for redo log files, use */pm/q/d/*`redo`*n*`.log`
- for data files use, */pm/q/d/tn*`.dbf`

This syntax is explained in Table 1–5.

Table 1–5 Syntax for Naming Database Files

pm	a mount point name described earlier in this chapter
q	a string distinguishing Oracle data from all other files (usually named ORACLE or oradata)
d	the DB_NAME of the database
t	an Oracle tablespace name
n	a two-digit string

Note: Do not store files other than a control, redo log or data file associated with database *d* in the path */pm/q/d*.

Following this convention could produce for example, a data file with the name, `/u03/oradata/sab/system01.dbf`, making it easy to see to which database the file belongs.

Separate Segments with Different Requirements

Separate groups of segments with different lifespans, I/O request demands, and backup frequencies across different tablespaces.

For each Oracle database, create the special tablespaces described in Table 1–6. These tablespaces are in addition to those needed for application segments.

Table 1–6 Special Tablespace

SYSTEM	data dictionary segments
TEMP	temporary segments
RBS	rollback segments
TOOLS	general-purpose tools
USERS	miscellaneous user segments

This method is effective because dictionary segments are never dropped, and no other segments that can be dropped are allowed in the SYSTEM tablespace. This ensures that the SYSTEM tablespace does not require a rebuild due to tablespace free space fragmentation.

Because rollback segments are not stored in tablespaces holding applications data, the administrator is not blocked from taking an application's tablespace offline for maintenance. The segments are partitioned physically by type, and the administrator can record and predict data growth rates without complicated tools.

Naming Tablespaces

Name tablespaces descriptively using a maximum of eight characters.

Although Oracle8 tablespace names can be thirty characters long, portable LINUX file names are restricted to fourteen characters. The recommended standard for a data file basename is *tn*`.dbf`, where *t* is a descriptive tablespace name and *n* is a two-digit string. Because the extension plus the two-digit string occupy a total of six characters, only eight characters remain for the tablespace name.

Descriptive names allow the name of a data file to be associated with the tablespace that uses it. For example, the names `GLD` and `GLX` might be used for the tablespaces storing General Ledger data and indices, respectively.

> **Note:** Do not embed reminders of the word "tablespace" in your tablespace names. Tablespaces are distinguishable by context, and names do not need to convey information about type.

Exploiting OFA Structure for Oracle Files

Table 1–7 shows the syntax used for identifying classes of files.

Table 1–7 Directory Structure Syntax for Identifying Classes of Files

`/u[0-9][0-9]`	user data directories
`/*/home/*`	user home directories
`/*/app/*`	user application software directories
`/*/app/applmgr`	Oracle apps software subtrees
`/*/app/oracle/product`	Oracle Server software subtrees
`/*/app/oracle/product/8.0.5`	Oracle Server 8.0.5 distribution files
`/*/app/oracle/admin/sab`	`sab` database administrative subtrees
`/*/app/oracle/admin/sab/arch/*`	`sab` database archived log files

Table 1–7 Directory Structure Syntax for Identifying Classes of Files

`/*/oradata`	Oracle data directories
`/*/oradata/sab/*`	`sab` database files
`/*/oradata/sab/*.log`	`sab` database redo log files

OFA File Mapping

Table 1–8 shows an hierarchical file mapping of a sample OFA-compliant database, including each file's mount point, application, database, and tablespace. The file names indicate the file type (control, log, or data).

Table 1–8 Hierarchical File Mapping for OFA Installation

/					root mount point
u01/					'user data' mount point #1
	app/				subtree for app software
		oracle/			home for *oracle* software owner
			admin/		subtree for database admin files
				TAR/	subtree for Support logs
				db_name1/	admin subtree for *db_name1* database
				db_name2/	admin subtree for *db_name2* database
			doc/		online documentation
			local/		subtree for local Oracle software
				aps6/	an Oracle6 admin package
				aps7/	an Oracle7 admin package
			product/		distribution files
				7.3.2/	ORACLE_HOME for 7.3.2 instances
				7.3.3/	ORACLE_HOME for 7.3.3 instances
				8.0.4/	ORACLE_HOME for 8.0.4 instances
	home/				subtree for login home directories
		ltb/			home for a user
		sbm/			home for a user
	oradata/				subtree for Oracle data
		db_name1/			subtree for *db_name1* database files
		db_name2/			subtree for *db_name2* database files
u02/					'user data' mount point #2
	home/				subtree for login home directories
		cvm/			home for a user
		vrm/			home for a user
	oradata/				subtree for Oracle data
		db_name1/			subtree for *db_name1* database files
		db_name2/			subtree for *db_name2* database files
u03/					'user data' mount point #3
	home/				subtree for login home directories
	oradata/				subtree for Oracle data
		db_name1/			subtree for *db_name1* database files
		db_name2/			subtree for *db_name2* database files

Raw Device Sizes

Choose a small set of standard sizes for all raw devices that may be used to store Oracle database files.

In general, standardizing on a single size is recommended. If a single size is used, raw files can be moved from one partition to another safely. The size should be small enough so that a fairly large number can be created, but large enough to be convenient.

For example, a 2 GB drive could be divided into 10 partitions of 200 MB each—a good balance between size and number. Any tablespace using raw devices should stripe them across several drives. If possible, the striping should be done with a logical volume manager.

File Mapping for Multiple-Instance OFA Database

Multiple-instance databases (Oracle Parallel Server installations) have an additional guideline for file mapping.

Administrative Home for Oracle Parallel Server

When using the Oracle Parallel Server, select one node to act as the Oracle *administrative home* for the cluster. The administrative home contains the administrative subtree. Create subdirectories for each instance accessing the database within the `bdump`, `cdump`, `logbook`, `pfile`, and `udump` directories of `~/admin/`*`d`*`/`. The `admin` directory for the administrative home should be mounted as the `admin` directory for every instance. An example is shown in Table 1–9.

Table 1–9 Administrative Directory Structure for Dual-Instance Oracle Parallel Server

u01/app/oracle/admin/sab/			administrative directory for **sab** database
adhoc/			directory for miscellaneous scripts
arch/			log archive dest for all instances
	redo001.arc		archived redo log file
bdump/			directory for background dump files
	inst1/		background dump dest for *inst1* instance
	inst2/		background dump dest for *inst2* instance
cdump/			directory for core dump files
	inst1/		core dump dest for *inst1* instance
	inst2/		core dump dest for *inst2* instance
create/			directory for creation scripts
	1-rdbms.sql		SQL script to create *inst* database
exp/			directory for exports
	970920full.dmp		Sept 20 full export dump file
	export/		directory for export parfiles
	import/		directory for import parfiles
logbook/			directory for *inst* logbook entries
	inst1/		directory for *inst1* instance reports
		params.lst	v$parameter report for *inst1* instance
	inst2/		directory for *inst2* instance reports
		params.lst	v$parameter report for *inst2* instance
	user.lst		dba_users report
pfile/			directory for instance parameter files
	inst1/		directory for *inst1* instance parameters
		init	instance parameters for *inst1* instance
	inst2/		directory for *inst2* instance parameters
		init	instance parameters for *inst2* instance
udump/			directory for user dump files
	inst1/		user dump dest for *inst1* instance
	inst2/		user dump dest for *inst2* instance

Directory Structure

ORACLE_BASE Directory

ORACLE_BASE is the root of the Oracle directory structure. ORACLE_BASE directory structure and content is described in Table 1–10. When installing an OFA-compliant database using the Oracle Installer, ORACLE_BASE is by default, set to */pm/app/oracle*.

Table 1–10 ORACLE_BASE Directory Structure and Content

admin	administrative files
doc	online documentation
local	subtree for local Oracle software
product	Oracle software

ORACLE_HOME Directory

If you install an OFA-compliant Oracle Server, the ORACLE_HOME directory is /mount_point/app/oracle/product/*release_number*. ORACLE_HOME directory structure and content is described in Table 1–11. Under LINUX , the ORACLE_HOME directory contains the following subdirectories, as well as a subdirectory for each Oracle product:

Table 1–11 ORACLE_HOME Directory Structure and Content

bin	binaries for all products
ctx	ConText cartridge
dbs	init*sid*.ora, lk*sid*
jdbc	JDBC drivers
lib	Oracle product libraries
md	Spatial cartridge
mlx	Xerox Stemmer (for ConText cartridge)
network	Net8
nlsrtl	NLS runtime loadable data
ocommon	common files for all products
odg	data gatherer
opsm	Parallel Server Manager Components

Table 1–11 ORACLE_HOME Directory Structure and Content

`oracore`	core libraries
`orainst`	master installation files and programs
`ord`	data cartridges
`otrace`	Oracle TRACE
`plsql`	PL/SQL
`precomp`	precompilers
`rdbms`	server files and libraries required for the database
`slax`	SLAX parser
`sqlplus`	SQL*Plus
`svrmgr`	Server Manager

Oracle Product Subdirectories

Product subdirectories may include those described in Table 1–12, depending on the Oracle products available on your system and the products you purchase.

Table 1–12 Oracle Product Subdirectories

`network`	Oracle Net8
`ocommon`	libraries and SQL messages. All products depend on this directory, which is installed automatically
`plsql`	PL/SQL version 2, procedural option
`sqlplus`	SQL*Plus
`svrmgr`	Server Manager

Contents of Product Subdirectories

Each product subdirectory contains the subdirectories described in Table 1–13:

Table 1–13 Contents of Product Subdirectories

`admin`	administrative SQL and shell scripts (for example, `catalog.sql`, `catexp.sql`, and `demo.sql`)
`admin/*`	special directories for other products
`admin/resource`	resource files
`admin/terminal`	runtime terminal files

Table 1–13 Contents of Product Subdirectories

`demo`	demonstration scripts and datafiles
`doc`	README files (for example, `readmeLINUX.doc`)
`install`	product installation scripts
`lib`	product libraries and distributed makefiles
`log`	trace files and log files (for example, `orasrv.log` and `*.trc` files)
`mesg`	U.S. message files, and Multilingual Option (formerly National Language Support) message text and binary files (for example, `oraus.msg` and `oraus.msb`)

Examples of Product Subdirectories

Examples of product subdirectories and their contents are shown in Table 1–14.

Table 1–14 Examples of Product Subdirectories

`rdbms`	install, lib, admin, doc, mesg, log
`sqlplus`	install, demo, lib, admin, doc, mesg

File Naming Conventions in the admin Directory

The `rdbms/admin` directory contains the SQL scripts shown in Table 1–15.

Table 1–15 admin Directory, File Naming Conventions

`cat*.sql`	creates catalog and data dictionary tables and views. The following files are run automatically during installation: `catalog.sql` (for all installations) `catproc.sql` (for all installations) `catparr.sql` (for Parallel Server option installations) `catrep.sql` (for all installations)
`dbms*.sql`	additional database packages
`utl*.sql`	creates tables and views for database utilities

Filename Extensions

A description of filename extensions is shown in Table 1–16.

