



MIMER

Installation and Usage Guide for UNIX

Version 8.1

Copyright © 1999 Sysdeco Mimer AB

MIMER version 8.1 Installation and Usage Guide for UNIX

January, 2000

Copyright © 1999 Sysdeco Mimer AB.

Published by Sysdeco Mimer AB,

P.O. Box 1713,

S-751 47 Uppsala, Sweden.

Tel +46(0)18-18 50 00.

Fax +46(0)18-18 51 00.

Internet: <http://www.mimer.com>

Produced by Sysdeco Mimer AB, Uppsala, Sweden.

All rights reserved under international copyright conventions.

The contents of this manual may be printed in limited quantities for use at a MIMER installation site. No parts of the manual may be reproduced for sale to a third party.

FOREWORD

Documentation objectives

This guide serves as a complement to the *MIMER System Management Handbook* and to the *MIMER/SQL Programmer's Manual* and covers subjects that are specific to UNIX platforms.

Prerequisites

There are no prerequisites for users of this guide. However, it is to the reader's advantage to be familiar with the operating system on the host UNIX computer.

A working knowledge of system management within the UNIX environment is required for the installation of MIMER, since installation is performed by the superuser, i.e. 'root'.

Organization of this manual

This guide contains the following chapters:

- Chapter 1** is a brief introduction to this guide.
- Chapter 2** describes the installation and the distributed components.
- Chapter 3** describes the creation of an initial database.
- Chapter 4** describes how the database server is managed.
- Chapter 5** describes how to create a basic MIMER-based application.
- Chapter 6** describes how to execute a MIMER-based application.
- Chapter 7** describes how to build and execute example programs.

The guide also contains the following appendices:

- Appendix A** describes matters concerning data types.
- Appendix B** describes how to use applications based on MIMER7 with a MIMER8 installation.

Related MIMER publications

- **MIMER/SQL Reference Manual** contains a complete description of the syntax and usage of all statements in MIMER/SQL.
- **MIMER/SQL User's Manual** contains a description of the Batch SQL facilities. A user-oriented guide to the SQL statements is also included, which may provide help for less experienced users in formulating statements correctly (particularly the SELECT statement, which can be quite complex).
- **MIMER/SQL Programmer's Manual** contains a description of how MIMER/SQL can be used within the context of application programs, written in conventional programming languages.
- **MIMER System Management Handbook** describes system administration functions, including export/import, backup/restore, and the statistics functionality. The information in this manual is used primarily by the system administrator, and is not required by application developers. The SQL statements which are part of the System Management API are described in the *MIMER/SQL Reference Manual*.
- **MIMER Shadowing Handbook** describes the optional Shadowing module, which allows several concurrent copies of a databank to be maintained at the same time. This provides extra resilience to disk crashes and allows “backup on the fly”.
- **MIMER Release Notes** contain information relating to the MIMER release for which they are supplied. New and changed functions are described, known problems and restrictions are pointed out and some platform specific information is given.

Acronyms and trademarks

API	Application Programming Interface.
Databank	“Databank” is the MIMER term for the physical file in which one or more MIMER tables are stored. A databank corresponds to one file in the operating system. A database contains several databanks.
Database	A database is a collection of databanks, tables, shadows, etc., all defined as objects in the data dictionary. A computer may have several databases operating simultaneously, but no information is shared between them . Each database has a unique name, registered in the /etc/sqlhosts file.
Database home directory	The directory where the system databank file containing the data dictionary and some other files that form part of the database are located.
Dynamic SQL	SQL statements constructed at runtime and passed to the database management system for execution.

Embedded SQL	The term used for SQL statements when they are embedded in a traditional host programming language.
MIMER8	The general term used for version 8 of MIMER.
MRS	MIMER Release and Security - a system that manages the MIMER software licenses.
ODBC	Microsoft's Open Database Connectivity, a specification for a database API in the C language, independent of any specific DBMS or operating system.
PDF	Portable Document Format. File format for the on-line documentation to be read by an Adobe Acrobat Reader.
PSM	Persistent Stored Modules (i.e. Stored Procedures).
SQL	Structured Query Language.
Unix	Unix is a trademark registered by the X/Open Company.

(All other trademarks are the property of their respective holders.)

