Oracle8i™

Administrator's Reference for Intel-Linux

Release 8.1.5

July 1999

Part No. A74957-01

Topics Include:

Oracle8*i* Administrator's Reference, Release 8.1.5 for Intel-Linux

Part No. A74957-01

Release 8.1.5

Primary Author: Donn Fisher

Technical Contributors: Jamshed Patel, Sanjib Das, Ravi Kuntamukkla, Sandy Lee, and Mangesh Nijasure.

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the programs.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these programs, no part of these programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

Contents

2 Tuning Oracle8*i*

3 Administering SQL*Plus

4 Using Oracle Precompilers and the Oracle Call Interface

5 Configuring Net8

6 Running Oracle Data Option Demos

A Optimal Flexible Architecture

Index

Send Us Your Comments

Oracle8*i* Administrator's Reference, Release 8.1.5 for Intel-Linux

Part No. A74957-01

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:

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

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

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

Preface

Purpose

This reference provides Intel-Linux-specific information required to successfully administer and tune Oracle8*i*. This reference supplements product information provided in your Oracle8*i* Documentation Library Set.

Audience

This document is intended for anyone responsible for installing Oracle8*i* on a Intel-Linux system.

Oracle8*i* and Oracle8*i* Enterprise Edition

Unless noted otherwise, features and functionality described in this document are common to both Oracle8*i* and Oracle8*i* Enterprise Edition.

Typographic Conventions

`monospace` Monospace type indicates UNIX commands, directory names, usernames, pathnames, 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. It is also used for emphasis.

UPPERCASE Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment variables.

Because UNIX is case-sensitive, conventions in this document may differ from those used in other 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]`*

Related Documentation

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

- *Oracle8i Administrator's Guide*. Use this as a starting point for tasks associated with Oracle8*i*, such as database creation, managing database objects, and creating users.
- *Net8 Administrator's Guide*
- Oracle8*i Tuning*

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

Oracle Services and Support

A wide range of information about Oracle products and global services is available on the Internet, from *http://www.oracle.com*. The following sections provide URLs for selected services:

Oracle Support Services

Support contact information worldwide is listed at *http://www.oracle.com/support*. Templates are provided to help you prepare information about your problem before you call. You will also need your CSI number (if applicable) or complete contact details, including any special project information.

Products and Documentation

Oracle Store, for U.S.A. customers, is at *http://oraclestore.oracle.com*. Links to Stores in other countries are provided from this site.

Education and Training

Training information and worldwide schedules are available from *http://education.oracle.com*.

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at *http://technet.oracle.com*. OTN delivers technical papers, code samples, product documentation, self-service developer support, and key developer products to enable rapid development and deployment of applications built on Oracle technology.

1

Administering Oracle8*i*

This chapter provides the following Oracle8*i* administration information:

Setting the Environment

This section describes how to establish a common environment for your Oracle8*i* system.

Displaying Environment Variables

To display the current 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*i* 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 Oracle8*i* 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 Oracle8*i* databases

For example, you need to move the database from `/usr/oracle` to `/usr1/oracle`. Without a common environment-setting routine, you need to update user startup files individually. With `oraenv`, each user profile calls the `oraenv` command file.

Local bin Directory

Placing `oraenv` (or `coraenv`) and `dbhome` in a 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. The local `bin` directory is specified by the `root.sh` script, which is run following installation. The default location for the local `bin` directory on Linux is `/usr/local/bin`.

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 environment variable at startup, set the ORAENV_ASK environment variable to `no`.

In the following examples, ORAENV_ASK is reset to the default, `Yes`, after `oraenv` is executed. This ensures that the system prompts you 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
setenv 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}’ /var/opt/oracle/oratab
echo "SIDS on this machine are $SIDLIST"
ORAENV_ASK=
./usr/local/bin/oraenv
```

For the C shell:

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

Environment Variables for Oracle8*i*

This section describes the most commonly used Oracle8*i* and Linux environment variables. Some of these variables must be defined before you install Oracle8*i*.

They are described in the *Oracle8i Installation Guide for Intel-Linux*.

Oracle Environment Variables on Linux

Table 1–1 provides the syntax and examples for Oracle8*i* variables.

Table 1–1 Oracle8i Environment Variables on Linux

Variable	Detail	Definition
EPC_DISABLED	Function	Disables Oracle Trace
	Syntax	`true` or `false`
NLS_LANG	Function	Specifies the language and character set used for output. See the *Oracle8i Installation Guide for Intel-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.
	Example	`$ORACLE_HOME/ocommon/nls/admin/data`
ORACLE_BASE	Function	Specifies the BASE of the Oracle directory structure for OFA-compliant databases.
	Syntax	*directory_path*
	Example	/*mount_point*/`app/oracle`
ORACLE_HOME	Function	Specifies the directory containing the Oracle software.
	Syntax	*directory_path*
	Example	`$ORACLE_BASE/product/8.1.5`
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/8.1.5/bin:` **Note:** The period adds the current working directory to the search path.

Table 1–1 Oracle8i Environment Variables on Linux

Variable	Detail	Definition
ORACLE_SID	Function	Specifies the Oracle System Identifier.
	Syntax	The string of numbers and characters must begin with a letter. A maximum of four characters is recommended. For more information, see the *Oracle8i Installation Guide for Intel-Linux*.
	Example	SAL1
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`, then `(c)oraenv` does not prompt and, if set to anything else, it does prompt.
	Syntax	string
	Range of Values	`NO` or anything else.
TNS_ADMIN	Function	Sets the directory containing the Net8 configuration files.
	Syntax	*directory_path*
	Range of Values	Any directory; for more information, see the *Oracle8i Installation Guide for Intel-Linux.*
	Example	`$ORACLE_HOME/network/admin`
TWO_TASK	Function	Sets the default Net8 connect string descriptor alias defined in the `tnsnames.ora` file.
	Syntax	Available network alias.
	Range of Values	Any valid Net8 alias defined in the `tnsnames.ora` file.
	Example	PRODDB_TCP

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

Abbreviations for ORACLE_HOME and ORACLE_SID

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

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

The at sign (@) sign represents `$ORACLE_SID`. For example, to indicate a file belonging to the current instance, enter:

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

Linux Environment Variables Used with Oracle8*i*

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

Table 1–2 Linux Environment Variables Used with Oracle8i

Variable	Detail	Definition
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. See your X Windows documentation for more information.
	Syntax	*hostname:display* The *hostname* is your machine name (either IP address or alias); *display* is the monitor number - if you have single monitor, the number is 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 the *Oracle8i Installation Guide for Intel-Linux*.
LDOPTS	Function	Specifies the default linker options on some platforms. See `man` pages on `ld` for details.
PRINTER	Function	Specifies the user’s default printer for Linux systems.
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:$ORACLE_HOME/lib`
PATH	Function	Used by the shell to locate executable programs; needs to include `$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

Table 1–2 Linux Environment Variables Used with Oracle8i

Variable	Detail	Definition
PRINTER	Function	Selects the 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 and other Linux tools to determine terminal types.
	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 you to adjust the clock for daylight saving time changes or different time zones. The adjusted time is used to time-stamp files, produce the output of the `date` command, and obtain the current SYSDATE.

WARNING: You are discouraged from changing your 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.

Initialization Parameters

Initialization parameters allow you to configure and tune your system. This section describes:

- customizing initialization parameters in the `init`*sid*`.ora` file for the Oracle8*i* instance
- pre-set default initialization parameters

There are many optional initialization parameters described in the generic Oracle8*i* documentation.

See Also: *Oracle8i Administrator's Guide* and *Oracle8i Tuning*.

Customizing the init*sid*.ora File

This section documents the default `init`*sid*`.ora` file provided with the Oracle8*i* software. The Oracle Universal Installer (OUI) creates it in the `$ORACLE_BASE/admin/`*sid*`/pfile` directory. You can modify it to customize your Oracle8*i* 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 1–3 suggests the approximate SGA sizes corresponding to the three scenarios provided in the `init`*sid*`.ora` file.

Table 1–3 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 your Oracle8*i* installation.

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

db_files = 80                                                         # SMALL
# db_files = 400                                                      # MEDIUM
# db_files = 1500                                                     # 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 = 32768                                                    # SMALL
# log_buffer = 32768                                                  # MEDIUM
# log_buffer = 163840                                                 # LARGE

# audit_trail = true            # if you want auditing
# timed_statistics = true       # if you want timed statistics
max_dump_file_size = 1040       # limit trace file size to 5 Meg each

# Uncommenting the lines 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 40/5 = 4:
# transactions = 40
# transactions_per_rollback_segment = 5

# 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

# Uncomment the following line if you want to use some of the new 8.1
# features. Please remember that using them may require some downgrade
# actions if you later decide to move back to 8.0.

# compatible = 8.1.0
```