Table 1–16 Filename Extensions

`.a`	object file libraries; Ada runtime libraries

Table 1–16 Filename Extensions

`.ada`	Ada source files
`.aud`	Oracle audit file
`.bdf`	X11 font description file
`.bmp`	X11 bitmap file
`.c`	C source file
`.ctl`	SQL*Loader control file; Oracle Server control file
`.dat`	SQL*Loader datafile
`.dbf`	Oracle Server tablespace file
`.dei`	ORCA de-installation script
`.dmp`	Export file
`.doc`	ASCII text file
`.env`	shell script file for setting environment
`.f`	FORTRAN source file
`.h`	C header file; also, `sr.h` is a SQL*Report Writer help file
`.ins`	ORCA installation script
`.l`	LINUX manual page
`.lis`	output of SQL*Plus scripts
`.log`	installation log files; Oracle Server redo log files
`.map`	Installer product component files
`.mk`	`make` files
`.msb`	NLS message file (binary)
`.msg`	NLS message file (text)
`.o`	object module
`.ora`	Oracle configuration files
`.orc`	installation prototype files
`.pad`	Pro*Ada source file
`.pc`	Pro*C source file
`.pco`	Pro*COBOL source file
`.ppd`	printer driver file
`.pfo`	Pro*FORTRAN source file

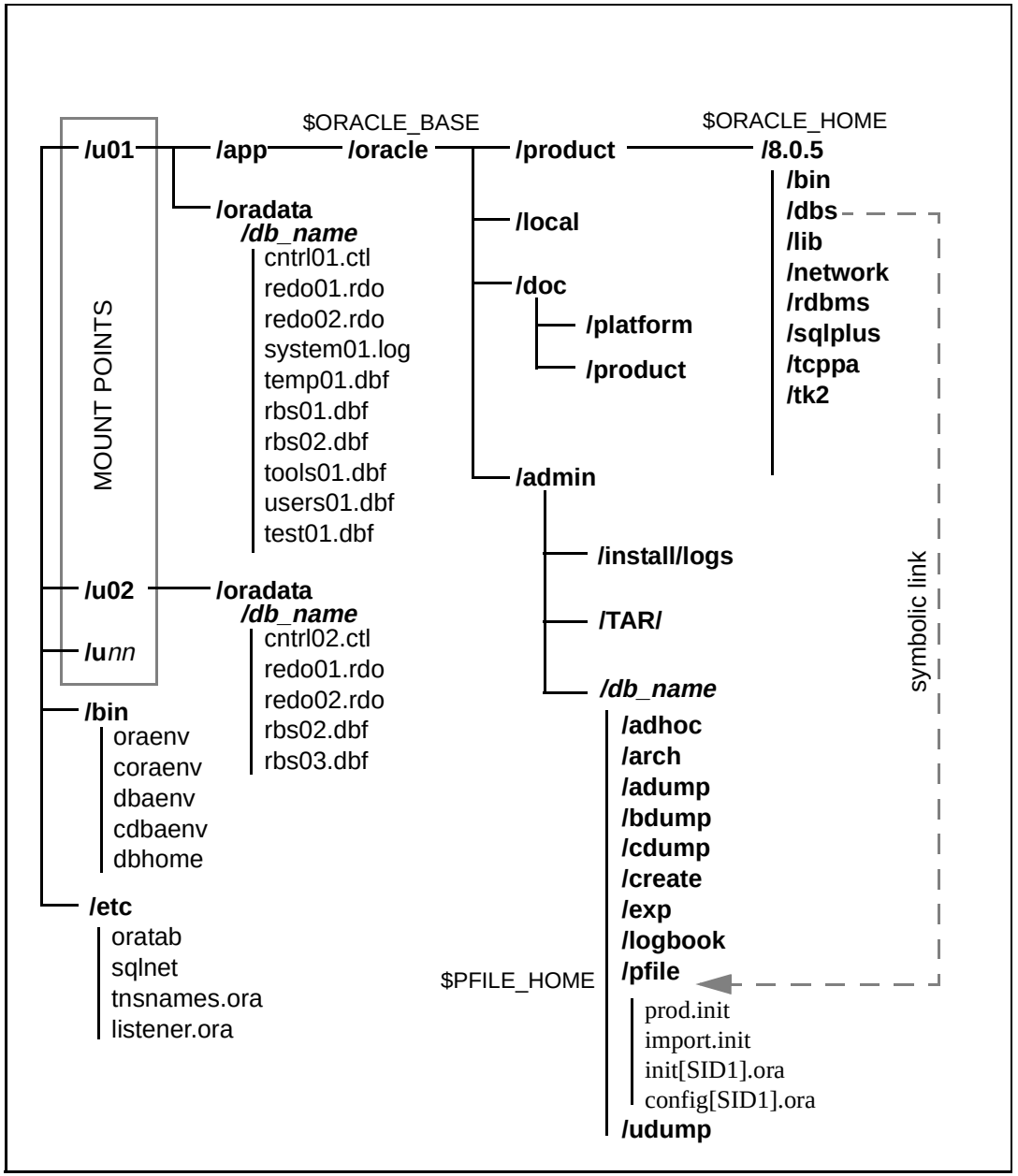
Table 1–16 Filename Extensions

`.prd`	product registration template file (for `orainst`)
`.res`	Toolkit II resource file
`.sh`	Bourne shell script file
`.sql`	SQL* script files
`.sys`	Bourne shell script file
`.tab`	SQL* script file
`.trc`	trace files
`.tut`	Bourne shell script file
`.us`	`orainst` message file
`.utd`	Uniform Terminal Definitions
`.vrf`	Installer Dependencies Verification Script

Default OFA Database

An OFA default database created using the Oracle Installer is shown in Figure 1–1.

Figure 1–1 Default Oracle Installation



2

Administering Oracle8 on Linux

- Customizing the initsid.ora File
- Setting the Environment
- Environment Variables for Oracle8
- Estimating Oracle8 Server Memory Usage
- Calculating Cluster Size and Index Size
- Initialization Parameters
- Controlling the System Global Area
- Managing Special Accounts and Groups
- Managing Security
- Administering Login Home Directories
- Building and Running Demonstrations
- Oracle8 Server SQL Reference

Customizing the init*sid*.ora File

This section documents the default `init`*sid*`.ora` file provided with the Oracle8 distribution. The Oracle Installer creates it in the `$ORACLE_BASE/admin/`*db_name*`/pfile` directory. You can modify it to customize your Oracle8 installation.

Some `init`*sid*`.ora` parameter settings are generic to any size installation. For those parameter settings requiring different values for different size installations, three scenarios are provided: small, medium, and large. In the sample `init`*sid*`.ora` file, parameters dependent on installation size are shown for each setting. You can comment out settings that do not apply to your installation by inserting a number sign (#) at the beginning of a line.

Table 2–1 suggests the approximate SGA sizes corresponding to the three scenarios provided for in the `init`*sid*`.ora` file.

Table 2–1 Block and SGA Sizes for Sample initsid.ora File

Installation/Database Size			
Block Size	**Small**	**Medium**	**Large**
2 KB	4500 KB	6800 KB	17000 KB
4 KB	5500 KB	8800 KB	21000 KB

Sample init*sid*.ora File

This file is provided by Oracle Corporation to assist in customizing the RDBMS installation. Some parameter settings are generic to any size installation. For parameters that require different values in different size installations, three scenarios are provided: SMALL, MEDIUM and LARGE. Any parameter that needs to be tuned according to installation size will have three settings, each one commented according to installation size.

```
# replace DEFAULT with your database name
db_name=DEFAULT

db_files = 80                                                    # SMALL
# db_files = 400                                                 # MEDIUM
# db_files = 1000                                                # LARGE

db_file_multiblock_read_count = 8                                # SMALL
# db_file_multiblock_read_count = 16                             # MEDIUM
# db_file_multiblock_read_count = 32                             # LARGE

db_block_buffers = 100                                            # SMALL
```

```
# db_block_buffers = 550                                        # MEDIUM
# db_block_buffers = 3200                                       # LARGE

shared_pool_size = 3500000                                      # SMALL
# shared_pool_size = 5000000                                    # MEDIUM
# shared_pool_size = 9000000                                    # LARGE

log_checkpoint_interval = 10000

processes = 50                                                  # SMALL
# processes = 100                                               # MEDIUM
# processes = 200                                               # LARGE

parallel_max_servers = 5                                        # SMALL
# parallel_max_servers = 4 x (number of CPUs)                   # MEDIUM
# parallel_max_servers = 4 x (number of CPUs)                   # LARGE

log_buffer = 8192                                               # SMALL
# log_buffer = 32768                                            # MEDIUM
# log_buffer = 163840                                           # LARGE

sequence_cache_entries = 10                                     # SMALL
# sequence_cache_entries = 30                                   # MEDIUM
# sequence_cache_entries = 100                                  # LARGE

sequence_cache_hash_buckets = 10                                # SMALL
# sequence_cache_hash_buckets = 23                              # MEDIUM
# sequence_cache_hash_buckets = 89                              # LARGE

# audit_trail = true            # if you want auditing
# timed_statistics = true       # if you want timed statistics
max_dump_file_size = 10240      # limit trace file size to 5 Meg each

# Uncommenting the line below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest = disk$rdbms:[oracle.archive]
# log_archive_format = "T%TS%S.ARC"

# If using private rollback segments, place lines of the following
# form in each of your instance-specific init.ora files:
# rollback_segments = (name1, name2)

# If using public rollback segments, define how many
# rollback segments each instance will pick up, using the formula
```

```
#   # of rollback segments = transactions / transactions_per_rollback_segment
# In this example each instance will grab 40/10 = 4:
# transactions = 40
# transactions_per_rollback_segment = 10

# Global Naming -- enforce that a dblink has same name as the db it connects to
global_names = TRUE

# Edit and uncomment the following line to provide the suffix that will be
# appended to the db_name parameter (separated with a dot) and stored as the
# global database name when a database is created.  If your site uses
# Internet Domain names for e-mail, then the part of your e-mail address after
# the '@' is a good candidate for this parameter value.

# db_domain = us.acme.com # global database name is db_name.db_domain

#_db_block_cache_protect = true                        # memory protect buffers
#event = "10210 trace name context forever, level 2" # data block checking
#event = "10211 trace name context forever, level 2" # index block checking
#event = "10235 trace name context forever, level 1" # memory heap checking
#event = "10049 trace name context forever, level 2" # memory protect cursors

# define parallel server (multi-instance) parameters
#ifile = ora_system:initps.ora

# define two control files by default
control_files = (ora_control1, ora_control2)

# Uncomment the following line if you wish to enable the Oracle Trace product
# to trace server activity.  This enables scheduling of server collections
# from the Oracle Enterprise Manager Console.
# Also, if the oracle_trace_collection_name parameter is non-null,
# every session will write to the named collection, as well as enabling you
# to schedule future collections from the console.

# oracle_trace_enable = TRUE
```

Setting the Environment

Displaying and Setting Environment Variables

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

```
$ echo $ORACLE_SID
```

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

For the Bourne or Korn shell, enter:

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

For the C shell, enter:

```
% setenv ORACLE_SID test
```

where *test* is the value of the variable ORACLE_SID.

Setting a Common Environment

Oracle8 allows a DBA to set a common environment for all users. A common environment makes it easier for system administrators and database administrators to make changes to the physical Oracle Server system.

The oraenv Command File

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

- a central means of updating all user accounts with database changes
- a mechanism for switching back and forth between Oracle Server databases

For example, a database needs to move from `/usr/oracle` to `/usr1/oracle`. Without a common environment-setting routine, user startup files would need to be updated individually. With `oraenv`, each user profile calls the `oraenv` command file and the changes must be made only to that file.

Local bin Directory

Placing `oraenv` (or `coraenv`) and `dbhome` in the local `bin` directory, separate from the Oracle software home directory, ensures that these files are accessible to all users. It also ensures that `oraenv` (`coraenv`) continues to work even if you change the path to point to a different `ORACLE_HOME`.

Moving Between Databases

To switch from one database or instance to another, call the `oraenv` routine, and reply to the prompt with the *sid* of the desired database. Always provide the full path of the `oraenv` command file. For example:

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

Database Examples

In the following examples, it is assumed your local bin directory is called `/usr/local/bin` and your production database is called PROD. If you prefer not to be prompted for the ORACLE_SID at startup, set the ORAENV_ASK environment variable to **no**.

In the examples below, ORAENV_ASK is reset to the default, **Yes**, after `oraenv` is executed. This ensures that the system prompts for a different ORACLE_SID the next time `oraenv` is executed.

Single Instance

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

```
. local_bin_directory/oraenv
```

with the following lines:

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

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

```
source local_bin_directory/coraenv
```

with the following lines:

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

Multiple Instances

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

For the Bourne or Korn shell**:**

```
PATH=${PATH}:/usr/local/bin
```

```
ORACLE_SID=PROD
export PATH ORACLE_SID
SIDLIST= 'awk -F: '/^[^#]/ {printf "%s ", $1}' /etc/oratab`
echo "SIDS on this machine are $SIDLIST"
ORAENV_ASK=
oraenv
```

For the C shell**:**

```
setenv PATH ${PATH}:/usr/local/bin
setenv ORACLE_SID PROD
set sidlist = 'awk -F: '/^[^#]/ {printf "%s ", $1}' /etc/oratab`
echo "SIDS on this machine are $sidlist"
unset ORAENV_ASK
source /usr/local/bin/coraenv
```

Environment Variables for Oracle8

Certain variables in the LINUX environment must be set prior to installation of the Oracle system.

See Also: *Oracle8 Installation Guide*.

Oracle Environment Variables on LINUX

Table 2–2 provides the syntax and examples for Oracle8 variables.