CONTENTS

1	INTRODUCTION	
2	MIMER SYSTEM COMPONENTS	
2.1	Installing MIMER	2-1
2.1.1	The installed directory structure	2-2
2.1.2	Verifying the 'mimer' TCP/IP service	2-2
2.2	Distributed files	2-3
2.2.1	Documentation files (./doc)	2-3
2.2.2	Example files (./examples).....	2-3
2.2.3	Executable programs (./bin).....	2-4
2.2.4	Library files (./lib)	2-5
2.3	Removing a MIMER installation	2-6
3	CREATING A DATABASE	
3.1	Installing the MRS license key	3-1
3.2	Installing a database using dbinstall	3-2
3.3	Distribute databank files.....	3-3
3.4	Verifying the installed database	3-3
3.4.1	Database home directory	3-3
3.4.2	Database server state	3-4
3.5	Database registry file.....	3-4
3.5.1	The mimhosts command.....	3-4
3.5.2	Adding an existing database to the registry file	3-5
3.5.3	Default database	3-5
4	MANAGING A DATABASE SERVER	
4.1	The mimadmin command.....	4-1
4.2	Using mimadmin	4-2
4.3	Automatic database start and stop	4-3
4.4	Using raw device partitions.....	4-5
4.4.1	Creating a raw disk partition.....	4-5
4.4.2	Arranging for putting a databank on a raw device	4-6
4.4.2	Arranging for taking a databank off a raw device.....	4-6
5	CREATING MIMER APPLICATIONS	
5.1	The ESQL command.....	5-1
5.1.1	ESQL syntax.....	5-2
6	RUNNING MIMER APPLICATIONS	
6.1	Selecting a MIMER installation	6-1
6.1.1	Several installations on one machine	6-1
6.2	Selecting a database server.....	6-2
6.3	Executing a MIMER application.....	6-2
6.4	BSQL command line editing	6-3

7	EXAMPLE PROGRAMS	
7.1	Embedded SQL example programs.....	7-1
7.1.1	Simple embedded program, EXAMPLE	7-1
7.1.2	Dynamic SQL program, DSQSAMP.....	7-3
7.1.3	BLOB program, BLOBSAMP	7-4
7.1.4	Programs calling PSM, FREQCALL/WAKECALL.....	7-6
7.2	ODBC example programs	7-7
7.2.1	Dynamic SQL program, OSQSAMP.....	7-7
A	DATA TYPES USED IN MIMER	
A.1	Internal MIMER representation	A-1
A.2	External data types supported by MIMER	A-1
B	USING MIMER7 APPLICATIONS WITH MIMER8	
B.1	Client-server access.....	B-1
B.2	Local client/server access.....	B-2
B.3	Re-link applications	B-3

1 INTRODUCTION

MIMER is an advanced database management system developed by Sysdeco Mimer AB. This guide describes UNIX-specific information, such as how to install, establish, manage and maintain a MIMER database.

From time to time there are descriptions that vary between different implementations of the UNIX operating system. Such variations are described in an enclosing text box and are marked with an icon that specifies the operating system.

2 MIMER SYSTEM COMPONENTS

This chapter describes the operations performed and the components made available when installing MIMER. It also describes how to uninstall MIMER.

2.1 Installing MIMER

The MIMER software is installed using the **miminstall** command, as described in the README file that was obtained when unpacking the distribution (using the “tar xvf” command). During installation the following points should be noted:

- **Installation directory.** The default is /opt.
- **Service registration.** The default will update /etc/services by adding a TCP/IP service for MIMER on port number 1360, registered for MIMER use. The default /etc/services entry looks as follows:

```
mimer 1360/tcp
```

- **TCP/IP port number dispatcher setup.** The default will update /etc/inetd.conf and make the UNIX inetd daemon listen to the 1360 port for automatic start of the mimtcp program when a connection attempt is made. If the /etc/inetd.conf file is updated, the UNIX inetd server must be forced to read the updated configuration, which is automatically taken care of by the miminstall command. (For manual re-initiation of the inetd daemon, the command ‘kill -HUP *inetd_pid*’ should be used). The default /etc/inetd.conf entry looks as follows:

```
mimer stream tcp nowait root /usr/bin/mimtcp mimtcp -l
```

By default, mimtcp is started using the “-l” option which produces a log file as **/etc/mimtcp.log**.

- **System-wide environment variable setup.** The default will update /etc/profile and define the environment variable MIMER_HOME to be the location of the installed software.

2.1.1 The installed directory structure

When miminstall has executed successfully, the software installation is complete. The installed directory structure is shown in the following example:

```
# ls -l /opt/mimer811A
total 8
drwxr-xr-x  2 root    other      512 Aug 20 15:15 bin
drwxr-xr-x  2 root    other      512 Aug 20 15:15 doc
drwxr-xr-x  2 root    other      512 Aug 20 15:15 examples
drwxr-xr-x  2 root    other      512 Aug 20 15:15 lib
#
```

2.1.2 Verifying the 'mimer' TCP/IP service

When MIMER is installed using default options, the UNIX inetd daemon should be listening to the MIMER TCP/IP service. This can be verified by using the **netstat** command:

```
$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         (state)
.
.
tcp        0      0 *.mimer                 *.*                     LISTEN
.
.
$
```

If the “mimer” service cannot be found in the listing, the `/etc/services` and `/etc/inetd.conf` files must be verified. To employ the `/etc/inetd.conf` file the inetd daemon should be reinitiated as follows, where the PID (Process-ID) of the inetd daemon is obtained and then used together with the kill command:

```
$ ps -ef | grep inetd
root    129      1   0   May 10   ?           0:12 /etc/inetd
root  22875  22869   2  16:25:25 ttys1     0:00 grep inetd
$ kill -HUP 129
$
```

2.2 Distributed files

The distributed files are all located in the installed directory structure - see Section 2.1.1 for a directory listing.