Default Initialization Parameter Values

Table 1–4 lists default initialization parameter values on Linux. All Oracle8*i* instances assume these values if you do not specify different values for them in the `init`*sid*`.ora` file. Oracle Corporation recommends that you include in the `init`*sid*`.ora` file only those parameters that differ from the default initialization parameter values.

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

See Also: *Oracle8i Server Reference.*

Table 1–4 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
LOCK_SGA	FALSE

Table 1–4 Default Initialization Parameters

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

Database Limits

Table 1–5 lists the default and maximum values for parameters in a CREATE DATABASE or CREATE CONTROL FILE statement.

> **NOTE:** Interdependencies among these parameters may affect allowable values.

Table 1–5 Create Control File Parameters

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

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 1–6.

Table 1–6 Linux Accounts

Account	Description
oracle	The *oracle* software owner represents the account that owns the Oracle8*i* 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.

Oracle Server accounts are described in Table 1–7.

Table 1–7 Oracle Server Accounts

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

Special group accounts are described in Table 1–8.

Table 1–8 Special Group Accounts

Account	Description
`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 SQL*Plus 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 SQL*Plus. 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*i* 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*i* improves security by dividing work and address space between the user program and the `oracle` program. All database access is achieved through the shadow process and special authorizations on the `oracle` program.

> **See Also:** Security issues are dealt with extensively in the *Oracle 8i Administrator's Reference*, "The Oracle Database Administrator" chapter.

Groups and Security

To ensure greater security for an Oracle8*i* 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 Oracle8*i*, 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`.

- Create a group named oinstall. The oinstall group will own the OUI oraInventory, and is responsible for installing and upgrading the Oracle8*i* system. All oracle accounts must belong to this group.
- Add another group of users to allow a subset of Linux users limited access to Oracle8*i*. Publicly executable programs and Oracle utilities, such as SQL*Plus, should be executable by this group. Set the permissions on the utilities to 751 to grant execute permissions to this group, but not other. If you create such a user group, members of the dba group must also be included in it.
- 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 have multiple databases with the same ORACLE_HOME (a configuration which Oracle Corporation strongly discourages), they should have the same database administrators' group.

> **Note:** 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 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*i* should own the database files. The default user ID is the *oracle* software owner. Set the authorizations on these files to 0600: read and 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.

The Oracle Universal Installer automatically sets the user ID of the `oracle` executable to:

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

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.

If you need to set this manually, enter:

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

Security and Remote Passwords

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

Local password files are 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=filename password=password entries=max_users
```

This syntax is described in Table 1–9:

Table 1–9 Syntax for Executing orapwd

Syntax	Description
filename	the name of the file where password information is written. The name of the file must be `orapw`*sid*, and you must supply the full pathname. Its contents are encrypted and not user-readable. This parameter is mandatory.
password	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.
max_users	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 will need to increase the maximum number of users in the future. Therefore, set *max_users* to a higher number than you expect to require.

orapwd Example

```
$ orapwd file=/u01/app/oracle/product/8.1.5/dbs/orapwV815 \
 password=rsdb3t4 entries=30
```

See Also: *Oracle8i Administrator's Guide.*

Access to a Database from a Remote PC

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

```
SQL> connect internal/dba_password@alias as {sysdba|sysoper}
```

Remote Authentication

The following `init`*sid*`.ora` parameters, shown in Table 1–10, control the behavior of remote connections through non-secure protocols:

Table 1–10 Parameters for Controlling Remote Connections

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

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 `/var/opt/oracle/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 1–1 Using lhd and grpx Scripts

This example shows how the administrator can propagate the basic framework for a `.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
```

Estimating Oracle8*i* Memory Usage

Estimating the memory usage requirements of Oracle8*i* helps to determine the number of users that you can have on your system. The estimates also help determine the physical memory and swap space requirements.

Before starting Oracle8*i*, estimate the memory requirements 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.

To determine the SGA size, see "Calculating the Size of the SGA" on page 1-26.

For each client-server 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 the text size of an executable, 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.

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

To compute actual Oracle physical memory usage while the database is up and users are connecting to it, use the `ps -aux` command. Look for all the front end, server, and background Oracle process entries. For each entry, total the VSZ columns. This is reported in kB.

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.

Server Resource Limits

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

Disk quotas established for the *oracle* user ID may hinder the operation of the Oracle8*i* system. Confer with your Oracle8*i* database administrator and the Linux system manager before establishing disk quotas.