Table 2–2 Oracle8 Environment Variables on LINUX

Variable	Detail	Definition
EPC_DISABLED	Function	Disables oracle trace
	Syntax	`true` or `false`
	Example	`true`
NLS_LANG	Function	Specifies the language and character set used for output. See the *Oracle8 Installation Guide* for LINUX for a range of values.
	Syntax	language_territory.characterset
	Example	french_france.we8dec
ORA_NLS33	Function	Points to the directory where languages and character sets are stored.
	Set to	`$ORACLE_HOME/ocommon/nls/admin/data`

Table 2–2 Oracle8 Environment Variables on LINUX

Variable	Detail	Definition
ORACLE_BASE	Function	Specifies the base of the Oracle directory structure for OFA-compliant databases.
	Syntax	directory_path
	Example	/*mount_point*/app/oracle
ORACLE_HELP	Function	Specifies the directory containing help files.
	Syntax	directory_path
	Example	$ORACLE_HOME/help/admin/resource
ORACLE_HOME	Function	Specifies the directory containing the Oracle software distribution.
	Syntax	directory_path
	Example	/*mount_point*/app/oracle/product/*release_number*
ORACLE_PATH	Function	Specifies the search pathname for files used by Oracle applications, such as SQL*Plus. If not specified, the application reads from and writes to the current directory.
	Syntax	colon-separated list of directories *directory:directory:directory*
	Example	/u01/oracle/adhoc/sqlplus:. **Note:** The period adds the current working directory to the search path.
ORACLE_SID	Function	Specifies the Oracle System Identifier.
	Syntax	The string of numbers and characters must begin with a letter. For more information, see the *Oracle8 Installation Guide* for LINUX.
	Example	SAL1
ORACLE_TERM	Function	Specifies the terminal type identifier. Used by the Installer and Oracle products to determine the correct Toolkit II (.res) resource file. If not set, the value of the operating-system variable TERM is used.
	Syntax	string of characters
	Range of Values	The value of this variable must be set such that the pattern tk2c${ORACLE_TERM}.res corresponds to valid resource files in the Toolkit II resource directory or directories. See the *Oracle8 Installation Guide* for LINUX for a list of valid values.
	Example	vt100

Table 2–2 Oracle8 Environment Variables on LINUX

Variable	Detail	Definition
ORACLE_TERMINAL	Function	Specifies an additional directory to search for Toolkit II (`.res`) resource files.
	Syntax	directory_name
	Example	`$ORACLE_HOME/guicommon/tk21/admin/terminal`
ORACLE_TRACE	Function	Turns on tracing of Bourne shell scripts during install. If set to **T**, many Oracle shell scripts run with `set-x` flag on.
	Range of Values	**T** or anything else.
ORAENV_ASK	Function	Controls whether `(c)oraenv` prompts for ORACLE_SID or ORACLE_HOME. If set to `NO (c)oraenv` does not prompt and, if set to anything else, it does.
	Syntax	string
	Range of Values	`NO` or anything else.
TNS_ADMIN	Function	Sets the directory containing the Oracle Net8 configuration files.
	Syntax	directory_path
	Range of Values	Any directory; for more information, see the *Oracle8 Installation Guide* for LINUX.
	Example	`$ORACLE_HOME/network/admin`
TWO_TASK	Function	Sets the default Oracle Net8 connect string descriptor alias defined in the `tnsnames.ora` file.
	Syntax	available network alias
	Range of Values	Any valid Oracle Net8 alias defined in the `tnsnames.ora` file.
	Example	PRODDB_TCP

Note: Environment variables should not be defined with names that are identical to names of Oracle Server processes, for example: `arch`, `pmon`, and `dbwr`.

Abbreviations for ORACLE_HOME and ORACLE_SID

In Oracle8 Server files and programs, a question mark (?) represents the value of ORACLE_HOME. For example, Oracle8 expands the question mark in the following SQL statement to the full pathname of ORACLE_HOME:

```
alter tablespace TEMP add datafile '?/dbs/dbs2.ora' size 2M
```

The @ sign represents $ORACLE_SID. For example, to indicate that a file belongs to an instance, enter:

```
alter tablespace tablespace_name add datafile 'dbsfile@.ora'
```

LINUX Environment Variables Used with Oracle8

Table 2–3 provides the syntax and examples for LINUX environment variables used with Oracle8.

Table 2–3 LINUX Environment Variables Used with Oracle8

Variable	Detail	Definition
ADA_PATH	Function	Specifies the directory containing the Ada compiler.
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. See vendor's X Windows documentation for details.
	Syntax	hostname:display Hostname is the network identifier for the display device; display is a number which is almost always 0.
	Example	135.287.222.12:0 bambi:0
HOME	Function	The user's home directory.
LANG or LANGUAGE	Function	Specifies the language and character set used by the operating system for messages and other output. See the operating system documentation, and your *Oracle8 Installation Guide* for LINUX.
LDOPTS	Function	Specifies the default linker options on some platforms. See `man` pages on `ld` for details.
LPDEST	Function	Specifies the user's default printer for System V-based systems.
	Syntax	printer_name
	Example	`docqms`
LDPATH	Function	Default directories used by the linker to find shared object libraries. See `man` pages on `ld` for details.
LD_LIBRARY_PATH	Function	Used on some platforms by the shared library loader (`ld.so`) at runtime to find shared object libraries. See `man` pages on `ld.so` for details.
	Syntax	colon-separated list of directories *directory:directory:directory*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib`

Table 2–3 LINUX Environment Variables Used with Oracle8

Variable	Detail	Definition
PATH	Function	Used by the shell to locate executable programs; needs to include `$ORACLE_HOME/bin`.
	Syntax	colon-separated list of directories *directory:directory:directory*
	Example	`/bin:/usr/bin:/usr/local/bin: /usr/bin/X11:$ORACLE_HOME/bin:$HOME/bin`:. **Note**: The period adds the current working directory to the search path
PRINTER	Function	Selects the user's default printer for LINUX systems.
	Syntax	printer_name
	Example	`docqms`
SHELL	Function	Specifies the command interpreter used during a host command.
	Syntax	shell pathname
	Range of Values	`/bin/sh` or `/bin/csh` or `/bin/ksh` or any other command interpreter supplied with LINUX
	Example	`/bin/sh`
TERM	Function	Used by Oracle Toolkit II character mode tools to determine terminal types; also used by other LINUX tools for the same purpose.
	Example	vt100
TMPDIR	Function	Specifies the default directory for temporary disk files; if set, tools that create a temporary files do so in this directory.
	Syntax	directory_path
	Example	`/u02/oracle/tmp`
XENVIRONMENT	Function	Specifies a file containing X Windows system resource definitions. See your X Windows documentation for more information.

Setting the System Time

The TZ variable sets your time zone. Check your LINUX documentation to see if your operating system uses this environment variable.

It allows a user to adjust the clock for daylight saving time changes, or different time zones. The adjusted time is used to time-stamp files, produce the output of the `date` command, and obtain the current SYSDATE.

> **WARNING: Users are discouraged from changing their personal TZ value. Using different values of TZ such as GMT+24 may change the day a transaction is recorded. This affects Oracle applications that use SYSDATE, such as Oracle Financials. Use sequence numbers to order a table instead of date columns to avoid this problem.**

Estimating Oracle8 Server Memory Usage

Before starting the Oracle8 Server, virtual memory requirements can be estimated using this formula:

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

where *n* = number of background processes.

For each Oracle back-end connection, use the following formula to estimate virtual memory requirements:

```
      <size of oracle executable data section>
+     <size of oracle executables uninitialized data section>
+     <8192 bytes for the stack>
+     <2048 bytes for processes user area>
+     <cursor area needed for the application>
```

Use the `size` command to estimate an executable's text size, private data section size, and uninitialized data section size (or *bss*). Program text is only counted once, no matter how many times the program is invoked, because all Oracle executable text is always shared.

To compute actual Oracle physical memory usage while the database is up and users are connecting to it, use the `ps` command. Look for all the front end, server, and background Oracle process entries. For each entry, add the "real size of process" columns for the resident memory use subtotal. Now add the text size for the Oracle executable and every other Oracle tool executable

running on the system to that subtotal. Remember to count executable sizes only once, regardless of how many times the executable was invoked.

> **See Also:** Refer to your LINUX `man` pages or documentation for a list of available switches for the `ps` command.

Calculating Cluster Size and Index Size

Calculating Cluster Size

Use size guidelines in Table 2–4 to calculate cluster size using the formula in Appendix A of the *Oracle8 Administrator's Guide*.

Table 2–4 Cluster Size Values

Type	Size
Fixed header size	68 bytes
Variable transaction header	24**INITRANS* value for the table
Row directory	4 bytes per row of a clustered table

Calculating Index Size

Use Table 2–5 to calculate the size required by an index using the formula in Appendix A of the *Oracle8 Administrator's Guide*.

Table 2–5 Index Size Values

Type	Size
Fixed header size	113 bytes
Variable transaction header	24**INITRANS* value for the index
Entry header	5 bytes

Server Resource Limits

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

Disk quotas established for the Oracle `dba` user ID may hinder the operation of the Oracle8 system. Confer with your Oracle8 database administrator and the LINUX system manager before establishing disk quotas.

Initialization Parameters

Initialization parameters can be modified in the `init`*sid*`.ora` file for the Oracle8 Server instance.

See Also: *Oracle8 Administrator's Guide*.

Default Initialization Parameter Values

Table 2–6 lists default initialization parameter values on LINUX. All Oracle8 Server instances assume these values if you do not specify different values for them in the `init`*sid*`.ora` file. Oracle Corporation recommends that you include in the `init`*sid*`.ora` file only those parameters that differ from the default initialization parameter values.

To display the current values of these parameters on the system, use Server Manager to execute the SQL statement SHOW PARAMETERS.

See Also: *Oracle8 Server Reference*.

Table 2–6 Default Initialization Parameters

Parameter	Default Value
BACKGROUND_DUMP_DEST	`$ORACLE_BASE/admin/`*sid*`/bdump`
BITMAP_MERGE_AREA_SIZE	1048576
COMMIT_POINT_STRENGTH	1
CONTROL_FILES	`$ORACLE_HOME/dbs/ctrl@.dbf` (where @ represents ORACLE_SID)
CREATE_BITMAP_AREA_SIZE	8388608
DB_BLOCK_BUFFERS	200
DB_BLOCK_SIZE	2048
DB_FILES	80 (maximum of 2000000)
DB_FILE_DIRECT_IO_COUNT	64 (maximum of 1048576)
DB_FILE_MULTIBLOCK_READ_COUNT	8 (range of 1-128, but should not exceed one quarter of DB_BLOCK_BUFFERS)
DISTRIBUTED_TRANSACTIONS	16
HASH_AREA_SIZE	0
HASH_MULTIBLOCK_IO_COUNT	1

Table 2–6 Default Initialization Parameters

Parameter	Default Value
LOCK_SGA	FALSE
LOCK_SGA_AREAS	0
LOG_ARCHIVE_BUFFER_SIZE	64
LOG_ARCHIVE_BUFFERS	4 (maximum of 128)
LOG_ARCHIVE_DEST	`$ORACLE_HOME/dbs/arch/`
LOG_ARCHIVE_FORMAT	"%t_%s.dbf"
LOG_BUFFER	8192
LOG_CHECKPOINT_INTERVAL	10000
LOG_SMALL_ENTRY_MAX_SIZE	80
MTS_MAX_DISPATCHERS	5
MTS_MAX_SERVERS	20
MTS_SERVERS	0
MTS_LISTENER_ADDRESS	ADDRESS=*address* (See Chapter 6)
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
OBJECT_CACHE_MAX_SIZE_PERCENT	10
OBJECT_CACHE_OPTIMAL_SIZE	102400
OPEN_CURSORS	50
OS_AUTHENT_PREFIX	ops$
PROCESSES	50
SHARED_POOL_SIZE	3500000
SORT_AREA_SIZE	65536
SORT_READ_FAC	5
SORT_SPACEMAP_SIZE	512
USER_DUMP_DEST	`$ORACLE_BASE/admin/`*`sid`*`/udump`

Controlling the System Global Area

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

Size limits of the SGA

The maximum size of a single shared memory region is specified by the LINUX parameter SHMMAX. An SGA that is 2048 KB can use four shared memory regions of 512 KB each.

If the size of the SGA exceeds the maximum size of a shared memory segment (SHMMAX), Oracle8 attempts to attach more contiguous segments to fulfill the requested SGA size. SHMSEG is the maximum number of segments that can be attached by a process. To attach the segments at contiguous addresses, SHMMAX must be set to its maximum value on systems where its size is limited.