2.2.1 Documentation files (./doc)

readme_doc.txt	A plain ASCII text file containing information on how to get started using the on-line documentation, i.e. the Adobe Acrobat PDF files.
rn81mim.pdf	MIMER Release Notes.
shadow.pdf	MIMER Shadowing Handbook.
sqlprog.pdf	MIMER/SQL Programmer's Manual.
sqlref.pdf	MIMER/SQL Reference Manual.
sqlusers.pdf	MIMER/SQL User's Manual.
sysman.pdf	MIMER System Management Handbook.
unixguide.pdf	MIMER Usage Guide for UNIX (this guide).

2.2.2 Example files (./examples)

blobsamp.ec	Example of a program written in embedded C that stores and retrieves binary data.
crehotdb.dat	Contains SQL commands that create an example database.
dsq.ec	Embedded C example that demonstrate the use of dynamic SQL.
dsq.h	Header file for the dynamic SQL example.
dsq1samp.c	Main program for the dynamic SQL example.
ex_makefile	Example makefile to be copied and modified for site specific purposes.
example.ec	Very simple embedded C example.
examples.dat	Text file containing examples of SQL commands. This assumes that the example database is loaded.
freqcall.ec	Example of a program written in embedded C that calls a stored procedure.
makeopt	Example settings of various make, compiler and loader options.
osq.ec	Example of a program written in embedded C that uses the ODBC API.

singledefs.dat	Contains a template for the database parameters used in single-user mode (see Section A.4 of the <i>MIMER System Management Handbook</i>).
wakecall.ec	Example of a program written in embedded C that calls a stored procedure.

2.2.3 Executable programs (./bin)

asctoexp	Program that converts a readable export file to a databank export file usable on for example UNIX.
bsql	Program that executes SQL statements which are entered interactively or read from a command file (described in the <i>MIMER/SQL User's Manual</i>).
dbc	Program that can check if a databank file is internally consistent (described in the <i>MIMER System Management Handbook</i>).
dbinstall	The database installation command, used when creating a new database. (Invokes the mimsdbgen, mimdevenv and mimexampldb commands).
dbopen	Program that opens and brings up to date all databanks in a database (described in the <i>MIMER System Management Handbook</i>).
dbserver	The database server. (Do not start this program directly, use the mimcontrol or mimadmin commands to do this).
esql	Pre-processor for embedded SQL.
expptoasc	Program that converts databank export files to formatted, readable files.
mimadmin	Database administration menu system. (Invokes the mimcontrol, mimhosts and miminfo programs).
mimcontrol	Program that can be used to control database servers (see the <i>MIMER System Management Handbook</i>).
mimdevenv	Command for creating an initial example development environment.
mimexampldb	Command for creating the example databank and example data that is used in examples given in the MIMER documentation.
mimhosts	Program that can be used to manage the database registry file /etc/sqlhosts.
miminfo	Program that can display status information for a database server.
mimlink	Command for creating symbolic links between a MIMER installation and /usr, i.e. /usr/lib and /usr/bin.

mimlistdb	Command for listing started MIMER database servers on the current node.
mimpath	Program that can display the databank search path for a given database.
mimsdbgen	Command for creating an initial database, i.e. the MIMER system databanks. (This is a front-end to the sdbgen program).
mimstatln	Program that can display the source file for a symbolic link (internal utility, used from other tools).
mimtcp	Program that can act as a TCP/IP port server for MIMER client/server connections. The UNIX inetd daemon listens to a port number, generally the port registered for MIMER (1360). When a connection attempt is made on the mimer service, inetd starts the mimtcp program which distributes the incoming database connection to the correct MIMER8 server running on the computer node.
mimuninstall	Command used to remove a MIMER software installation.
mimunlink	Command used to remove symbolic links from /usr that were created by the mimlink command.
mimversion	Program that can display the version for the MIMER product installed.
sdbgen	Program used to create the initial databank files in a database (used by dbinstall).
up7to8	Program that converts a MIMER version 7 database to MIMER8 (see the <i>MIMER Release Notes</i>).
util	Utility functions for a database (see the <i>MIMER System Management Handbook</i>).

2.2.4 Library files (./lib)

libmimdb	Shared library containing code for running a MIMER database in single-user mode. This library is mapped in dynamically when required.
libmimer	MIMER applications should be linked with this shared library, containing code for the MIMER database client API. When the application is activated, the UNIX system locates the shared library in /usr/lib.

2.3 Removing a MIMER installation

To remove a MIMER installation, perform the steps listed below. (For a description of how to remove MIMER databases, see Section 3.11 of the *MIMER System Management Handbook*).

- Check that no MIMER applications or database servers are using the installation.
- Execute the **mimuninstall** command. If this is the only MIMER installation on the node, all the files can be removed. Otherwise, files that are used by other installations on the node should be kept (mimuninstall will prompt you to confirm file deletion for each of these files).

```
$ su
# mimuninstall /opt/mimer811A
.
```

- When MIMER is installed, the default configuration enables the computer node to act as a server in a client/server environment and entries will have been added in the `/etc/services` and `/etc/inetd.conf` files. You should consider removing these entries or updating them to point to an alternative installation.
- Any MIMER-related commands for this installation that have been added to the common UNIX login file (`/etc/profile`) or to the UNIX startup/shutdown procedure should be removed.