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. For example, if SHMMAX is 512 KB and the SGA is 2048 KB, the SGA requires four segments.

If the size of the SGA exceeds the maximum size of a shared memory segment (SHMMAX), Oracle8*i* 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`*`sid`*`.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 physical memory of the machine is devoted to shared memory, resulting in poor performance.

Calculating the Size of the SGA

You can determine the SGA size in the following ways:

- The approximate size of an SGA per instance can be calculated with this formula:

  (DB_BLOCK_BUFFERS × DB_BLOCK_SIZE)
  + SORT_AREA_SIZE
  + SHARED_POOL_SIZE
  + LOG_BUFFER

- To display the size of the SGA for a running database, in bytes, use the SQL*Plus `show sga` command.
- You can also find the size of the SGA when you start your database system. The SGA size is displayed next to the heading Total System Global Area.

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. Use the `tstshm` executable included in this release of Oracle8*i*:

   ```
   $ tstshm
   ```

   Note: The system may experience problems when executing `tstshm` while using Intimate Shared Memory (ISM). To turn ISM off, add this line to the `init`*sid*`.ora` file:

   ```
   use_ism = false
   ```

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

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

3. 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.

4. Shut down the existing Oracle database.

5. 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` completes the following:

   - 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*.

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

Building and Running Demonstrations

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. Run the `ulcase`*n*`.sql` script corresponding to the demonstration you want to run. As `scott/tiger`, invoke SQL*Plus from the command line:

   ```
   $ sqlplus scott/tiger ulcasen.sql
   ```

2. As scott/tiger, invoke the demonstration from the command line:

   ```
   $ sqlldr scott/tiger ulcasen
   ```

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

- ulcase1: Follow steps 1 - 2.
- ulcase2: Follow step 2 to invoke the demonstration (you do not have to run the ulcase2.sql script).
- ulcase3: Follow steps 1 - 2.
- ulcase4: Follow steps 1 - 2.
- ulcase5: Follow steps 1 - 2.
- ulcase6: Run the ulcase6.sql script as scott/tiger, then enter the following at the command line:

  ```
  $ sqlldr scott/tiger ulcase6 DIRECT=true
  ```

- ulcase7: Run the ulcase7s.sql script as scott/tiger, then enter the following at the command line:

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

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

Administering SQL*Loader

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

File Processing Option

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

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

Table 1–11 File Processing Option Strings

Option String	Description
str	Specifies a stream of records, each terminated by a newline character, which are read in one record at a time (default).

Table 1–11 File Processing Option Strings

Option String	Description
`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.

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*i* database open and mounted:

1. Invoke SQL*Plus and connect with the user/password `scott/tiger`:

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

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

   ```
   SQL > @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 1–12 lists the kernel demonstrations.

Table 1–12 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 1–13 lists the precompiler demonstrations.

Table 1–13 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
SQL> @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
```

To run the `extproc` demo, first 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 SQL*Plus session, enter:

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

Finally, to run the tests:

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

Relinking Network Executables

You can manually relink your network executables with a relink shell script. Relinking is necessary after an operating system upgrade, or when the error message "relink network executables" is displayed.

The relink script performs manual relinking of Oracle product executables based on what has been installed in the ORACLE_HOME.

To relink, enter the following:

```
$ relink parameter
```

Table 1–14 Relink Script Parameters

Parameter	Value
`all`	everything which has been installed
`oracle`	Oracle database executable only
`network`	`net_client`, `net_server`, `nau`, `cman`, `cnames`
`client`	`net_client`, `otrace`, `plsql`, `client_sharedlib`
`interMedia`	`ctx`, `ordimg`, `ordaud`, `ordvir`, `md`
`precomp`	all precompilers which have been installed
`utilities`	utilities
`oemagent`	`oemagent`, `odg`

2

Tuning Oracle8*i*

This chapter provides the following tuning information:

The Importance of Tuning

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

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

> **See Also:** *Oracle8i Parallel Server Concepts and Administration* and *Oracle8i 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, paging, block I/O, traps, and CPU activity on Linux, depending on the switches you supply with the command. The following command displays a summary of system activity eight times, at five second intervals:

```
% vmstat 5 8
```

Sample output from the `vmstat` command is shown in Figure 2–1.

Figure 2–1 Output from vmstat 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  6548 234812  17696 482620   0    0      2     0    29     3    0    0  10
0 0 0  6548 234812  17696 482620   0    0      0     0   106     8    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     1   107     7    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     0   113     7    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     0   123     7    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     0   121     7    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     0   123     6    0    0 100
0 0 0  6548 234812  17696 482620   0    0      0     0   124     8    0    0 100
```

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

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

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 *Oracle8i Tuning*. On Linux, the scripts are located in the `$ORACLE_HOME/rdbms/admin/` directory.

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 `swapon -s` or `vmstat` 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

On Linux use `swapon -s` to determine how much swap space is currently in use. Use `swapon -a` to add swap space to your system. Consult your Linux documentation for further information

Start with swap space two to four times the amount of random access memory (RAM) of your system. Use a higher value if you plan to use Developer/2000, Oracle Applications, or Oracle InterOffice. Monitor the use of swap space and increase it as necessary.

Controlling Paging

Paging may not present as serious a problem as swapping, because an entire program does 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 time to compare against measurements from periods of slow response.

Use `vmstat` to monitor paging.

- added to the free list by page-out activity. This value should be zero.

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

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

Hold the SGA in a Single Shared Memory Segment

You will not be able to start the database without sufficient shared memory. You can reconfigure the Linux kernel to increase shared memory.

> **See Also:** For more information, see "Controlling the System Global Area" in Chapter 1 and "Configure Linux Kernel for Oracle" in Chapter 2 of the *Oracle8i Installation Guide for Intel-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.

Tuning the Database Writer to Increase Write Bandwidth

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

- use I/O slaves

I/O Slaves

I/O Slaves are specialized processes whose only function is to perform I/O. They replace the Oracle7 feature, Multiple DBWRs. They are a generalization of Multiple DBWRs and can be deployed by other processes as well and they can operate whether or not asynchronous I/O is available. They are allocated from LARGE_POOL_SIZE if set, otherwise they are allocated from shared memory buffers. I/O Slaves come with a new set of initialization parameters that allow a degree of control over the way they operate, shown in Table 2–1.

Table 2–1 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 2–1, DISK_ASYNCH_IO and TAPE_ASYNCH_IO, allow asynchronous I/O to be switched off respectively for disk and tape devices. Because the number of I/O Slaves for each process type defaults to zero, no I/O Slaves will be deployed unless specifically set.

DBWR_IO_SLAVES should only be set to a value greater than zero (0) 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 is 4. In the case of LGWR_IO_SLAVES, you should not deploy more than 9 slaves.

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

Choosing the Appropriate File System Type

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

- `ext:  the Extended File System for Linux`
- `ext2: the second Extended File System for Linux`
- `ufs`: the UNIX File System, derived from BSD UNIX
- `sysv: the UNIX system V/Coherent file system`

The suitability of a file system to an application is usually undocumented. Although `ext2` is often the high-performance choice, performance differences vary, depending on the file system chosen.

Monitoring Disk Performance

Use vmstat to monitor the block IO activity. this gives blocks sent and received from a block device.

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

This section describes CPU tuning issues.

Keeping All Oracle Users and Processes at the Same Priority

Oracle is designed to operate with all users and background processes operating at the same priority level. Changing priorities 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.

Using Processor Affinity and Binding on Multi-Processor Systems

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

Using 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*i* (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_utilities.mk singletask
```

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

Tuning Oracle Resource Contention

Tuning 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`.