> **Note:** Intimate Shared Memory (ISM) may cause problems when SHMMAX is smaller than the database SGA size.

The following `init.ora` parameters control the size of the SGA:

- DB_BLOCK_BUFFERS
- DB_BLOCK_SIZE
- SORT_AREA_SIZE
- SHARED_POOL_SIZE

Use caution when setting values for these parameters. When values are set too high, too much of the machine's physical memory is devoted to shared memory resulting in poor performance. As a guideline, the total of all instance's SGA sizes should be no more than one-third of the total physical RAM.

Calculating the Size of the SGA

The approximate size of an instance's SGA can be calculated with this formula:

```
  (DB_BLOCK_BUFFERS × DB_BLOCK_SIZE)
+ SORT_AREA_SIZE
+ SHARED_POOL_SIZE
+ 1MB
```

To display the size of the SGA for a running database in bytes, use the Server manager `show sga` command. This command displays the size of the SGA in bytes.

Relocating the SGA

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

1. Determine the valid virtual address range for attaching shared memory segments (in the resulting `tstshm` display, the lines "Lowest shared memory address" and "Highest shared memory address" indicate the valid range):

   ```
   $ tstshm
   ```

   Note: The system may experience problems when executing `tstshm` while using Intimate Shared Memory (ISM).

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

3. Determine the size of your SGA. SGA size is displayed next to the heading Total System Global Area when your database system starts.

4. Move to the `$ORACLE_HOME/rdbms/lib` directory, and run `genksms` to generate the file `ksms.s`:

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

   where *sgabeg* is the starting address of the SGA (which defaults to 0x80000000), and should fall within the range determined in step 2.

5. Shut down the existing Oracle database.

6. Rebuild the `oracle` executable in the `$ORACLE_HOME/rdbms/lib` directory:

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

   Using `ioracle`:

   - backs up the old executable (`oracle0`)
   - assigns the correct privileges to the new `oracle` executable
   - moves the new executable into the `$ORACLE_HOME/bin` directory

   The result is a new Oracle kernel that loads the SGA at the address specified by `sgabeg`.

genksms -b

The `genksms -b` utility is used to adjust the starting point of fixed SGA.

Managing Special Accounts and Groups

The DBA should be familiar with special accounts required by the Oracle Server, and should make sure these accounts belong to the appropriate groups. The following section describes special user accounts. LINUX accounts are described in Table 2–7, Oracle server accounts are described in Table 2–8.

Table 2–7 LINUX Accounts

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

Table 2–8 Oracle Server Accounts

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

.

Table 2–9 Special Group Accounts

`dba` group	The *oracle* software owner is the only required member of the `dba` group. You can add the `root` user, or any other LINUX user, to the `dba` group. Members of this group have access to Server Manager specially privileged functions. If your account is not a member of the `dba` group, you must enter a password in order to connect as INTERNAL or gain access to the other administrative functions of Server Manager. The default group ID is `dba`.
`oper` group	This is an optional LINUX group you can create. Members have database OPERATOR privileges. OPERATOR privileges are a restricted set of `dba` privileges.
`root` group	Only the `root` user should be a member of the `root` group.

Managing Security

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

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

Groups and Security

To ensure greater security on an Oracle8 database, create user groups at the operating system level. Groups are controlled by the LINUX file `/etc/group`. Oracle programs are divided into two sets for security purposes: those executable by all (*other*, in LINUX terms), and those executable by DBAs only. A recommended approach to security is:

- Before installing the Oracle Server, create a database administrators' group *(*`dba`*)* and assign the `root` and *oracle* software owner IDs to this group. Programs executable by `dba` only have permission `710`. Server Manager system-privileged commands are assigned automatically to the `dba` group upon installation.
- Add an `oracle` group of authorized users to allow a subset of LINUX users limited access to Oracle8. Give Oracle utilities the `oracle` group ID. Publicly executable programs, such as SQL*Plus, should be executable by this group. Set the permissions on the utilities to `710` to grant execute permissions to this group, but not *other.*
- Grant permission `711` to programs executable by *other.* Restrict this permission to programs that do not affect database security.

Although you can assign any name to the database administrators' group, `dba` is the default group name, and the convention used in this document. If you change this group name, the Oracle Installer relinks the kernel automatically during Installation. If you have multiple databases with the same ORACLE_HOME (a configuration which Oracle Corporation *strongly* discourages), they should have the same database administrators' group. These restrictions do not apply to the group name for ordinary users (known as the `oracle` group).

WARNING: Even though both the *oracle* software owner and `root` user should belong to the `dba` group, the *oracle* software owner should *not* be a member of the `root` group. The `root` user should be the *only* member of the `root` group.

Security for Oracle Server Utilities

Protect the Oracle8 executables from unauthorized use. The method you use depends on your environment and whether you use single-task utilities. These are suggestions for protecting Oracle8 executables:

- Keep all programs in the `$ORACLE_HOME/bin` directory and give ownership to the *oracle* software owner.
- Give all user utilities (`sqlplus, exp, imp`) a protection of `711` so all users on the machine can access the Oracle Server.
- Give all DBA utilities (such as Server Manager) a protection of `700` to restrict the use of these utilities to the DBA username, usually the *oracle* software owner.

Security for Server Manager Commands

If you do not have SQL*Plus, you can use Server Manager to make SQL queries. However, be careful how you assign access to Server Manager. The following system-privileged statements should not be accessible to anyone but the *oracle* software owner and the `dba` group users, as they grant special operating system privileges:

- STARTUP
- SHUTDOWN
- CONNECT INTERNAL

WARNING: System-privileged statements can damage your database if used incorrectly. Note that non-dba group users can connect as internal if they have the necessary password.

Security for Database Files

The user ID used to install Oracle8 should own the database files. The default user ID is the *oracle* software owner. Set the authorizations on these files to permission 0600: read/write (rw) by owner only, with no write authorizations for group or other users.

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

To access the protected database files, the `oracle` program must have its set user ID (`setuid`) bit on. To set this bit, enter:

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

This sets the authorization for the `oracle` program to:

```
-rwsr-s--x 1 oracle dba  443578 Mar 10 23:03 oracle
```

Setting the User ID

The Oracle Installer automatically sets the user ID. The `s` in the user execute field means when you execute the `oracle` program, it has an effective user ID of *oracle*, regardless of the actual user ID of the person invoking it.

Network Security

Using Passwords on the Network

Remote users on the network can enter their passwords in clear or encrypted text. When you use clear text, passwords can be picked up by unauthorized users, resulting in a breach of security. Oracle Net8 supports encrypted passwords.

DBA Privileges Over the Network

To control DBA privileges over the network choose one of the following options:

- set remote DBA access to `denied` in the `/var/opt/oracle/listener.ora` file
- set a special password in `orapwd` for DBA privileges

Automatic (ops$) Logins

Oracle8 supports automatic logins (operating system authorized logins) over the network.

LINUX treats a dollar sign ($) as the beginning of an environment variable. Therefore, when you specify an operating system authorized (`ops$`) login on the command line or in a script, first escape the `$` with a backslash (\). For example, user ID `scott` should specify `ops\$scott` when logging in remotely.

Automatic logins are not allowed for the `root` user ID.

Note: Automatic logins by PC, Apple MacIntosh, and OS/2 users are not secure. Anyone can edit the Oracle configuration file and change their user ID. For security reasons, if users of these systems are logging in over the network, Oracle Corporation strongly recommends you disable the `ops$` logins in the `listener.ora` file.

Enabling Automatic Logins for Oracle Net8

Automatic and remote DBA logins are not controlled by Oracle Net8. They are controlled by the Oracle8 Server and configured using parameters in the `init`*sid*`.ora` file. Although automatic logins are supported, they are disabled by default. To enable them, set the REMOTE_OS_AUTHENT initialization parameter to `true`, then start up the database.

Because `oracle` controls these logins, it is not necessary to run the Oracle Net8 `listener` as `setuid` to `root`.

See Also: Configuring Oracle Net8 is described in Chapter 6.

To perform an automatic login with Oracle Net8, create a user called `daemon` in your `/etc/passwd` file. The `daemon` user must not have an `ops$` account in any of the local databases, nor be in any of the DBA groups. That is, there should be no `ops$daemon` account that would allow an outside user to intrude into your local database.

DBA Group ID Keywords

Table 2–10 describes the keywords used in the `/var/opt/oracle/listener.ora` file to enable and control remote logins:

Table 2–10 Keywords Used to Control Remote Logins

DBA_GROUP	Use this keyword if the name is constant for all instances serviced by the listener.
DBA_GROUP_*sid*	Use this keyword for each ORACLE_SID if the listener services more than one `$ORACLE_HOME`, and the group IDs are different.
OPS_DOLLAR_LOGIN_ALLOWED OPS_DOLLAR_LOGIN_DENIED	Use these keywords to control remote login**.** OPS_DOLLAR_LOGIN_DENIED is the default.
REMOTE_DBA_OPS_ALLOWED REMOTE_DBA_OPS_DENIED	Use these keywords to control remote DBA access**.** REMOTE_DBA_OPS_DENIED is the default.

If the DBA group ID for the database accessed is not the default name (`dba`), you can specify a non-default name.

Set remote login and remote DBA access parameters to the individual ORACLE_SIDs of databases on the network, or specify all *sid*s at once. For example, either of the following statements are valid:

```
PARAMETER=ALL_SIDS
PARAMETER=sid1[, sidn...]
```

To see which privileges are assigned to the *sid*s, enter:

```
$ lsnrctl status
```

Checking Order

The system checks remote login parameters in the following order:

1. parameters that deny access
2. parameters that permit access
3. the default value (denied)

These privileges are implemented by manipulating the user ID and group ID of the shadow process forked by the Oracle Net8 listener. For example:

- If OPS_DOLLAR_LOGIN_DENIED is `true` for a particular instance, or if the user ID as reported by the client-side operating system has no account on the database host machine, the user ID and group ID are found in the `/etc/passwd` file under the entry for `daemon`.
- If both OPS_DOLLAR_LOGIN_ALLOWED and REMOTE_DBA_OPS_ALLOWED are `true` for a particular ORACLE_SID**,** and if the user ID as reported by the client

operating system does have an account on this system, the user ID and group ID are found in `/etc/passwd` for this user ID.

- If OPS_DOLLAR_LOGIN_ALLOWED is `true` for a particular ORACLE_SID, but REMOTE_DBA_OPS_ALLOWED is `false`, then, if the user ID has DBA privileges, the process has the user ID and group ID of `daemon`. Otherwise, the process has the user ID and group ID of this user.

> **Note:** REMOTE_DBA_OPS_ALLOWED is `false` by default. Oracle Corporation recommends that you do not change this value. When this parameter is set to `false`, users with DBA privileges cannot make operating system authorized logins over the network. They can, however, proceed with ordinary (password-protected) network logins.

Security and Remote Passwords

You can access or administer a database from a remote machine, such as a personal computer, without operating system accounts. User validation is accomplished by using an Oracle8 password file, created and managed by the `orapwd` utility. You can also use password file validation on systems that support operating system accounts.

Local password files are in the `$ORACLE_HOME/dbs` directory and contain the username and password information for a single database. If there are multiple `$ORACLE_HOME` directories on a machine, each has a separate password file.

Running orapwd

The `orapwd` utility exists in `$ORACLE_HOME/bin` and is run by the *oracle* software owner. Invoke `orapwd` by entering:

```
$ orapwd file=$ORACLE_HOME/dbs/orapwsid password=password entries=max_users
```

This syntax is described in Table 2–11:

Table 2–11 Syntax for Executing orapwd

file	is the name of the file where password information is written. The name of the file must be `orapws`*sid*, and you must supply the full pathname. Its contents are encrypted and not user-readable. This parameter is mandatory.
password	is the initial password you selected for INTERNAL and SYS. You can change this password after you create the database using an ALTER USER statement. This parameter is mandatory.

Table 2–11 Syntax for Executing orapwd

entries	is the maximum number of users allowed to connect to the database as SYSDBA or SYSOPER. This parameter is mandatory only if you want this password file to be EXCLUSIVE.

Note: You must create a new password file if you ever need to increase the maximum number of users. Therefore, set *max_users* to a higher number than you expect to require.

orapwd Example

```
$ orapwd file=/u01/app/oracle/product/8.0.5/dbs/orapwV805 \
 password=manager entries=30
```

See Also: *Oracle8 Server Administrator's Guide*.