3 CREATING A DATABASE

This chapter describes how to create a database on a UNIX platform. Important information about database installation and administration is detailed in the *MIMER System Management Handbook*. It is strongly recommended that this information be read.

Tasks to perform when initially installing a MIMER database

- Install the MRS license key.
- Execute the dbinstall command.
- Distribute the databank files.
- Verify the database installation.
- Database registry file management

3.1 Installing the MRS license key

To get the license key data (see Section 3.8 of the *MIMER System Management Handbook* for details), the node name must be provided to the MIMER distributor.

The key data provided by the MIMER distributor should be entered into the file **/mimkey8**.

Make sure that all UNIX users authorized to run MIMER have read access to the /mimkey8 file.

3.2 Installing a database using dbinstall

Details about the database environment can be found in Chapter 2 of the *MIMER System Management Handbook*.

A MIMER database server is installed by the dbinstall command. A personal single-user database can be created by using the mimsdbgen program (see example in Section 5.2.1) or by using the sdbgen program standalone (see Chapter 3 of the *MIMER System Management Handbook*).

The dbinstall syntax is as follows:

dbinstall [options]

Parameter	Parameter options	Parameter description
options	<i>-s database directory password</i>	<p><i>-s</i> is an optional flag and specifies a silent install.</p> <p><i>database</i> is the database name to be used</p> <p><i>directory</i> is the base directory to install in (a subdirectory will be created)</p> <p><i>password</i> is the password to be used for the System Administrator in the installed database</p>

Note: All database installation operations, including execution of the dbinstall command, should be performed by the superuser, i.e. 'root'.

The dbinstall command can be invoked as in the following example, using a default MIMER installation path:

```
$ su
# /opt/mimer811A/bin/dbinstall
```

When the dbinstall command is executed it will prompt for input on certain options for the database creation. The prompts request input on the following:

- Database name (No default. Each database must be given a unique name)
- Database home directory (Default is the current working directory)
- MIMER database administrator password (No default).

During the execution of dbinstall the system-wide database registry file, /etc/sqlhosts, is created automatically if it does not exist already. The installed database is registered, the MIMER system databanks are created and the database server program is started for the database. If the execution of dbinstall was successful, the database is ready to use. Information on establishing a MIMER database can be found in Chapter 3 of the *MIMER System Management Handbook*.

Note: It is important that the databank files are distributed over the available disks before the database is used in a production situation (see the next section).

3.3 Distribute databank files

The MIMER databank files are by default created in the database home directory, which is not the ideal arrangement from a **security** or **performance** point of view. Please refer to Section 2.3 of the *MIMER System Management Handbook* for important information about databanks and guidelines relating to their placement and organization.

If there is more than one disk available on the system, it is recommended that directories be created on those disks specifically for locating databanks. When created, the LOCAL definition of the database should be updated in the `/etc/sqlhosts` file by changing the single home directory path to a directory path list that includes these directories. This update is shown in the following example for the database “hotel”, where the database home directory is located on a disk mounted as `/mnt` and the additional databank directories are located on disks mounted as `/extra` and `/extra2`:

```
hotel      /mnt/db/hotel:/extra/dbopt/hotel:/extra2/dbopt/hotel
```

After this update is made, the database server can be stopped and the selected databank files can be moved from the database home directory to their new locations, using the UNIX `mv` command. The information given in the *MIMER System Management Handbook* (referred to earlier in this section) should be used when deciding where to locate each databank file.

3.4 Verifying the installed database

3.4.1 Database home directory

When `dbinstall` has executed successfully, the database installation directory, containing the database home directory, appears as in the following example (the server configuration file “multidefs” was automatically created when the database server was first started and it initially contains the recommended default values for your system):

```
# ls -lR /mnt/db
/mnt/db:
total 2
drwxrwx---  2 root      other          512 Aug 24 11:00 hotel

/mnt/db/hotel:
total 17478
-rw-----  1 root      other        2048000 Aug 24 11:00 logdb.dbf
-rwx-----  1 root      other           230 Aug 24 11:00 mimer.log
-rw-rw----  1 root      other         1283 Aug 24 11:00 multidefs
-rw-----  1 root      other        2048000 Aug 24 11:00 sqldb.dbf
-rw-----  1 root      other        2764800 Aug 24 11:00 sysdb8.dbf
-rw-----  1 root      other        2048000 Aug 24 11:00 transdb.dbf
#
```

3.4.2 Database server state

The database server is started automatically and it can be seen by using the UNIX **ps** command or by using the **mimcontrol** command:

```
# ps -ef | grep dbserver
  root 25431      1  0 11:07:03 ?          0:02 dbserver HOTEL
  root 25436 25388  0 11:10:19 pts/5    0:00 grep dbserver
# mimcontrol -c hotel
The database server is Running
New logins are Enabled
Directory for SYSDB8 is /mnt/db/hotel
Number of connections to server: 0
PID of database server process: 25431
Database server start date/time: 1998-08-24 11:07
#
```

3.5 Database registry file

A database installed using the **dbinstall** command is automatically added to the database registry file (`/etc/sqlhosts`). The **mimhosts** command can be used to manually administer the `/etc/sqlhosts` file. (See Appendix B of the *MIMER System Management Handbook* for details on the `sqlhosts` file).