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.

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

Tuning Block Size and File Size

> **Note:** 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 2 KB and the maximum block size is 16 KB. You can set the actual block size to any multiple of 2 KB up to 16 KB, 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 before creating the database:

```
db_block_size=new_block_size
```

Using Trace and Alert Files

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

Trace File Names

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

Table 2–2 Format Key to Process Name

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

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

Alert Files

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

3

Administering SQL*Plus

This chapter provides the following SQL*Plus administration information:

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

Administering SQL*Plus

This section provides SQL*Plus administration information.

Setup Files

The setup files for SQL*Plus are `glogin.sql`, the global setup file that defines the Site Profile, and `login.sql`, which defines the User Profile. The `glogin.sql` and `login.sql` files contain 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 `$ORACLE_HOME/sqlplus/admin/glogin.sql`. SQL*Plus runs this command file when any user starts SQL*Plus. The default Site Profile is placed in the `$ORACLE_HOME/sqlplus/admin` directory when SQL*Plus is installed. If a Site Profile already exists, it will be overwritten. An existing Site Profile is deleted when SQL*Plus is de-installed.

The User Profile

The User Profile file is `login.sql`. SQL*Plus runs this command file when any user starts SQL*Plus. SQL*Plus always searches the current directory first for the User Profile. The environment variable SQLPATH may be set to a colon-separated list of directories that SQL*Plus will search in the order listed.

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

```
/home:/home/oracle:/u01/oracle
```

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

Since `login.sql` is run last, options set in `login.sql` override those set in `glogin.sql`.

See Also: Chapter 3 in the *SQL*Plus User's Guide and Reference*.

The PRODUCT_USER_PROFILE Table

During a [Typical] installation, the PRODUCT_USER_PROFILE table (PUP) is created automatically. The PUP table is used to disable certain SQL and SQL*Plus commands. If you need to recreate this table, run the $ORACLE_HOME/sqlplus/admin/pupbld.sql script in the SYSTEM schema.

For example:

```
% sqlplus system/manager
SQL> @?/sqlplus/admin/pupbld.sql
```

SQL*plus will use the value of $ORACLE_HOME wherever a question mark (?) appears.

Demonstration Tables

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

Typical Install

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

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
```

The $ORACLE_HOME/sqlplus/demo/demobld.sql script 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
```

The $ORACLE_HOME/sqlplus/demo/demodrop.sql script 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 before running either script, or the table data will be lost.

Help Facility

Typical Install

When you copy a starter database with pre-built datafiles as part of the [Typical] installation or as an option in the Database Configuration Assistant, the Help Facility is installed automatically.

Database Configuration Assistant (DBCA)

DBCA gives you the option to create help tables when creating a database.

Installing the Help Facility Manually

The Help Facility may be installed manually using the shell script $ORACLE_HOME/bin/helpins. Before you run the script, the SYSTEM_PASS environment variable must be set to the SYSTEM schema name and password. For example:

```
% setenv SYSTEM_PASS SYSTEM/MANAGER
% helpins
```

See Also: Refer to the *SQL*Plus User's Guide and Reference.*

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.

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

You can specify which editor will be invoked by defining the SQL*Plus _editor variable. This variable can be set in glogin.sql, login.sql, or entered during a SQL*Plus session.

For example, to set the default editor to vi, enter:

```
define_editor=vi
```

If you do not set the _editor variable, then the value of either the EDITOR or VISUAL environment variables is used. If both are set, the EDITOR variable value is used.

When _editor, EDITOR and VISUAL are not specified, the default editor is ed.

When you invoke the editor, SQL*Plus uses a temporary file called afiedt.buf to pass text to the editor. You can rename this file, using the SET EDITFILE command. For example:

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

SQL*Plus does not delete the temporary file.

Running Operating System Commands from SQL*Plus

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

You can perform operating system commands without leaving SQL*Plus by entering the HOST or (!) commands.

For example, to enter one command, enter:

```
SQL>! command
```

where *command* represents the operating system command you wish to execute. Once the command has executed, control is returned to SQL*Plus.

To execute more than one operating system command, press [Enter} after the [!] or HOST command.

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.

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

Resizing Windows

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

Return Codes

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

4

Using Oracle Precompilers and the Oracle Call Interface

This chapter provides the following precompiler and Oracle Call Interface information:

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*i* or any other ANSI SQL DBMS.

Precompiler Configuration Files

The `.cfg` system configuration files in the `$ORACLE_HOME/precomp/admin` directory are described in Table 4–1.

Table 4–1 System Configuration Files

Product	Configuration File
Pro*C/C++, 8.1.5	`pcscfg.cfg`
Object Type Translator, 8.1.5	`ottcfg.cfg`

Issues Common to All Precompilers

> **Note:** In order to run Oracle Precompiler demonstrations, you must already have installed Oracle8*i*.

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. 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*
- *Oracle8i Server Application Developer's Guide*

Pro*C/C++

For additional information regarding Pro*C/C++, Release 8.1.5, 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++

Before you use Pro*C/C++, verify that the correct version of the Operating System compiler is properly installed. The required version is documented in the *Oracle8i Installation Guide for Intel-Linux*.

Demonstration Programs

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

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

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

Use the makefile, $ORACLE_HOME/precomp/demo/proc/demo_proc.mk, 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, use the following command, which achieves the same result, 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 C demonstration programs for Pro*C/C++, enter the following command:

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

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

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

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

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

Some demonstration programs require you to run a SQL script from $ORACLE_HOME/precomp/demo/sql. To build a demonstration program and run the corresponding SQL script, include the make macro argument, RUNSQL=run, on the command line. For example, to create the calldemo demonstration program and run the required $ORACLE_HOME/precomp/demo/sql/calldemo.sql script, use the following command syntax:

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

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, enter the following:

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

For C source, statically linked, enter the following:

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

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

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

For C++ source, statically linked, enter the following:

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

For Linux restrictions on the use of shared libraries, refer to the Linux documentation.