Shared Password File for Multiple Databases

The default password file `/dbs/orapwd` should be used when the initialization parameter REMOTE_LOGIN_ PASSWORDFILE is set to SHARED for multiple databases. There is no *sid* specific password file for multiple databases.

Access to a Database from a Remote PC

When there is an Oracle8 password file, networked PC users can access this database as INTERNAL. Non-privileged users can connect to the database by invoking an Oracle application that uses the database. Privileged users who want to perform DBA functions on the database can enter the appropriate Server Manager command from their PC, adding the `dba` user password. For example:

```
SVRMGR> connect internal/dba_password
```

To connect as OPERATOR, use the same command with the OPERATOR password.

Remote Authentication

The following `init`*sid*`.ora` parameters, shown in Table 2–12 control the behavior of remote connections through non-secure protocols:

Table 2–12 Parameters For Controlling Remote Connections

REMOTE_OS_AUTHENT	enables or disables `ops$` connection
OS_AUTHENT_PREFIX	used by `ops$` accounts
REMOTE_OS_ROLES	enables or disables roles through remote connections

Note: If REMOTE_OS_AUTHENT is set to `true`, users who are members of the `dba` group on the remote machine are able to connect as INTERNAL without a password.

User-Visible Effects of the Shutdown Mechanism

Clients connected to an Oracle instance while a shutdown takes place will receive one of the following error messages upon subsequent SQL operations.

```
ORA-03113: end-of-file on communication channel
ORA-12571: TNS:packet writer failure
```

Administering Login Home Directories

To add or move login home directories without modifying programs that refer to them, you must:

- refer to explicit path names in files designed to store them, for example: `/etc/passwd` and `/etc/oratab`
- refer to group memberships in the `/etc/group` file

It is not necessary to record a pathname except in a central reference file, because a user's home directory can be derived in either of the following ways:

- C shell and Korn shell users can use ~*login* to refer to a user's home directory.
- Bourne shell users can construct a simple program to do this. See the sample `lhd` script later in this section.

Similarly, group memberships are computed from `/etc/group.` See the sample `grpx` script later in this section.

> **Note:** Local general-purpose utilities such as these should be stored in the `/var/opt/bin` directory.

Sample lhd Script

```
#!/bin/sh
#
# lhd - print login home directory name for a given user
#
# SYNTAX
# lhd [login]
#
prog=‘basename $0‘
if [ $# -eq 0 ] ; then
    login=‘whoami‘

elif [ $# -eq 1 ] ; then
    login=$1
else
    echo ”Usage: $prog login” >&2

    exit 2
fi
awk -F: ’$1==login {print $6}’ login=$login /etc/passwd
```

Sample grpx Script

```
#!/bin/sh
# grpx - print the list of users belonging to a given group
#
prog=‘basename $0‘
if [ $# -ne 1 ] ; then
    echo ”Usage: $prog group” >&2
    exit 2
fi
g=$1
# calculate group id of g
gid=‘awk -F: ’$1==g {print $3}’ g=$g /etc/group‘
# list users whose default group id is gid
u1=‘awk -F: ’$4==gid {print $1}’ gid=$gid /etc/passwd‘
# list users who are recorded members of g
u2=‘awk -F: ’$1==g {gsub(/,/,” ”); print $4}’ g=$g /etc/group‘
# remove duplicates from the union of the two lists
echo $u1 $u2 | tr ” ” ”\012” | sort | uniq | tr ”\012” ” ”
echo
```

Example 2–1 Using lhd and grpx Scripts

This example shows how the administrator can propagate a skeleton `.profile` file to the home directory for each member of a group. If the membership list of the `clerk` group changes, the code does not require modification.

```
$ for u in ‘grpx clerk‘ ; do
> cp /etc/skel/.profile ‘lhd $u‘
> done
```

Building and Running Demonstrations

Loading PL/SQL Demonstrations

PL/SQL includes a number of sample programs you can load. Demonstration and message files are in the `rdbms` directory. Perform these steps with the Oracle8 Server open and mounted:

1. Invoke Server Manager and connect with the user/password `scott/tiger`:

```
$ cd $ORACLE_HOME/plsql/demo
$ svrmgrl
SVRMGR > connect scott/tiger
```

2. To load the demonstrations, invoke `exampbld.sql` from Server Manager:

```
SVRMGR > @exampbld
```

> **Note:** Build the demonstrations under any Oracle account with sufficient permissions. Run the demonstrations under the same account you used to build them.

Running PL/SQL Demonstrations

Table 2–13 lists the kernel demonstrations.

Table 2–13 Kernel Demonstrations

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

Table 2–14 lists the precompiler demonstrations.

Table 2–14 Precompiler Demonstrations

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

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

```
$ sqlplus scott/tiger
SQLPLUS > @examp1
```

To build the precompiler PL/SQL demonstrations, enter:

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

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

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

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

```
$ examp9
```

In order to run the `extproc` demo, you first have to add the following line to the file, `tnsnames.ora`:

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

and the following line to the file, `listener.ora`:

```
SC=(SID_NAME=extproc)(ORACLE_HOME=/vobs/oracle)(PROGRAM=extproc))
```

then from your current shell, enter:

```
SVRMGR> connect scott/tiger
Connected.
SVRMGR> connect system/manager
Connected.
SVRMGR> grant create library to scott;
Statement processed.
SVRMGR> connect scott/tiger
Connected.
SVRMGR> create library demolib as
'$ORACLE_HOME/plsql/demo/extproc.so';
Statement processed.
```

Then finally, to run the tests:

```
SVRMGR> connect scott/tiger
Connected.
SVRMGR> @extproc
```

SQL*Loader Demonstrations

SQL*Loader demonstrations require that:

- the user `scott/tiger` has CONNECT and RESOURCE privileges
- the EMP and DEPT tables exist and are empty

To create and run a demonstration:

1. Connect to the database as the user/password `scott/tiger` from Server Manager (line mode).
2. Run the `ulcase`*n*`.sql` corresponding to the demonstration you want to run.
3. As `scott/tiger`, invoke the demonstration from the command line:

   ```
   $ sqlldr scott/tiger ulcase n
   ```

As `scott/tiger`, run the SQL*Loader demonstrations in the following order:

- `ulcase1`: Follow steps 1 - 3 above.
- `ulcase3`: Follow steps 1 - 3 above.
- `ulcase4`: Follow steps 1 - 3 above.
- `ulcase5`: Run the `ulcase*.sql` script as `scott/tiger`, then enter the following at the command line:

  ```
  $ sqlldr scott/tiger ulcase*
  ```

- `ulcase2`: Invoke the demonstration (you do not have to run the `ulcase2.sql` script).
- `ulcase6`: Run the `ulcase6.sql` script as `scott/tiger`, then enter the following at the command line:

  ```
  $ sqlldr scott/tiger ulcase1 DIRECT=true
  ```

- `ulcase7`:Run the `ulcase6.sql` script as `scott/tiger`, then enter the following at the command line:

  ```
  $ sqlldr scott/tiger ulcase7
  ```

Administering SQL*Loader

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

File Processing Option

The SQL*Loader release 1.1 control file includes the following additional file processing option strings, the default being `str`, which takes no argument:

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

`str` (the default) specifies a stream of records, each terminated by a newline character, which are read in one record at a time.

`fix` indicates that the file consists of fixed-length records, each of which is *n* bytes long, where *n* is an integer value.

`var` indicates that the file consists of variable-length records, each of which is *n* bytes long, where *n* is an integer value specified in the first five characters of the record.

If the file processing options are not selected, the information is processed by default as a stream of records (`str`). You might find that `fix` mode yields faster performance than the default `str` mode because it does not need to scan for record terminators.

Newlines in Fixed Length Records

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

For example, to read the following file:

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

specify `fix 4` instead of `fix 3` to account for the additional newline character.

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

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

Removing Newlines

Use the `position(x:y)` function in the control file to discard the newlines from fixed length records rather than loading them. To do this, enter the following in your control file:

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

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

Oracle Security Server

See Also: For information on the Oracle Security Server, see the Oracle Security Server Guide.

Oracle8 Server SQL Reference

CREATE CONTROLFILE Parameters

Use the parameter values in Table 2–15 to determine the size of control files for a database.

Table 2–15 Determining the Size of Control Files

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

3

Tuning Oracle8 on LINUX

- The Importance of Tuning
- LINUX Tools
- SQL Scripts
- Tuning Memory Management
- Tuning Disk I/O
- Monitoring Disk Performance
- Tuning CPU Usage
- Tuning Oracle Resource Contention
- Tuning Block Size and File Size
- Tuning the LINUX Buffer Cache Size
- Using Trace and Alert Files

The Importance of Tuning

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

Before Tuning the System

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

See Also: *Oracle8 Parallel Server Concepts and Administration*. *Oracle8 Tuning*.

LINUX Tools

LINUX provides performance monitoring tools that can be used to assess database performance and determine database requirements.

In addition to providing statistics for `oracle` processes, these tools provide statistics for CPU usage, interrupts, swapping, paging, and context switching for the entire system.

See Also: LINUX tools are described in the operating system documentation.

vmstat

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

```
% vmstat -n 5 8
```

Sample output from the `vmstat` command is shown in Figure 3–1.

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

Figure 3–1 Output from the vmstat -n command

```
procs                  memory    swap        io    system         cpu
r b w  swpd  free  buff cache  si  so   bi   bo   in   cs  us  sy  id
0 0 0 16124   964   524 29904   2  14   13   21  208    3   9   4  87
1 0 0 16124   648   524 30140   0   0    1    0  806 1763   5   8  87
1 0 0 16124   608   524 29904   0   0    0    0  856 1894   5   7  87
0 0 0 16124   612   524 29624   0   0    0    5  734 1586   5   8  88
2 0 0 16124  1656   520 28296   0   0  221    0  687 1395  14  10  77
0 0 0 16124   840   520 29060   0   0   38    0  621 1287   3   4  93
0 0 0 16124   856   520 29196   0   0    0    0  647 1395   4   6  91
1 0 0 16124   708   520 29288   0   0    1    0  618 1287   3   4  93
```

free

The `free` utility reports information about swap space usage. A shortage of swap space can result in the system hanging and slow response time.

SQL Scripts

utlbstat and utlestat SQL Scripts

The `utlbstat` and `utlestat` SQL scripts are used to monitor Oracle database performance and tune the Shared Global Area (SGA) data structures. For information regarding these scripts, see the *Oracle8 Server Tuning*. On LINUX, the scripts are located in `$ORACLE_HOME/rdbms/admin/`.

Tuning Memory Management

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

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

Allocate Sufficient Swap Space

Swapping causes significant LINUX overhead and should be minimized. Use `free` or `vmstat -n` on LINUX to check for swapping.

If your system is swapping and you need to conserve memory:

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

Procedures for adding swap space vary between LINUX implementations. On LINUX `free` to determine how much swap space is currently in use. Use `free` to add swap to your system. Consult your LINUX documentation for further information

Start with swap space two to four times your system's random access memory (RAM). a higher value if you plan to use CASE, Oracle Applications, or Oracle Office. Monitor use of swap space and increase it as necessary.

Control Paging

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

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

Use `vmstat` or free to monitor paging. The following columns are from `the vmsta` output are important:

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

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

- install more memory
- move some of the work to another system
- configure your kernel to use less memory

Hold the SGA in a Single Shared Memory Segment

Although this performance gain is minor, you cannot start the database without configu sufficient shared memory.

You may need to reconfigure the LINUX kernel to increase shared memory. The LINU kernel parameters for shared memory include SHMMAX, SHMMNI, and SHMSEG. In

order to ensure that the SGA resides in a single shared memory segment, set the value of SHMAX to 4294967295 (4 GB).

The size of the SGA can be estimated using the following steps:

1. Multiply DB_BLOCK_BUFFERS by DB_BLOCK_SIZE.
2. Add the result of Step 1 to SORT_AREA_SIZE.
3. Add the result of Step 2 to SHARED_POOL_SIZE.
4. Add the result of Step 3 to LOG_BUFFER.

You can also use the LINUX utility `ipcs` to monitor the status of shared memory.

> **See Also:** "Configure LINUX Kernel for Oracle" in Chapter 2 of the *Oracle8 Installation Guide for LINUX*.