3.5.1 The mimhosts command

The **mimhosts** command is used to administer the database registry file (`/etc/sqlhosts`). If the command is executed without any parameters, it will list the complete `/etc/sqlhosts` file.

The **mimhosts** syntax is as follows:

mimhosts [*options*]

The table below lists the various alternatives that can be specified for *options*:

Parameter options	Parameter description
	List all entries in the <code>/etc/sqlhosts</code> file.
<i>-d</i>	List the default database entry.
<i>-d database-name</i>	Set the named database to be the default database.
<i>-l</i>	List the local database entries.
<i>-l database-name pathlist</i>	Add an entry for the named local database, <i>pathlist</i> is a colon-separated directory path list specifying the databank file locations.
<i>-L database-name</i>	Remove the named local database entry.
<i>-r</i>	List the remote database entries.
<i>-r database-name node [protocol interface service]</i>	Add an entry for the named remote database, which is local on <i>node</i> . By entering <i>protocol</i> , <i>interface</i> and <i>service</i> , the defaults are overridden.
<i>-R database-name</i>	Remove the named remote database entry.

Note: The **mimhosts** command only operates on the `/etc/sqlhosts` file. E.g. removing a database using the **mimhosts** command only removes the entry for it from the `/etc/sqlhosts` file. The database files remain intact and unaltered.

Example, to display local databases:

```
$ su
# mimhosts -l
  SINGLE      .
  hotel       /mnt/db/hotel
#
```

3.5.2 Adding an existing database to the registry file

In some situations, use of the `dbinstall` command is not applicable. This is the case if the database already exists, e.g. where the database has been upgraded from version 7. If such database is to be added to the `/etc/sqlhosts` registry file, the `mimhosts` command can be used as follows (the database used in the example is called “customers”):

```
$ su
# mimhosts -l customers /opt/customers:/extra/customers
# mimhosts -l
  SINGLE      .
  hotel       /mnt/db/hotel
  customers   /opt/customers:/extra/customers
#
```

3.5.3 Default database

A system-wide default database can be specified in the database registry file. In the following example, the database “hotel” is set to be the default database (The default database specified in this way will be used if the environment variable `MIMER_DATABASE` is not set):

```
$ su
# mimhosts -d hotel
# mimhosts -d
Default database: hotel
#
```


4 MANAGING A DATABASE SERVER

This chapter describes how to administer a database server under UNIX. Important information relating to the management of database servers is found in the *MIMER System Management Handbook*. It is strongly recommended that this information be read.

Note: It is assumed that most database management operations are performed by the superuser, i.e. 'root'.

4.1 The mimadmin command

Any number of database servers may be started on a machine, the number is limited only by the computer system resources, but each database server operates on its own database, i.e. different database servers can not operate on the same database.

Database servers can be administered by using the **mimadmin** command, which has a menu based interface for easy access to all database servers on a node. One mimadmin session can be used to perform commands directed toward the server for any database installed on the machine.

The mimadmin command invokes programs such as mimcontrol and miminfo which are described in the *MIMER System Management Handbook*.

The mimadmin command can be executed successfully by any UNIX user, however, the superuser (root) password is required in order to perform the majority of the operations available. If already in superuser mode, no password is required.

Note: Not all the options for the invoked commands are supported in the mimadmin menu interface. Any options not supported in the mimadmin menu interface can be performed by starting the individual standalone programs.

The syntax for the mimadmin command is as follows:

mimadmin [database-name]

Parameter	Parameter description
<i>database-name</i>	Optional, specifies the name of the database server to manage initially (others can be selected once mimadmin is running). If <i>database-name</i> is not specified, the server for the default database is managed.

4.2 Using mimadmin

The `mimadmin` command operates against a target database. The name of the target database can be specified as a command-line argument. If the database name argument is omitted, the definition of the `MIMER_DATABASE` environment variable will be used to determine the target database. If `MIMER_DATABASE` is undefined, the default database specified in the `/etc/sqlhosts` file will be used. The current target database can be switched to another by using an option in the `mimadmin` menu system.

The `mimadmin` command can be used for controlling databases, monitoring databases and administering database definitions in the `/etc/sqlhosts` file.