Oracle Call Interface

Using the Oracle Call Interface

Before using the Oracle Call Interface (OCI), verify that the correct version of the Pro*C/C++ is properly installed. The required version for the Linux system is specified in Chapter 1 of the *Oracle8i Installation Guide for Intel-Linux*.

Demonstration Programs

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

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

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.

Use the makefile, `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk`, 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 to achieve the same result 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 this 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 -lC: not found
ld: fatal: library -lC_mtstubs: not found
ld: fatal: library -lcx: not found
```

You must include in LD_LIBRARY_PATH, the directory in which the specified libraries exist.

For example, if you are using the Intel-Linux Compiler C++, then the directory is `/opt/CDS++`. And, the following command must be entered:

```
$ setenv LD_LIBRARY_PATH \
${LD_LIBRARY_PATH}:/opt/CDS++
```

Some demonstration programs require you to run a SQL script manually before you execute the program. All of the scripts are located in the `$ORACLE_HOME/rbdms/demo` directory. 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, enter the following:

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

For C source, statically linked, enter the following:

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

To create the program `myprog` from the C++ source `myprog.cc`, use one of the following commands depending on the type of executable desired:

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

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

For C++ source, statically linked, enter the following:

```
$ 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, note the following:

- Do not modify the ordering of the Oracle libraries. Oracle libraries are included on the link line more than once so all symbols are resolved during linking. There are two reasons for this:
  - 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.
  - The Linux linker is a one-pass linker, meaning that the linker searches a library exactly once at the point it is encountered in the link line.
- If you add your own library to the link line, add it to the beginning or to the end of the link line. User libraries should not be placed between the Oracle libraries.
- If you choose to use a make utility such as `nmake` or GNU `make`, be aware of how macro and suffix processing differs from the make utility provided with Linux, `/usr/bin/make`. Oracle makefiles have been tested and are supported with the Linux `make` utility.
- Oracle library names and the contents of those libraries are subject to change between releases. Always use the `demo_`*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/8.1.5/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/8.1.5
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 runtime. Libraries that are linked at runtime 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 (for example, 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 startup. 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 re-create the Oracle shared library, exit **all** client applications using the

Oracle shared library, including all Oracle client applications such as 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 libclntsh.so
```

Using Signal Handlers

This section describes signals Oracle8*i* 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*i* uses the following signals for two-task communications:

Table 4–2 Signals for Two-Task Communications

Signal	Description
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 UNIX 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 Net8 protocol methods to indicate incoming networking events.
SIGURG	used by the 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 the Oracle client shared library. This link restriction is required only when using XA's dynamic registration (Oracle XA switch `xaoswd`).

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

5

Configuring Net8

This chapter provides the following Net8 configuration information:

Supplementary Documentation

The following documents provide a full discussion of Net8 features:

- *Oracle Net8 Administrator's Guide*
- *Oracle Networking Quick Reference Card for Net8*

Supplementary Information in README Files

Table 5–1 shows the location of `README` files for various bundled products. The `README` files describe changes since the last release.

Table 5–1 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`

Core Net8 Products and Features

This section provides information on the core features of Net8.

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

Net8 Files and Utilities

Location of Net8 Configuration Files

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. `$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 directory_name
```

For each system level configuration file, users may have a corresponding local private configuration file (stored in the user’s home directory). The settings in the 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 the $ORACLE_HOME/network/admin/samples directory.

The adapters Utility

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

```
% adapters
```

To display adapters linked with a specific executable, enter:

```
% adapters executable
```

For example, the following command displays the Net8 protocols linked with the oracle executable:

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

Oracle Connection Manager

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

Multi-Threaded Server

See Also: For information on the Multi-Threaded Server, see the *Oracle8i Server Concepts* and *Oracle8i Administrator's Guide*.

Oracle Names

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

Net8 Assistant

Oracle Java Runtime Environment is installed with Net8 Assistant (`$ORACLE_HOME/bin/net8asst`). When the Net8 Assistant command script is executed, the java command script supplied with JRE 1.1.6.2 is called explicitly, regardless of other Java installations on the system.

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

Net8 Protocol Support

The supported protocols for Net8, Release 8.1.5 on Linux are BEQ protocol, IPC protocol, RAW protocol, TCP/IP protocol.

ADDRESS Specification

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

Table 5–2 shows a summary of ADDRESS specifications for each protocol.

Table 5–2 ADDRESS Specification Summary

Supported Protocol	ADDRESS Specification

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

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

RAW	N/A

TCP/IP	(ADDRESS =
	    (PROTOCOL=TCP)
	    (HOST=*hostname*)
	    (PORT=*port_id*)
	)

The BEQ Protocol

The BEQ protocol 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 is used. In which case, a dedicated server will always be used, and the multi-threaded server is never used. This dedicated server is started automatically by the BEQ protocol, which waits for the server process to start and attach to an existing SGA. If the startup of the server process is successful, the BEQ protocol then provides inter-process communication through UNIX.

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

Specifying a BEQ ADDRESS

The BEQ protocol connection parameters are part of the ADDRESS keyword and 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')
)
```

The syntax for BEQ protocol connection parameters is described in Table 5–3.

Table 5–3 Syntax for BEQ Protocol Connection Parameters

Item	Description
PROTOCOL	Specifies the protocol 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 which the connection will be made. And, ORACLE_SID is the system identifier of the database to which the connection will be made.

Example 5–1 BEQ ADDRESS Specifying a Client

The following is an example of a BEQ ADDRESS:

```
(ADDRESS =
    (PROTOCOL = BEQ)
    (PROGRAM = /u01/app/oracle/product/8.1.5/bin/oracle)
    (ARGV0 = oracleV815)
    (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    (ENVS = 'ORACLE_HOME=/u01/app/oracle/product/8.1.5,ORACLE_SID=V815')
)
```

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

The IPC Protocol

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

For the IPC protocol, the location of the UNIX Domain Socket (IPC) file on UNIX systems changed after Oracle7, Release 7.1. Thus, if you have Oracle7, Release 7.1 installed on the same machine as Oracle8*i* 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 connection parameters are part of the ADDRESS keyword-value pair. You can enter the parameters in any order.

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

The syntax for IPC protocol connection parameters is described in Table 5–4.

Table 5–4 Syntax for IPC Protocol Connection Parameters

Item	Description
PROTOCOL	specifies the protocol to be used. The value is `ipc` and may be specified in either uppercase or lowercase.
KEY	service name of database or database identifier (ORACLE_SID).

Example 5–2 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 RAW Protocol

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

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

This feature is transparently turned on whenever it is appropriate. That is, if no existing features require that header information be transmitted, the headers are stripped off. For example, encryption and authentication require certain information to be sent along with each packet of information, so Raw Transport would not be enabled.

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

The TCP/IP Protocol

Oracle Corporation recommends that you reserve a port for your Net8 listener in the `/etc/services` file of each node on the network that defines the Net8 listener port. The port is commonly 1521. The entry lists 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 connection parameters are part of the ADDRESS keyword and value pair. You can enter the three parameters in any order.

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

Syntax for TCP/IP protocol connection parameters is described in Table 5–5.

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

Item	Description
PROTOCOL	specifies the protocol 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 5–3 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.

The Net8 Naming Support

See Also: For details on configuring the NIS Naming Support, see the *Net8 Administrator's Guide*.

Oracle Enterprise Manager (OEM)

Agent Service Discovery and Auto-Configuration

The Oracle Intelligent Agent requires no configuration, unless you want to integrate it with an SNMP system (see "").

See Also: For information on Oracle Names and the Net8 Assistant, see the *Net8 Administrator's 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 the `$ORACLE_HOME/network/agent/tcl` directory.

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

6

Running Oracle Data Option Demos

This chapter provides the following demonstration information:

Additional Documentation

The following documents provide in-depth information about the Oracle options available in release Release 8.1.5:

- *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*
- *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference*
- *Using Oracle8i interMedia with the Web*
- *Oracle8i interMedia Locator User's Guide and Reference*
- *Oracle8i interMedia Text Reference*
- *Oracle8i ConText to interMedia Text Migration*
- *Oracle8i Visual Information Retrieval User's Guide*
- *Oracle8i Visual Information Retrieval Java Client User's Guide*
- *Oracle8i Time Series User's Guide*
- *Oracle8i Spatial User's Guide and Reference*

Oracle8*i* interMedia

Oracle8*i* interMedia includes the following components:

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

Text

> **See Also:** *Oracle8i interMedia Text Reference*, and *Oracle8i ConText interMedia Text Migration*.

There are no demos for Text in Oracle8*i*.

Audio, Video, and Image

> **See Also:** *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* and *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference.*

Oracle8*i* interMedia includes a number of scripts and sample programs in the following directories:

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

Sample Audio Scripts

The audio scripts consist of the following files:

- `auddemo.sql` - audio demonstration that shows features of the audio object including:
  - checking interMedia objects
  - creating a sample table with audio in it
  - inserting NULL rows into the audio table
  - checking the rows out
  - checking all the audio attributes directly
  - checking all the audio attributes by calling methods
  - installing your own format plug-in using the two files, `fplugins.sql` and `fpluginb.sql` described in the next two list items and in *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* on how to extend interMedia Audio to support a new audio data format
- `fplugins.sql` - demo format plug-in specification that you can use as a guideline to write any format plug-in you want to support.
- `fpluginb.sql` - demo format plug-in body that you can use as a guideline to write any format plug-in you want to support.

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

Sample Program for Modifying Images or Testing the Image Installation

Once you have installed Oracle8*i* interMedia Image, you may choose to run the Oracle8*i* interMedia Image demonstration program. This program can also be used as a test to confirm successful installation.

This section contains the steps required to build and run the interMedia image demo.

The interMedia Image demo files are located in the `$ORACLE_HOME/ord/img/demo` directory, where `$ORACLE_HOME` is the ORACLE_HOME directory.

Demonstration (Demo) Installation Steps

1. The Oracle8*i* interMedia Image demo uses the SCOTT/TIGER database user. If this user does not exist, you must create it:

   ```
   % svrmgrl
   SVRMGRL> connect internal;
   SVRMGRL> create user SCOTT identified by tiger;
   SVRMGRL> grant connect,resource to SCOTT;
   ```

2. Create the image demo directory where `$ORACLE_HOME` is the ORACLE_HOME directory.

   ```
   % svrmgrl
   SVRMGRL> connect internal;
   SVRMGRL> create or replace directory imgdemodir as ‘$ORACLE_HOME/ord/img/
   demo’;
   ```

3. Grant privileges on the directory to PUBLIC:

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

4. If needed, make the `imgdemo` program.

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

Running the Demo

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

The program operates on `imgdemo.dat`, which is a bitmap (BMP) image in the demo directory. Optionally, you can supply an image file name on the command line, provided the file resides in the same directory as the demo. In either case, once the image has been manipulated by Oracle8*i* interMedia Image, the resulting image is written to the file `imgdemo.out` and can then be viewed with common rendering tools that you supply.

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

The image properties are extracted within the database using the setProperties( ) method. An UPDATE command is issued after the setProperties( ) invocation. This is required because the setProperties( ) invocation has only updated a local copy of the type attributes.

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

Example 6–1 Execute the Demo from the Command Line

Execute the demo by typing `imgdemo` on the command line. Optionally, you can use a different image in the demo by first copying the file to the directory in which the demo resides and then specifying its file name on the command line as an argument to `imgdemo`.

Use the following command:

```
   $ imgdemo optional-image-filename
```

The demo displays a number of messages describing its progress, along with any errors encountered if something was not set up correctly. Expect to see the following messages:

```
Dropping table IMGDEMOTAB...
Creating and populating table IMGDEMOTAB...
Loading data into cartridge...
Modifying image characteristics...
Writing image to file imgdemo.out...
```

```
Disconnecting from database...
Logged off and detached from server.
Demo completed successfully.
```

If the program encounters any errors, it is likely that either Oracle8*i* interMedia Image software has not been installed correctly or the database has not been started. If the program completes successfully, the original image and the resultant image, which has undergone the cutting and scaling described earlier, can be viewed with common image rendering tools.

Sample Video Scripts

The Video scripts consist of the following files:

- `viddemo.sql` - video demo that shows features of the video object including:
  - checking interMedia objects
  - creating a sample table with video in it
  - inserting NULL rows into the video table
  - checking the rows out
  - checking all the video attributes directly
  - checking all the video attributes by calling methods
  - installing your own format plug-in using the two files, `fplugins.sql` and `fpluginb.sql` described in the next two list items and in *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* on how to extend interMedia Video to support a new video data format
- `fplugins.sql` - demo format plug-in specification that you can use as a guideline to write any format plug-in you want to support
- `fpluginb.sql` - demo format plug-in body that you can use as a guideline to write any format plug-in you want to support

See the README.`txt` file in the `$ORACLE_HOME/ord/vid/demo` directory for requirements and instructions on how to run this SQL demo.

Java Demo

A Java demo has been provided to help you learn to use both the audio and video client-side Java classes so you can build your own applications. In these two demos, the audio and video object is instantiated at the client side and a number of access methods are invoked. The audio Java demo files are located in the

ORACLE_HOME/ord/aud/demo directory and the video Java demo files are located in the $ORACLE_HOME/ord/vid/demo directory. See the `README.txt` file in each directory for requirements and instructions on how to run each respective Java demo.

MediaAnnotator

The MediaAnnotator program is not contained on the Oracle8*i* interMedia CD. It (along with other free Oracle software) may be found at the following URL:

http://www.oracle.com/products/free_software/

Locator

> **See Also:** *Oracle8i interMedia Locator User's Guide and Reference.*

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

Sample Scripts

Sample Oracle8*i* interMedia Locator scripts are available in the following directory after you install this product:

```
$ORACLE_HOME/md/demo/geocoder
```

These scripts consist of the following files:

- `geohttp.sql`

  This file contains two parts. One part is for running a geocode function in interactive mode and the other is for running the geocode function in batch mode.

  – Interactive mode.

    See Example 1 in "GEOCODE1 Function (with lastline field)" in the *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this part of the file.

  – Batch mode.

    You must update the setup tables in the `nh_cs.sql` file before you run the `geohttp.sql` in batch mode. See the following examples in the *Oracle8i interMedia Locator User's Guide and Reference*: Example 2 in "GEOCODE1 Function (with lastline field)" or Example 3 in "GEOCODE1 Function (with lastline field)" for a listing of this part of the file.

- `geoindex.sql`

  This file contains:

  - A function named ESTIMATE_LEVEL to better estimate the index level for use with the spatial locator index for within-distance queries that use a radius distance greater than 100 miles. See the example in "ESTIMATE_ LEVEL" in the *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this file.
  - A procedure statement named SETUP_LOCATOR_INDEX that builds a setup spatial locator index on the location column that contains the spatial information within the cust_table table where the spatial information is stored. See the example in "SETUP_LOCATOR_INDEX", Chapter 2 in *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this file.

- `geolocate.sql`

  This file contains a routine that dynamically creates a geometry of interest and then queries against the NH_COMPUTER_STORES table to find out how many stores are within a 10-mile radius of the office. See Example 2 in "LOCATOR_ WITHIN_DISTANCE" the *Oracle8i interMedia Locator User's Guide and Reference* for a listing of this file.

Web Agent and Clipboard

See Also: *Using Oracle8i interMedia with the Web*

For this release, two components of Oracle8*i* interMedia, the Clipboard and Web Agent, are not available on the Oracle8*i* media. You can download the components from the Free Software download area of the Oracle Corporation web site:

http://www.oracle.com/products/free_software/index.html

The documentation, which includes `README` files and the manual *Using Oracle8i interMedia with the Web*, is included in the download.

Oracle8*i* Time Series Demos

See Also: *Oracle8i Time Series User's Guide*

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

Table 6–1 Oracle8i Time Series Demos

Description	Directory
Quick-start demo: quick and easy start using Oracle8*i* Time Series (See Chapter 1 in *Oracle8i Time Series User's Guide*.)	`demo/tsquick`
Usage demo for end users and product developers who want to use existing Oracle8*i* Time Series features (See Chapter 1 in *Oracle8i Time Series User's Guide*.)	`demo/usage`
Electric utility application demonstrating how to compute peak and off-peak summaries of 15-minute data	`demo/usageutl`
Java-based retrieval of time series data, using the prototype Oracle8*i* Time Series Java API and designed to run in a Web browser (See Chapter 1 in *Oracle8i Time Series User's Guide*.)	`demo/applet`
Simple Java code segments that perform time series operations and print the results (See Chapter 1 in *Oracle8i Time Series User's Guide*.)	`demo/java`
Demo showing the use of administrative tools procedures to "retrofit" existing time series detail tables; also, how to support time series queries for multiple qualifier columns in the time series detail table.	`demo/retrofit`
Advanced-developer demo for those who want to extend Oracle8*i* Time Series features	`demo/extend`
OCI demo showing how to call Oracle8*i* Time Series functions using the Oracle Call Interface	`demo/oci`
PRO*C/C++ demo showing how to call Oracle8*i* Time Series functions in applications created using the Oracle Pro*C/C++ Precompiler	`demo/proc`
Oracle Developer demo showing how to call Oracle8*i* Time Series functions in an Oracle Forms application	`demo/dev2k`

The `README.txt` file in the demo directory introduces the demos. Also, the directory for each demo contains a `README.txt` file with a more detailed description of that demo.

Oracle8*i* Visual Information Retrieval

> **See Also:** *Oracle8i Visual Information Retrieval User's Guide and Reference* and *Oracle8i Visual Information Retrieval Java Client User's Guide and Reference*

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

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

Environment

The following assumptions are made:

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

Running the Sample Program

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

Example 6–2 runs the sample program using the included image files. The images are compared using equal attribute weights:

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

Example 6–2 Run the Sample Program with Included Images

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

Example 6–3 shows how to specify your own images on the command line. The images must reside in the `$ORACLE_HOME/ord/vir/demo` directory.

Example 6–3 Run the Sample Program with Your Own Images

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

All six parameters: the 2 file names and 4 attribute weights (ranging from 0.0 to 1.0) must be specified in this sample program. Note that when using the VIRScore( ) operator in your own applications, it is only necessary to provide at least one attribute weight.

Several other sample image files have been provided in the VIRDEMODIR directory to demonstrate the effects of emphasizing the different visual attributes. You can use an image viewer (such as `xv`) to display the images, and then compare them using the sample program, experimenting with different weights.

> **See Also:** Appendix B in the *Oracle8i Visual Information Retrieval User's Guide and Reference* for more information.

Oracle8*i* Spatial

> **See Also:** *Oracle8i Spatial User's Guide and Reference*

Oracle8*i* Spatial does not contain any demos.

A

Optimal Flexible Architecture

This appendix provides the following Optimal Flexible Architecture information:

- Optimal Flexible Architecture (OFA)
- OFA Implemented on UNIX

Optimal Flexible Architecture (OFA)

Oracle Corporation recommends the Optimal Flexible Architecture (OFA) standard for Oracle8*i*. 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 such as 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 the following:

- 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 UNIX Directory Subtrees

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

Supports Concurrent Execution of Application Software

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

I/O Loads Tuning 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 UNIX

A careful naming strategy for database files eliminates data administration problems. The OFA rules provided here correspond to the original OFA recommendations published in *The OFA Standard: Oracle8i 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 A–1.

Table A–1 Syntax for Naming Mount Points

Item	Description
q	a string denoting that Oracle data is stored here
dm	the value of the initialization parameter DB_NAME (synonymous with the instance *sid* for single-instance databases)

For example, mount points named `/u01/oradata/test` and `/u02/oradata/test` allocate two drives for the Oracle test database.

Naming Directories

Home Directory Syntax

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

Table A–2 Syntax for Naming Home Directories

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

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

Placing home directories at the same level in the UNIX file system is advantageous 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*i* Server software in a directory matching the pattern /*pm*/*h*/`product`/*v*, as explained in Table A–3.

Table A–3 Syntax for Naming Oracle8i Server Software Directories

Item	Description
h	a standard directory name
v	the version of the software

For example, `/u01/app/oracle/product/8.1.5` indicates the start of the directory structure where the Oracle8*i* Server files are located. Set the `ORACLE_HOME` environment variable to this directory.

Naming 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 A–4:

Table A–4 Subdirectories for Database Administration Files

Subdirectory	Description
`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. Clean this subdirectory 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, if it was part of the database named `sab`:

```
/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 A–5.

Table A–5 Syntax for Naming Database Files

Item	Description
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 A–6. These tablespaces are in addition to those needed for application segments.

Table A–6 Special Tablespace

Tablespace	Description
SYSTEM	data dictionary segments
TEMP	temporary segments
RBS	rollback segments
USERS	miscellaneous user segments
INDX	index associated with data in USERS tablespace
OEM_REPOSITORY	repository for Oracle Enterprise Manager
DRSYS	Oracle interMedia segment

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

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

Naming Tablespaces

Name tablespaces descriptively using a maximum of eight characters. Although Oracle8*i* tablespace names can be 30 characters long, portable UNIX file names are restricted to 14 characters. The recommended standard for a data file basename is *tn*`.dbf`, where *t* is a descriptive tablespace name and *n* is a two-digit string. Because the extension plus the two-digit string occupy a total of six characters, only eight characters remain for the tablespace name.

Descriptive names 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 A–7 shows the syntax used for identifying classes of files.

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

Directory Structure Syntax	Description
`/u[0-9][0-9]`	user data directories
`/*/home/*`	user home directories
`/*/app/*`	user application software directories
`/*/app/applmgr`	Oracle apps software subtrees
`/*/app/oracle/product`	Oracle Server software subtrees
`/*/app/oracle/product/8.1.5`	Oracle Server, Release 8.1.5 distribution files
`/*/app/oracle/admin/sab`	`sab` database administrative subtrees
`/*/app/oracle/admin/sab/arch/*`	`sab` database archived log files
`/*/oradata`	Oracle data directories
`/*/oradata/sab/*`	`sab` database files
`/*/oradata/sab/*.log`	`sab` database redo log files

OFA File Mapping

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

Table A–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
					7.3.3/	ORACLE_HOME for 7.3.3 instances
					8.0.4/	ORACLE_HOME for 8.0.4 instances
					8.1.5/	ORACLE_HOME for 8.1.5 instances
			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
		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.

Note: Linux does not support raw devices at this time.

File Mapping for Multiple-Instance OFA Database

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

Table A–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
		19990120full.dmp		January 20, 1999 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 A–10. When installing an OFA-compliant database using the Oracle Universal Installer, ORACLE_BASE is by default set to `/pm/app/oracle`**.**

Table A–10 ORACLE_BASE Directory Structure and Content

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

ORACLE_HOME Directory

If you install an OFA-compliant Oracle Server, the ORACLE_HOME directory is `/pm/app/oracle/product/`*release_number*. ORACLE_HOME directory structure and content are described in Table A–11. Under UNIX, the ORACLE_HOME directory contains the following subdirectories, as well as a subdirectory for each Oracle product selected. You will have directories only for the products you have installed.

Contents of Product Subdirectories

Each product subdirectory contains the subdirectories described in Table A–11:

Table A–11 Contents of Product Subdirectories

Subdirectory	Description
admin	administrative SQL and shell scripts (for example, catalog.sql, catexp.sql, and demo.sql)
admin/*	special directories for other products
admin/resource	resource files
admin/terminal	runtime terminal files
demo	demonstration scripts and datafiles
doc	README files (for example, readmeunix.doc)
install	product installation scripts
jlib	product Java classes
lib	product libraries and distributed makefiles
log	trace files and log files (for example, orasrv.log and *.trc files)
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 A–12.

Table A–12 Examples of Product Subdirectories

Subdirectory	Contents
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 A–13.

Table A–13 admin Directory, File Naming Conventions

Script	Description
`cat*.sql`	creates catalog and data dictionary tables and views. The following files are run automatically during installation: `catalog.sql` (for all installations) `catproc.sql` (for all installations) `catparr.sql` (for Parallel Server option installations) `catrep.sql` (for all installations) `catproc.sql` in turn runs the scripts for creating the standard PL/SQL packages, such as DBMS_SQL and DBMS_OUTPUT.
`d*.sql`	downgrade scripts
`dbms*.sql`	additional database packages
`u*.sql`	upgrade scripts
`utl*.sql`	creates tables and views for database utilities

Filename Extensions

A description of filename extensions is shown in Table A–14.

Table A–14 Filename Extensions

Extension	Description
`.a`	object file libraries; Ada runtime libraries
`.aud`	Oracle audit file
`.bdf`	X11 font description file
`.bmp`	X11 bitmap file
`.c`	C source file
`.ctl`	SQL*Loader control file; Oracle Server control file
`.dat`	SQL*Loader datafile
`.dbf`	Oracle Server tablespace file
`.dmp`	Export file
`.doc`	ASCII text file
`.env`	shell script file for setting environment
`.h`	C header file; also, `sr.h` is a SQL*Report Writer help file

Table A–14 Filename Extensions

Extension	Description
`.jar`	Java class archive
`.l`	UNIX manual page
`.lis`	output of SQL*Plus scripts
`.log`	installation log files; Oracle Server redo log files
`.mk`	`make` files
`.msb`	NLS message file (binary)
`.msg`	NLS message file (text)
`.o`	object module
`.ora`	Oracle configuration files
`.orc`	installation prototype files
`.pc`	Pro*C source file
`.pco`	Pro*COBOL source file
`.ppd`	printer driver file
`.sh`	Bourne shell script file
`.sql`	SQL* script files
`.sys`	Bourne shell script file
`.tab`	SQL* script file
`.trc`	trace files
`.tut`	Bourne shell script file
`.utd`	Uniform Terminal Definitions
`.zip`	Zip file

Index