Tuning Disk I/O

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

Tune the Database Writer to Increase Write Bandwidth

Oracle offers solutions to prevent database writer (DBWR) activity from becoming a bottleneck:

- use asynchronous I/O
- use I/O slaves

Asynchronous I/O

Asynchronous I/O allows processes to proceed with the next operation without having to wait after issuing a write and therefore improves system performance by minimizing idle time. Solaris supports Asynchronous I/O to both raw and filesystem datafiles.

I/O Slaves

I/O Slaves are specialized processes whose only function is to perform I/O. They are new with Oracle8, and replace Multiple DBWRs (in fact, they are a generalization of Multiple DBWRs and can be deployed by other processes as well), and can operate whether or not

asynchronous I/O is available. I/O Slaves come with a new set of initialization parameters which allow a degree of control over the way they operate. These are shown in Table 3–1.

Table 3–1 Initialization Parameters for I/O Slaves

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

There may be times when the use of asynchronous I/O is not desirable or not possible. The first two parameters in Table 3–1, DISK_ASYNCH_IO and TAPE_ASYNCH_IO, allow asynchronous I/O to be switched off respectively for disk and tape devices. Because the number of I/O Slaves for each process type defaults to zero, no I/O Slaves will be deployed unless specifically set.

DBWR_IO_SLAVES should only be set to greater than 0 if ASYNC I/O (that is, DISK_ASYNCH_IO, or TAPE_ASYNCH_IO) has been disabled, otherwise DBWR will become a bottleneck. In this case the optimal value on LINUX for DBWR_IO_SLAVES should be 4. In the case of LGWR_IO_SLAVES, it is not recommended to deploy more than 9 slaves.

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

Monitoring Disk Performance

To monitor disk performance, use `vmstat.`

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

Key indicators are:

- The sum of `bread, bwrit, pread` and `pwrit` indicates the state of the disk I/O subsystem. The higher the sum, the greater the potential for disk I/O bottlenecks. The larger the number of physical drives, the higher the sum threshold number can be. A good default value is no more than 40 for two drives and no more than 60 for four to eight drives.
- The `%rcache` should be greater than 90 and `%wcache` should be greater than 60. Otherwise, the system may be disk I/O bound.
- If `%wio` is consistently greater than 20, the system is I/O bound.

Disk Performance Issues

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

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

Tuning CPU Usage

Keep All Oracle Users/Processes at the Same Priority

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

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

Use Processor Affinity/Binding on Multi-Processor Systems

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

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

Jobs

If you need to transfer large amounts of data between the user and Oracle8 (for example, using `export/import`), it is efficient to use single-task architecture. To make the single-task import (`impst`), export (`expst`), and SQL*Loader (`sqlldrst`) executables, use the `ins_rdbms.mk` makefile, which can be found in the `$ORACLE_HOME/rdbms/lib` directory.

The following example makes the `impst`, `expst`, and `sqlldrst` executables:

```
% cd $ORACLE_HOME/rdbms/lib
% make -f ins_rdbms.mk expst impst sqlldrst
```

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

Tuning Oracle Resource Contention

Tune LINUX Kernel Parameters

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

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

Look for memory mapped video drivers, networking drivers, and disk drivers. They can often be de-installed, yielding more memory for use by other processes.

WARNING: Remember to make a backup copy of your LINUX kernel. See your hardware vendor documentation for additional details.

Tuning Block Size and File Size

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

Specifying Oracle Block Size

On LINUX, the default Oracle block size is 2KB and the maximum block size is 16KB.

You can set the actual block size to any multiple of 2KB up to 16KB, inclusive.

The optimal block size is typically the default, but varies with the applications. To create a database with a different Oracle block size, add the following line to the `init`*`sid`*`.ora` file:

```
db_block_size=new_block_size
```

Tuning the LINUX Buffer Cache Size

To take full advantage of raw devices, adjust the size of the Oracle8 buffer cache and, if memory is limited, the LINUX buffer cache.

The LINUX buffer cache is provided by the operating system. It holds blocks of data in memory while they are being transferred from memory to disk, or vice versa.

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

When moving to raw devices, increase the size of the Oracle8 buffer cache. If the amount of memory on the system is limited, make a corresponding decrease in the LINUX buffer cache size.

The LINUX command `vmstat` may help you determine which buffer caches should be increased or decreased.

Adjusting Cache Size

- Increase Oracle8 cache size as long as the cache hit ratio goes up.
- Decrease cache sizes if the swapping/paging activity becomes high.

Using Trace and Alert Files

This section describes the trace (or dump) and alert files the Oracle Server creates to diagnose and resolve operating problems.

Trace File Names

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

Table 3–2 Format Key to Process Name

processname	is a three- or four-character process name showing which Oracle8 process the trace file is from (for example, PMON, DBWR, ORA, or RECO)
sid	is the instance system identifier
pid	is the LINUX process ID number
`.trc`	is a file name extension appended to all trace file names

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

Alert Files

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

4

Administering SQL*Plus on Linux

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

Administering SQL*Plus

Setup Files

The setup files for SQL*Plus are `glogin.sql`, the global setup file which defines the site profile, and `login.sql`, which defines the user profile. The `glogin.sql` and `login.sql` files contain either SQL statements or SQL*Plus commands that you choose to execute at the beginning of each SQL*Plus session. When you invoke SQL*Plus, `glogin.sql` is read first, followed by `login.sql`.

The Site Profile

The Site Profile file is `$ORACLE_HOME/sqlplus/admin/glogin.sql`. SQL*Plus executes this command file whenever any user starts SQL*Plus and SQL*Plus establishes the Oracle connection. The default Site Profile is placed in `$ORACLE_HOME/sqlplus/admin` whenever SQL*Plus is installed. If a Site Profile already exists, it will be overwritten. An existing Site Profile is deleted whenever SQL*Plus is de-installed.

The User Profile

The User Profile file is `login.sql`. SQL*Plus attempts to execute this command file whenever any user starts SQL*Plus and SQL*Plus establishes the Oracle connection. The User Profile is run after the Site Profile. SQL*Plus always searches the current directory for the User Profile. The environment variable SQLPATH may be set to a colon-separated list of directories that SQL*Plus will search in order.

For example, if the current directory is `/u02/oracle` and SQLPATH is set as follows:

```
% echo $SQLPATH
/home:/home/oracle:/u01/oracle
```

SQL*Plus will first look for `login.sql` in the current directory `/u02/oracle`. If it is not found there, SQL*Plus will then look in `/home`, `/home/oracle`, and `/u01/oracle`, respectively.

Here is a sample `login.sql` file:

```
set echo off
set feedback 4
set pause on
set pause "PLEASE PRESS RETURN TO CONTINUE"
set message on
set echo on
```

The PRODUCT_USER_PROFILE Table

The SQL script `$ORACLE_HOME/sqlplus/admin/pupbld.sql` may be run as the user SYSTEM to create the Product and User Profile tables.

`$ORACLE_HOME/sqlplus/admin/pupbld.sql` may also be run using the shell script `$ORACLE_HOME/bin/pupbld`. To use this script, the environment variables ORACLE_HOME and SYSTEM_PASS must be set. SYSTEM_PASS must be set to the SYSTEM's username and password. For example:

```
% setenv SYSTEM_PASS SYSTEM/manager
% pupbld

Installing product user profile tables...

Product user profile tables installed.
```

`pupbld.sql` will only be run by the Installer during SQL*Plus installation if Create Database Objects was selected.

Demonstration Tables

SQL*Plus is shipped with demonstration tables that may be used for testing.

Default Install

If using Default Install and Create Database Objects, the user SCOTT and the demonstration tables will be created automatically.

Custom Install

When installing SQL*Plus using Custom Install, if Create Database Objects is selected and you answer 'Yes' to the prompt "Would you like to load the SQL*Plus Demo Tables?", the Installer will create the user SCOTT with the password TIGER and create the demonstration tables.

Creating Demonstration Tables Manually

The SQL script `$ORACLE_HOME/sqlplus/demo/demobld.sql` is used to create the demonstration tables. The file `demobld.sql`, may be run in SQL*Plus as any user to create the demonstration tables in that schema. For example:

```
% sqlplus scott/tiger
SQL> @?/sqlplus/demo/demobld.sql
```

`$ORACLE_HOME/sqlplus/demo/demobld.sql` may also be run using the shell script `$ORACLE_HOME/bin/demobld` as follows:

```
% demobld scott tiger
```

Deleting Demonstration Tables

The SQL script `$ORACLE_HOME/sqlplus/demo/demodrop.sql` is used to drop the demonstration tables. The file `demodrop.sql` may be run in SQL*Plus as any user to drop the demonstration tables from that user's schema. For example:

```
% sqlplus scott/tiger
SQL> @?/sqlplus/demo/demodrop.sql
```

`$ORACLE_HOME/sqlplus/demo/demodrop.sql` may also be run using the shell script `$ORACLE_HOME/bin/demodrop` as follows:

```
% demodrop scott tiger
```

Note: Both SQL scripts `demobld.sql` and `demodrop.sql` drop the tables EMP, DEPT, BONUS, SALGRADE, and DUMMY. You must ensure that a table does not exist with the same name in the desired schema prior to running either script or the table data will be lost.

Help Facility

Default Install

If using Default Install and Create Database Objects, the Help Facility is installed automatically.

Custom Install

When installing SQL*Plus, if Create Database Objects is selected and you answer 'Yes' to the prompt "Would you like to load the SQL*Plus Help Facility?", the Installer will create the Help Facility.

Installing the Help Facility Manually

The Help Facility may be installed manually using the shell script `$ORACLE_HOME/bin/helpins`. To use this script, the environment variables ORACLE_HOME and SYSTEM_PASS must be set. SYSTEM_PASS must be set to SYSTEM's username and password. For example:

```
$ setenv SYSTEM_PASS SYSTEM/manager
$ helpins

SQL*Loader: Release 8.0.5.0.0 - Production

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 828

SQL*Loader: Release 8.0.5.0.0 - Production

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 1024
Commit point reached - logical record count 1207

SQL*Loader: Release 8.0.5.0.0 - Production

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 1024
Commit point reached - logical record count 1304
Commit point reached - logical record count 2328
Commit point reached - logical record count 2724
Commit point reached - logical record count 2835
```

See Also: Refer to the *SQL*Plus User's Guide and Reference*, and the `README` file, `$ORACLE_HOME/sqlplus/doc/release.doc`.

Using SQL*Plus

Using a System Editor from SQL*Plus

An `ed` or `edit` command entered at the SQL*Plus prompt calls a default operating system editor, such as `ed, emacs, ned`, or `vi`. Your PATH variable must include the directory of the editor.

The global default editor is usually set by the DBA in `glogin.sql` using the SQL*Plus `_editor` option. Override this setting by specifying an editor in `login.sql`. Both files are read by SQL*Plus at startup, the local file taking precedence. The `_editor` option can also be set during a SQL*Plus session, overriding the setting in either file.

If the `_editor` option is not set, the EDITOR and VISUAL environment variables specify the SQL*Plus editor. These variables are not set in `glogin.sql` or `login.sql`. They are set in a user startup file, or at the system prompt. If both are set, the EDITOR variable is used.

Setting the Order of the Editor

SQL*Plus searches for the default editor in this order:

1. The `_editor` variable during a SQL*Plus session.
2. The `_editor` variable in `login.sql`.
3. The `_editor` variable in `glogin.sql`.
4. The EDITOR environment variable.
5. The VISUAL environment variable.

When none of these values are set, SQL*Plus uses `ed`.

Setting the _editor option

Set the SQL*Plus `_editor` option by adding the following line to the `login.sql` file:

```
define _editor=editor_name
```

where *editor_name* is a LINUX editor.

Setting Environment Variables

For the Bourne or Korn shell, set the default editor with an environment variable by entering:

```
$ LINUX_VAR=editor_name; export LINUX_VAR
```

For the C shell, set the default editor with an environment variable by entering:

```
% setenv LINUX_VAR editor_name
```

Environment variable syntax is explained in Table 4–1.

Table 4–1 Syntax for LINUX Environment Variables

LINUX_VAR	the EDITOR or VISUAL environment variable
editor_name	the LINUX editor (for example, `vi` or `ed`)

Default Settings

If you call the system editor, the current SQL buffer is placed in the edit buffer and all statements available to the editor can change the SQL statement. SQL*Plus uses the `afiedt.buf` temporary file. When you exit the editor, the changed SQL buffer is returned to SQL*Plus.

Running Operating System Commands from SQL*Plus

An exclamation point (!) in the first position after the SQL*Plus prompt indicates subsequent character strings are passed to a sub-shell. The SHELL environment variable selects the shell you use to execute operating system commands. The default shell is `/bin/sh (sh)`. If the shell cannot be executed, an error message is displayed.