Controlling databases

Options are available for various control functions such as:

- start and stop a database server
- enable and disable logins to a database server
- display the operational state of a database server
- list the log file and the configuration file
- display the contents of the database home directory

Monitoring databases

Options are available for various monitoring functions such as:

- list users logged on to the database
- report performance parameters from the database server

Administering databases

Options are available for various administration functions such as:

- list registered databases, i.e. their names and stored properties
- add databases to the `/etc/sqlhosts` file
- delete databases from the `/etc/sqlhosts` file
- change system-wide default database

4.3 Automatic database start and stop

For automatic start and stop of a MIMER database server, a configuration should be set up in the operating system, usually in the `/etc/init.d` or `/sbin/init.d` directory (the exact location is platform dependent). See the corresponding man pages for `init.d`, `rc0`, `rc`, etc.

The `init.d` directory contains initialization and termination scripts, executed when changing operating system init states. These scripts are linked, when appropriate, to files in the `rc?.d` directories, where '?' is a single character corresponding to the init state. See `init(1M)` for definitions of the states.

File names in `rc?.d` directories are usually of the form `[SK]nn<init.d-filename>`, where `S` means start this job, `K` means kill this job and `nn` is the relative sequence number for killing or starting the job. When entering a state (`init S`, `0`, `2`, `3`, etc.) the `rc[S0-6]` script executes those scripts in `/etc/rc[S0-6].d` that are prefixed with `K` followed by those scripts prefixed with `S`. When executing each script in one of the `/etc/rc[S0-6]` directories, the `/sbin/rc[S0-6]` script passes a single argument. It passes the argument 'stop' for scripts prefixed with `K` and the argument 'start' for scripts prefixed with `S`. There is usually no harm in applying the same sequence number to multiple scripts. In this case the order of execution is deterministic but unspecified.

An `inet.d` script for automatic start and stop of a MIMER database server, typically called **mimer**, may look as in the example file below. In this example, the target MIMER database server is named **hotel** and the MIMER software is installed in `/opt/mimer811A`. By adding different entries into the file, several MIMER systems can be managed.

Example file `/etc/init.d/mimer`:

```
#!/bin/sh
PATH=/bin:/usr/bin
export PATH

case "$1" in
  'start')
    MIMER_HOME=/opt/mimer811A
    export MIMER_HOME
    if [ -f $MIMER_HOME/bin/mimcontrol ]
    then
      echo "Starting Mimer database server hotel..."
      $MIMER_HOME/bin/mimcontrol -s hotel &
      sleep 3
    fi
    ;;
  'stop')
    MIMER_HOME=/opt/mimer811A
    export MIMER_HOME
    if [ -f $MIMER_HOME/bin/mimcontrol ]
    then
      echo "Stopping Mimer database hotel..."
      $MIMER_HOME/bin/mimcontrol -t hotel
    fi
    ;;
  *)
    echo "init.d/mimer: $1 ignored"
    ;;
esac
```

Examples of links created for /etc/init.d/mimer to various rc?.d directories:

```
ln /etc/init.d/mimer /etc/rc0.d/K01mimer
ln /etc/init.d/mimer /etc/rc2.d/K04mimer
ln /etc/init.d/mimer /etc/rc2.d/S94mimer
```

AIX

The behavior on AIX is different. For automatic startup the following example lines should be added late in the /etc/rc script:

```
MIMER_HOME=/opt/mimer811A
export MIMER_HOME

if [ -f $MIMER_HOME/bin/mimcontrol ]
then
    $MIMER_HOME/bin/mimcontrol -s hotel &
    sleep 3
fi
```

For automatic shutdown, the file /etc/rc.shutdown should be created (or updated if it already exists). The file may look as shown in the following example:

```
#!/bin/sh
PATH=/bin:/usr/bin
export PATH
MIMER_HOME=/opt/mimer811A
export MIMER_HOME

if [ -f $MIMER_HOME/bin/mimcontrol ]
then
    $MIMER_HOME/bin/mimcontrol -t hotel
fi
```

The /etc/rc.shutdown file must be executable, i.e. do:

```
# chmod 755 /etc/rc.shutdown
```

Note: The following points are general guidelines:

- 1) The MIMER_HOME environment variable must be defined in order to execute the mimcontrol command.
 - 2) The database server startup should be performed in background to avoid any possibility of disturbing the operating system boot procedure. Instead, a three second sleep should be added after each database server initiation.
 - 3) At shutdown, the MIMER system should be closed down before the machine goes into single user mode.
-

4.4 Using raw device partitions

The term “raw devices” applies to the character oriented disk device files (as opposed to the block oriented ones) normally found in `/dev`. These device files are a part of the interface between the hardware disks and the UNIX system software.

In most UNIX systems it is a performance advantage to use raw device files for data storage. By using raw devices, the UNIX file system (which uses index pages to locate the file blocks) is bypassed and the operating system is able to perform more effective I/O. Of course the effect also depends on several other factors such as file size, transaction rate, I/O-implementation, etc. Generally, large and/or frequently used databanks should be stored in raw device files.