Use the following SQL*Plus commands to perform specific tasks:

- Enter [!]+[*command*] to execute one operating system command. After the command executes, control returns to SQL*Plus.
- Enter [!]+[Return] to execute *more than one* operating system command. When you finish, enter [Ctrl]+[d] to return to SQL*Plus.

Interrupting SQL*Plus

While running SQL*Plus:

- You can stop the scrolling record display and terminate a SQL statement by pressing [Ctrl]+[c] on BSD machines or [Delete] on System V machines.
- If you are at the SQL*Plus prompt, pressing [Interrupt] displays another SQL*Plus prompt.

Using the SPOOL Command

The default filename extension for files generated by the SPOOL command is `.lst`. To change the extension, specify a spool file containing a period (.).

For example:

```
SQL> SPOOL query.lis
```

Restrictions

COPY Command

The `COPY` command in SQL*Plus is supported without restrictions on different machines running the same version of the operating system.

`COPY` may also work between machines running different versions of the operating system. If `COPY` fails, test the connection using `rcp` or `ftp`. Restrictions in vendor-supplied networking software may prevent `rcp`, `ftp`, or `COPY` from functioning properly between systems.

> **Note:** The `rlogin` command does not send or receive large packets of data and is not an adequate test for connections.

If `COPY` does not function between systems, create a database link to the system and user ID indicating the table you want to copy. To do this, enter:

```
SQL> create table newtable as \
(SELECT * FROM table@database_link_name)
```

This selects the rows and columns from the original table on the remote system and enters them in the new table on the local system.

Resizing Windows

The default value for SQL*Plus LINESIZE is 80 and for PAGESIZE is 25. These variables do not automatically adjust for window size.

Return Codes

LINUX 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.

5

Using Oracle Precompilers and the Oracle Call Interface on Linux

- Overview of Oracle Precompilers
- Pro*C/C++
- Oracle Call Interface
- Oracle Precompiler and Oracle Call Interface Linking and Makefiles
- Thread Support
- Static and Dynamic Linking with Oracle Libraries
- Using Signal Handlers
- XA Functionality

Overview of Oracle Precompilers

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

Relinking Precompiler Executables

All precompiler executables are relinked using the makefile, `$ORACLE_HOME/precomp/lib/ins_precomp.mk`. The `make` command uses the following convention:

```
$ make -f ins_precomp.mk relink EXENAME=executable
```

This command will create the new executable in the `$ORACLE_HOME/precomp/lib` directory, and then move it to `$ORACLE_HOME/bin`. In order to create the new executable without it being moved to `$ORACLE_HOME/bin`, use the following command:

```
$ make -f ins_precomp.mk executable
```

where the name of the executable in respect of the product being used, can be determined from Table 5–1.

Table 5–1 Products and Their Corresponding Executable Names

Product	Executable
Pro*C/C++	`proc`
Object Type Translator	`ott`

For example, to relink the Pro*C/C++ executable, use the following command:

```
$ cd $ORACLE_HOME/precomp/lib
$ make -f ins_precomp.mk relink EXENAME=proc
```

Precompiler Configuration Files

There are six `.cfg` system configuration files in `$ORACLE_HOME/precomp/admin`. These are described in Table 5–2.

Table 5–2 System Configuration Files

Product	Configuration File
Pro*C/C++ v8.0.5	`pcscfg.cfg`
Object Type Translator v8.0.5	`ottcfg.cfg`

Issues Common to All Precompilers

Uppercase to Lowercase Conversion

In languages other than C, your compiler converts an uppercase function or subprogram name to lowercase. This can cause "No such user exit" errors. In this case, verify that the function or subprogram name in your option file matches the case in the `iapxtb` table.

Vendor Debugger Programs

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

Value of ireclen and oreclen

The `ireclen` and `oreclen` parameters do not have maximum values.

Supplemental Documentation

The following documents provide additional information about precompiler and interface features:

- *Programmer's Guide to the Pro*C/C++ Precompiler*
- *Programmer's Guide to the Oracle Call Interface*
- *Oracle8 Server Application Developer's Guide*

Pro*C/C++

For additional information regarding Pro*C/C++ release 8.0.5, see the README file, $ORACLE_HOME/precomp/doc/proc2/readme.doc.

Administering Pro*C/C++

System Configuration File

The system configuration file for Pro*C/C++ is $ORACLE_HOME/precomp/admin/pcscfg.cfg.

Using Pro*C/C++

Prior to using Pro*C/C++, verify that the correct version of the Operating System compiler is properly installed. The required version is documented in Chapter 1 of the *Oracle8 Installation Guide for LINUX*.

Demonstration Programs

Demonstration programs are provided to show the various functionality of the Pro*C/C++ precompiler. There are three types of demonstration programs: C, C++, and Object programs - the latter demonstrate the new Oracle8 Object features. All the demonstration programs are located in $ORACLE_HOME/precomp/demo/proc, and all of them assume that the demonstration tables created by $ORACLE_HOME/sqlplus/demo/demobld.sql exist in the SCOTT schema with the password TIGER.

For further information on building the demonstration programs using SQL*Plus, see “Demonstration Tables” on page 4-3 of this book. For further information on the demonstration programs see the *Programmer’s Guide to the Pro*C/C++ Precompiler.*

The makefile, $ORACLE_HOME/precomp/demo/proc/demo_proc.mk, should be used to create the demonstration programs. For example, to precompile, compile, and link the sample1 demonstration program, enter the following command.

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

Alternatively, the following command may be used, which achieves exactly the same result, only with more explicit syntax.

```
$ make -f demo_proc.mk build OBJS=sample1.o EXE=sample1
```

By default, all programs are dynamically linked with the client shared library, $ORACLE_HOME/lib/libclntsh.so.

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

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

To create all Pro*C/C++ C++ demonstration programs, enter this command:

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

To create all Pro*C/C++ Object demonstration programs, enter this command:

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

Some demonstration programs require a SQL script, found in `$ORACLE_HOME/precomp/demo/sql`, to be run. In order to build such a demonstration program and run the corresponding SQL script, the `make` macro argument, `RUNSQL=run`, must be included on the command line. For example, to create the `calldemo` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/calldemo.sql` script, use the following command syntax:

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

As another example, to create all Object demonstration programs and run all corresponding required SQL scripts, enter the following command:

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

The SQL scripts may also be run manually, if desired.

User Programs

The makefile, `$ORACLE_HOME/precomp/demo/proc/demo_proc.mk`, may be used to create user programs. The general syntax for linking a user program with `demo_proc.mk` is as follows:

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

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

For C source, dynamically linked with client shared library:

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

For C source, statically linked:

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

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

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

For C++ source, statically linked:

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

For LINUX issues on the use of shared libraries, refer to the LINUX documentation.

Oracle Call Interface

Using the Oracle Call Interface

Prior to using the Oracle Call interface (OCI), verify that the correct version of the compiler is properly installed. The required version for your operating system is specified in Chapter 1 of the *Oracle8 Installation Guide for LINUX*.

Demonstration Programs

Demonstration programs have been provided that show various functionality of the OCI. There are two types of demonstration programs: C and C++. All the demonstration programs are located in `$ORACLE_HOME/rdbms/demo`. Many of the demonstration programs assume that the demonstration tables created by `$ORACLE_HOME/sqlplus/demo/demobld.sql` exist in the SCOTT schema with the password TIGER.

For further information on building the demonstration programs using SQL*Plus, see "Demonstration Tables" on page 4-3 of this book.

See Also: For further information on the demonstration programs see the *Programmer's Guide to the Oracle Call Interface* and the program source for details of each program.

The makefile, `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk`, should be used to create the demonstration programs. For example, to compile and link the `cdemo1` demonstration program, enter the following command:

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

Alternatively, the following command may be used, which achieves exactly the same result, only with more explicit syntax:

```
$ make -f demo_rdbms.mk build OBJS=cdemo1.o EXE=cdemo1
```

By default, all programs are dynamically linked with the client shared library, `$ORACLE_HOME/lib/libclntsh.so`.

To create all OCI C demonstration programs, enter the following command:

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

To create all OCI C++ demonstration programs, enter this command:

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

Note: If you receive the following errors while linking a C++ program:

```
ld: fatal: library -lsunmath: not found
ld: fatal: library -lC: not found
ld: fatal: library -lC_mtstubs: not found
ld: fatal: library -lcx: not found
```

it will be necessary to include in LD_LIBRARY_PATH, the directory in which the specified libraries exist.

Some demonstration programs require a SQL script, located in `$ORACLE_HOME/rdbms/demo`, to be run manually prior to executing the program. In most cases, the SQL script name is the same as the program name with a `.sql` extension. For example, the SQL script for the program `oci02` is `oci02.sql`.

Read the comments at the beginning of the program to determine the required SQL script, if any.

User Programs

The makefile, `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk`, may be used to create user programs. The general syntax for linking a user program with `demo_rdbms.mk` is:

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

For example, to create the program `myprog` from the C source `myprog.c`, use one of the following commands depending on the type of executable desired:

For C source, dynamically linked with client shared library:

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

For C source, statically linked:

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

To create the program `myprog` from the C++ source `myprog.cc`

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

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

For C++ source, statically linked:

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

Oracle Precompiler and Oracle Call Interface Linking and Makefiles

Custom Makefiles

It is recommended that the provided `demo_`*product*`.mk` makefiles be used to link user programs as described in the specific product sections of this chapter. If it is necessary to modify the provided makefile, or if you decide to use a custom written makefile, the following should be noted:

- Do not modify the ordering of the Oracle libraries.
  Oracle libraries are included on the link line more than once so all symbols will be resolved during linking. There are two reasons for this:
  1. Oracle libraries are mutually referential, meaning that functions in library A call functions in library B, and functions in library B call functions in library A.
  2. The Solaris linker is a one-pass linker, meaning that the linker will search a library exactly once at the point it is encountered in the link line.
- If you add your own library to the link line, it should be added to the beginning or to the end of the link line.
  User libraries should not be placed between the Oracle libraries.
- Oracle library names and the contents of those libraries are subject to change between releases. Always use the `ddemo_`*product*`.mk` makefile that ships with the current release as a guide to determine which libraries are necessary.

Undefined Symbols

A common error when linking a program is undefined symbols, similar to the following:

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

This error occurs when the linker cannot find a definition for a referenced symbol. Generally, the remedy for this type of problem is to ensure that the library or object file containing the definition exists on the link line and that the linker is searching the correct directories for the file.

Oracle provides a utility called `symfind` to assist in locating a library or object file where a symbol is defined. Here is example output of `symfind` locating the symbol `sqlcex`:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
------------------------------------------------------
Command:          /u01/app/oracle/product/8.0.5/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/8.0.5
Output File:      (none)
Note:             I do not traverse symbolic links
                  Use '-v' option to show any symbolic links

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

Thread Support

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

Static and Dynamic Linking with Oracle Libraries

Precompiler and OCI applications can be linked with Oracle Libraries either statically or dynamically. With static linking, the libraries and objects of the whole application are

linked together into a single executable program. As a result, application executables can become fairly large.

With dynamic linking, the executing code partly resides in the executable program, and also resides in libraries that are linked by the application dynamically at run-time. Libraries that are linked at run-time are called dynamic or shared libraries. There are two primary benefits of dynamic linking:

1. **Smaller disk requirements:**
   Different applications, or different invocations of the same application, can use the same shared or dynamic library. As a result, the overall disk requirements are reduced.
2. **Smaller main memory requirements:**
   The same shared or dynamic library image (i.e., the in-memory copy), can be shared by different applications. This means that a library needs to be loaded only once into the main memory and then multiple applications can use the same library. As a result, main memory requirements are reduced.

Oracle Shared Library

The Oracle shared library is `$ORACLE_HOME/lib/libclntsh.so`. If the Oracle provided `demo_`*product*`.mk` makefile is used to link an application, the Oracle shared library is used by default.

It may be necessary to set the environment variable LD_LIBRARY_PATH so the runtime loader can find the Oracle shared library at process start-up. If you receive the following error when starting an executable, LD_LIBRARY_PATH must be set to the directory where the Oracle shared library exists:

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

Set LD_LIBRARY_PATH as follows:

```
% setenv LD_LIBRARY_PATH $ORACLE_HOME/lib
```

The Oracle shared library is created automatically during installation. If there is a need to recreate the Oracle shared library, exit **all** client applications using the Oracle shared library, including all Oracle client applications like SQL*Plus and Recovery Manager, and run the following command logged in as the *oracle* user:

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

Using Signal Handlers

This section describes signals Oracle8 uses for two-task communication, and explains how to set up your own signal handlers.