Note! Familiarity with raw device files is recommended. It is essential that a raw device file be correctly defined, since the normal UNIX file handling safeguards do not apply. Some important points to be considered:

- Usually the first cylinder on the disk (cylinder 0), or corresponding, should be avoided when partitioning, since operating system information for the disk is stored there.
 - Overlapping partitions must be manually avoided.
 - It should be verified that backup procedures include raw devices.
 - Always take a complete backup before enabling use of a raw device.
-

The major disadvantage of using a raw device file is that the maximum file size is fixed by the size of the partition. If the partition becomes full, the raw device file must be moved to a larger partition, if one is available. In the worst case, the disk must be reformatted in order to create a larger partition. ***Always take a complete backup before reformatting a disk and before setting up any raw device files for use!***

Use of the raw device interface is completely transparent from the point of view of a MIMER user. A user has no way of distinguishing a system using raw device files from a system where ordinary UNIX files are used to store databanks.

4.4.1 Creating a raw disk partition

If it is decided to use raw I/O towards a disk and a character special device for the disk partition in question is missing, such a device must be created. This can be done from a Logical Volume Manage (LVM) or on some systems by using the `mknod` command.

LVM

When a logical volume is created a corresponding character oriented raw device is created along with the block oriented one. When using raw I/O towards the character oriented device, make sure the block oriented device is not used (and not mounted at boot).

Mknod

The major and minor numbers for the disk partition must first be identified and then the character special, non-buffering, device can be created:

```
# ls -l sdb2
brw-rw---- 1 root    disk      8, 18 May  5 1998 /dev/sdb2
# mknod rsdb2 c 8 18
# ls -l *sdb2
crw-rw---- 1 root    root      8, 18 Oct 18 08:51 /dev/rsdb2
brw-rw---- 1 root    disk      8, 18 May  5 1998 /dev/sdb2
#
```

4.4.2 Arranging for putting a databank on a raw device

The database server must be stopped:

```
# mimcontrol -t m81pelle
```

The databank file should be renamed:

```
# mv transdb.dbf transdb.dbf.ORDINARY
```

Create a soft link to the raw device file, using the original databank file name. Make sure root is the only one with access to the partition:

Note! It must be verified that the partition that corresponds to the raw device is not part of a file system that can be mounted.

```
# ln -s /dev/rdisk/c1t2d0s5 transdb.dbf
# chmod 600 /dev/rdisk/c1t2d0s5
# chmod 600 /dev/dsk/c1t2d0s5
```

Copy the original databank file onto the soft link and restart the database server:

```
# cp transdb.dbf.ORDINARY transdb.dbf
# mimcontrol -s m81pelle
```

4.4.2 Arranging for taking a databank off a raw device

The database server must be stopped:

```
# mimcontrol -t m81pelle
```

The raw device databank file must be copied to an ordinary block oriented file, located on a partition large enough to hold the complete raw device partition. Then the soft link can be removed:

```
# cp transdb.dbf /extra/transdb.dbf
# unlink transdb.dbf
```

Now the raw device partition can be used for other purposes. However, to be able to start the database system again, the databank file (in this case /extra/transdb.dbf) must be locatable by the database server. This can be achieved by arranging for the databank file to be moved back to its original location. Another solution is to update the /etc/sqlhosts file by adding an optional databank location, see Section 3.3.

5 CREATING MIMER APPLICATIONS

An application that accesses a MIMER database server is usually created either by using SQL embedded in a host programming language or by using the ODBC database API with the C host language.

Information on the availability and use of Driver Managers for ODBC on UNIX can be provided by your MIMER distributor. The library to specify as the ODBC driver when defining a MIMER ODBC data source is **libmimer**.

The ESQL preprocessor is used to create embedded SQL applications, as described in this chapter.

Further information can be found in the *MIMER/SQL Reference Manual* and the *MIMER/SQL Programmer's Manual*.

5.1 The ESQL command

The host language supported by MIMER/ESQL on UNIX is C.

The output generated by the ESQL preprocessor is intended to be compiled by the host system C compiler. The C compiler defined for the **MIM_CC** symbol in the **/opt/mimer811A/examples/makeopt** file is supported. Other language compilers from other software distributors may or may not be able to compile the ESQL output.

64-bit integers are not supported by MIMER.

5.1.1 ESQL syntax

The syntax for the ESQL command is as follows:

esql language [options] input-file [output-file]

Parameter	Parameter options	Parameter description
language	<i>-c</i>	The language switch, i.e. C, which is the only language currently supported on UNIX.
options	<i>-l(ine)</i> <i>-s(ilent)</i>	Optional: line and/or silent. <i>-l(ine)</i> , generates #line preprocessing directives, which forces the C compiler to produce diagnostic messages with respect to the source code that is <u>input</u> to the pre-processor, rather than the C source that is generated (and subsequently compiled by the C compiler). <i>-s(ilent)</i> , suppresses display of the copyright message.
input-file	<i>file</i>	Name of input file to be processed by ESQL. If the file is specified without a file extension, the default extension is assumed, i.e. ".ec".
output-file	<i>file</i>	Optional: name of output file. If not specified, the output file will be generated with the same name as the input file, but with the default output file extension, i.e. ".c".

6 RUNNING MIMER APPLICATIONS

This chapter describes how to run applications in the MIMER8 environment on UNIX.