Signals

Signals are installed in a user process when you connect to the database, and are de-installed when you disconnect.

Oracle8 uses the following signals for two-task communications:

Table 5–3 Signals for Two-Task Communications

SIGCONT	used by the pipe two-task driver to send out-of-band breaks from the user process to the `oracle` process.
SIGINT	used by all two-task drivers to detect user interrupt requests. SIGINT is not caught by `oracle`; it is caught by the user process.
SIGPIPE	used by the pipe driver to detect end-of-file on the communications channel. When writing to the pipe, if no reading process exists, a SIGPIPE signal is sent to the writing process. SIGPIPE is caught by both the `oracle` process and the user process.
SIGCLD	used by the pipe driver. SIGCLD is similar to SIGPIPE, but only applies to user processes, not `oracle` processes. When an `oracle` process dies, the LINUX kernel sends a SIGCLD to the user process (**wait**() is used in the signal handler to see if the server process died). SIGCLD is not caught by `oracle`; it is caught by the user process.
SIGTERM	used by the pipe driver to signal interrupts from the user side to the `oracle` process. This occurs when the user presses the interrupt key [Ctrl]+[c]. SIGTERM is not caught by the user process; it is caught by `oracle`.
SIGIO	used by Oracle Net8 protocol adapters to indicate incoming networking events.
SIGURG	used by the Oracle Net8 TCP/IP drivers to send out-of-band breaks from the user process to the `oracle` process.

The listed signals affect Pro*C or other precompiler applications. You can install one signal handler for `SIGCLD` (or `SIGCHLD`) and `SIGPIPE` when connected to the oracle process. You can have multiple signal handlers for `SIGINT` as long as the `osnsui()` routine is called to set this up. You can install as many signal handlers as you want for other signals. If you are not connected to the oracle process**,** you can have multiple signal handlers.

Sample Signal Routine

The following example shows how you can set up your own signal routine and the catching routine. For `SIGINT`, use `osnsui()` and `osncui()` to register and delete signal-catching routines.

```
/* user side interrupt set */
word osnsui( /*_ word *handlp, void (*astp), char * ctx, _*/)
/*
** osnsui: Operating System dependent Network Set
**User-side
** Interrupt. Add an interrupt handling procedure
**astp.
```

```
** Whenever a user interrupt(such as a ^C) occurs,
**call astp
** with argument ctx. Put in *handlp handle for this
**handler so that it may be cleared with osncui.
** Note that there may be many handlers; each should
** be cleared using osncui. An error code is
**returned if an error occurs.
*/

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

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

```
/*
** My own user interrupt handler.
*/
void sig_handler()
{
...
}

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

   int handle, err;
   ...

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

```
...
/* clear my interrupt handler */
if (err = osncui(handle))
{
/* if the return value is non-zero, an error has occurred
Do something appropriate here. */
...
}
...
}
```

XA Functionality

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

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

6

Configuring Oracle Net8

- Supplemental Documentation
- Core Net8 Products and Features
- Oracle Net8 Protocol Adapters
- The BEQ Protocol Adapter
- The IPC Protocol Adapter
- The TCP/IP Protocol Adapter
- Net8 Naming Adapters
- Oracle Enterprise Manager (OEM) Intelligent Agent

Supplemental Documentation

See Also: The following documents provide a full discussion of Oracle Net8 features:

- *Oracle Net8 Administrator's Guide*
- Oracle Networking Quick Reference Card for Net8
- *Oracle Security Server Guide*
- *Oracle Cryptographic Toolkit Programmer's Guide*

Supplementary Information in README Files

Table 6–1 shows the location of README files for various bundled products. The README files describe changes since the last release.

Table 6–1 Location of README Files for Oracle Products

Product	README File
Net8	`$ORACLE_HOME/network/doc/README.Net8`
Oracle Intelligent Agent	`$ORACLE_HOME/network/doc/README.oemagent`
Oracle Security Server	`$ORACLE_HOME/network/doc/README.OSS`
Oracle Names Server	`$ORACLE_HOME/network/install/names/doc/README.doc`
Oracle TCP/IP Protocol Adapter	`$ORACLE_HOME/network/install/tcppa/doc/README.doc`

Core Net8 Products and Features

See Also: Sample files can be found in the *Oracle Net8 Administrator's Guide*.

Net8 Files and Utilities

Location of Net8 Configuration Files

The default directory for global Oracle Net8 and Connection Manager files is `/var/opt/oracle` on LINUX.

Oracle Net8 and Connection Manager search for global files in the following order:

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

If your files are not in the default directory, use the TNS_ADMIN environment variable in the startup files of all network users to specify a different location:

For the C shell, enter:

```
% setenv TNS_ADMIN new_default
```

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

Sample Configuration Files

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

The adapters Utility

To display installed Oracle Net8 adapters, enter:

```
% adapters
```

To display adapters linked with a specific executable, enter:

```
% adapters executable
```

For example, the following command displays the adapters linked with the `oracle` executable:

```
% adapters oracle
Protocol Adapters linked with oracle are:
    BEQ Protocol Adapter
    IPC Protocol Adapter
    TCP/IP Protocol Adapter
Net8 Naming Adapters linked with oracle are:
    Oracle TNS Naming Adapter
    Oracle Naming Adapter
Advanced Networking Option/Network Security products linked with oracle are:
    Oracle Security Server Authentication Adapter
```

Oracle Connection Manager

For information on the Oracle Connection Manager see the *Net8 Administrator's Guide*.

Multi-Threaded Server

For information on the Multi-Threaded Server see the *Oracle8 Server Concepts* and *Oracle8 Administrator's Guide.*

Oracle Names

For information on Oracle Names see the *Oracle Net8 Administrator's Guide*.

Oracle Net8 Protocol Adapters

The supported Protocol Adapters for Net8 version 8.0.5 on LINUX are BEQ Protocol Adapter, IPC Protocol Adapter, TCP/IP Protocol Adapter.

Prior to installing the TCP/IP Net8 Protocol Adapters, the appropriate operating system software must be installed and configured. Refer to the *Oracle8 Installation Guide for LINUX* for requirements details. The BEQ and IPC Net8 Protocol Adapters do not have any specific operating system requirement.

The IPC, TCP/IP Net8 Protocol Adapters each have a protocol-specific ADDRESS specification that is used for Net8 configuration files and for the MTS_LISTENER_ADDRESS database initialization parameter (`init.ora`). See the ADDRESS specification heading under each Protocol Adapter section in this chapter for details.

Table 6–2 shows a summary of ADDRESS specifications for each Protocol Adapter.

Table 6–2 ADDRESS Specification Summary

Protocol Adapter	ADDRESS Specification
BEQ	(ADDRESS = (PROTOCOL = BEQ) (PROGRAM = ORACLE_HOME/bin/oracle) (ARGV0 = oracleORACLE_SID) (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))') (ENVS = 'ORACLE_HOME=ORACLE_HOME,ORACLE_SID=ORACLE_SID'))
IPC	(ADDRESS = (PROTOCOL=IPC) (KEY=*key*))
TCP/IP	(ADDRESS = (PROTOCOL=TCP) (HOST=*hostname*) (PORT=*port_id*))

The BEQ Protocol Adapter

Overview of the BEQ Protocol Adapter

The BEQ Protocol Adapter, is both a communications mechanism and a process spawning mechanism. If a service name is not specified, either directly by the user on the command line or the login screen, or indirectly through an environment variable such as TWO_TASK, then the BEQ Protocol Adapter will be used. In which case, a dedicated server will always be used, and the multi-threaded server will never be used. This dedicated server is started automatically by the BEQ Protocol Adapter, which waits for the server process to start and attach to an existing SGA. If the startup of the server process is successful, the BEQ Protocol Adapter then provides inter-process communication via UNIX pipes.

An important feature of the BEQ Protocol Adapter is that no network Listener is required for its operation, since the adapter is linked into the client tools and directly starts its own server process with no outside interaction. However, the BEQ Protocol Adapter can only be used when the client program and the Oracle8 server reside on the same machine. The BEQ Protocol Adapter is always installed, and always linked in to all client tools and to the Oracle8 server.

Specifying a BEQ ADDRESS

The BEQ Protocol Adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the parameters in any order.

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

Syntax for BEQ Protocol Adapter connection parameters is described in Table 6–3.

Table 6–3 Syntax for BEQ Protocol Adapter Connection Parameters

PROTOCOL	Specifies the adapter to be used. The value is `beq` and may be specified in either uppercase or lowercase.
PROGRAM	The full path to the oracle executable.
ARGV0	The name of the process as it appears in a `ps` listing. The recommended value is oracleORACLE_SID.
ARGS	`'(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))'`
ENVS	Environment specification where ORACLE_HOME is the full path to the ORACLE_HOME directory of the database to connect, and ORACLE_SID is the system identifier of the database to connect.

Example 6–1 BEQ ADDRESS Specifying a Client

The following is an example of an BEQ ADDRESS:

```
(ADDRESS =
    (PROTOCOL = BEQ)
    (PROGRAM = /u01/app/oracle/product/8.0.5/bin/oracle)
    (ARGV0 = oracleV805)
    (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    (ENVS = 'ORACLE_HOME=/u01/app/oracle/product/8.0.5,ORACLE_SID=V805')
)
```

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

The IPC Protocol Adapter

Overview of the IPC Protocol Adapter

The IPC Protocol Adapter, is similar to the BEQ Protocol Adapter in that it can only be used when the client program and the Oracle8 server reside on the same machine. The IPC Protocol Adapter differs from the BEQ Protocol Adapter in that it can be used with dedicated server and multi-threaded server configurations. The IPC Protocol Adapter requires a network listener for its operation. The IPC Protocol Adapter is always installed, and always linked in to all client tools and to the Oracle8 server.

For the IPC Protocol Adapter, the location of the UNIX Domain Socket (IPC) file on LINUX systems changed after Oracle7 7.1. Thus, if you have Oracle7 7.1 installed on the same machine as Oracle8, and you attempt to make an IPC connection between the two instances, the connection may fail. The solution to this problem is to make a symbolic link between the directory where the IPC file used to be (`/var/tmp/o`) and where it now resides (`/var/tmp/.oracle`).

Specifying an IPC ADDRESS

The IPC Protocol Adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the parameters in any order.

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

Syntax for IPC Protocol Adapter connection parameters is described in Table 6–4.

Table 6–4 Syntax for IPC Protocol Adapter Connection Parameters

PROTOCOL	Specifies the adapter to be used. The value is `ipc` and may be specified in either uppercase or lowercase.
KEY	Service name of database or database identifier (SID).

Example 6–2 IPC ADDRESS Specifying a Client

The following is an example of an IPC ADDRESS:

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

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

The TCP/IP Protocol Adapter

Overview of the TCP/IP Protocol Adapter

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

```
listener     1521/tcp
```

where *listener* is the name of the listener, as defined in `listener.ora`.

Reserve more than one port to start more than one listener.

Specifying a TCP/IP ADDRESS

The TCP/IP Protocol Adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the three parameters in any order.

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

Syntax for TCP/IP Protocol Adapter connection parameters is described in Table 6–5.

Table 6–5 Syntax for TCP/IP Protocol Adapter Connection Parameters

PROTOCOL	Specifies the adapter to be used. The value can be uppercase or lowercase. The default is `tcp`.
HOST	The host name or the host IP address.
PORT	The TCP/IP port. Either a number or the name specified in the `/etc/services` file. Oracle Corporation recommends a value of 1521.

Example 6–3 TCP/IP ADDRESS Specifying a Client

Following is an example of the TCP/IP ADDRESS specifying a client on the MADRID host:

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

The last field could be specified by name, for example, (PORT=listener)**.** The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

Net8 Naming Adapters

NIS Naming Adapter

For details on configuring the NIS Naming Adapter, see the *Oracle Net8 Administrator's Guide*, Release 8.0.

Oracle Enterprise Manager (OEM) Intelligent Agent

Agent Service Discovery and Auto-Configuration

See Also: The *Oracle Enterprise Manager Configuration Guide*.

Debugging Tcl Scripts

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

See Also: The *Oracle Enterprise Manager Application Developer's Guide* for additional details.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X