Tasks performed when executing MIMER applications

- Select a MIMER installation - if several MIMER installations reside on the same computer it is essential that users access the correct one.
- Select a database server - in situations where several databases exist, users must connect to the right one.
- Execute the MIMER application.

6.1 Selecting a MIMER installation

When MIMER is installed using the `miminstall` command, executables and library files are automatically linked to `/usr/bin` and `/usr/lib` using symbolic links. These symbolic links are created by using the `mimlink` command and can be removed by using the `mimunlink` command.

If the MIMER installation is performed using the suggested defaults, programs and libraries will be located automatically since the `/usr/bin` and `/usr/lib` directories are searched by default. Otherwise the `PATH` environment variable must be updated to include the “bin” directory path for the installation, and the `LD_LIBRARY_PATH` environment variable, or corresponding, must be updated to include the “lib” directory path for the installation. (For platforms not supporting `LD_LIBRARY_PATH`, see the `MIM_SHLIB_ENV` symbol in the `/opt/mimer811A/examples/makeopt` file for the correct environment variable name).

The `MIMER_HOME` environment variable needs to be defined for complete and proper access to the MIMER installation. The default MIMER installation suggests that the `MIMER_HOME` environment variable be defined in the `/etc/profile` file, which gives the definition to all users automatically.

6.1.1 Several installations on one machine

A host computer may have several MIMER installations, of the same and different versions, installed simultaneously. Only one (MIMER8) version can be installed according to the description in the previous section, enabling automatic access.

Access to a specific installation, other than the one installed according to the description in the previous section, is done through user-specific definitions of the environment variables `LD_LIBRARY_PATH`, `MIMER_HOME` and `PATH`.

`LD_LIBRARY_PATH`, or the equivalent for your Unix, is used by the run-time loader when mapping shared libraries.

6.2 Selecting a database server

All MIMER databases are defined in the `/etc/sqlhosts` file. When an application connects to a database, the following choices are made:

- If the application specifies an explicit database name in the `CONNECT SQL` statement (or for `UTIL` and `BSQL`, if the database name is given as a command-line argument), the specified database name will be used.
- If the `CONNECT` statement specifies the “DEFAULT” database (or if a connection is made through a means that does not specify an explicit database name), the environment variable `MIMER_DATABASE` is checked. If the environment variable is defined, the database specified in it is used.
- If `MIMER_DATABASE` is undefined, the database specified in the `DEFAULT` section in the `/etc/sqlhosts` file is used.

6.3 Executing a MIMER application

If the guidelines given in this chapter are followed and the necessary definitions are made, the application can be started.

```
$ my_app1
```

(Execution in single-user mode is described in Appendix A of the *MIMER System Management Handbook*).

6.4 BSQL command line editing

Command line editing is available in the MIMER/BSQL program, which uses a line-oriented interface. The following functions are available:

ctrl-a	Move to beginning of command
ctrl-b	Move backwards in command
ctrl-d	Delete current character
ctrl-e	Move to end of command
ctrl-f	Move forwards in command
ctrl-h	Delete previous character
ctrl-k	Delete after current position in command
ctrl-n	Next command
ctrl-o	Execute retrieved command and get next from history list
ctrl-p	Previous command
ctrl-r	Retrieve command by search condition
ctrl-t	Change place for the previous two characters
ctrl-u	Delete command
ctrl-w	Delete before current position in command
ctrl-<space>	Set mark in command (or “esc <space>”)
ctrl-x ctrl-x	Go to mark set by “ctrl <space>”
ctrl-x ctrl-h	Show the history list
ctrl-x ctrl-r	Retrieve command by history list number
esc h	Delete previous word
esc d	Delete next word
esc b	Move to previous word
esc f	Move to next word

The arrow keys can be used for command retrieval and for positioning the cursor within a line, i.e. the same function as for ctrl-b, ctrl-f, ctrl-n and ctrl-p.

To change the number of commands that can be held in the history list, the environment variable MIMER_HISTLINES can be used (the default is 23).

Note: The operating system may have control sequences set for the terminal that, if they overlap, override those described above. The terminal settings can be listed using the UNIX **stty -a** command.

7 EXAMPLE PROGRAMS

This chapter describes how to build and run example applications provided in the MIMER8 distribution on UNIX.

See chapters 5 and 6 for general information on how to create and execute MIMER based applications.

7.1 Embedded SQL example programs

7.1.1 Simple embedded program, EXAMPLE

The example.ec program is written according to international SQL standards and it should be possible to use it on any standards-compliant database management system without modification.

The program logs in to the default database with the user SYSADM and password SYSADM (change the program as appropriate). The program will then select and print all the rows of the X/Open standard system catalog view INFORMATION_SCHEMA.TABLES, i.e. lists all tables in the system.

The program can be compiled by using the distributed example makefile:

```
$ mkdir example          # Do everything in a sub directory
$ cd example
$ cp /opt/mimer811A/examples/ex_makefile ./makefile
$ cp /opt/mimer811A/examples/example.ec .
```

Update the makefile so that the MYPROG symbol is set to “example”, as the name of the program to be created (avoid trailing spaces).

```
MYPROG = example
```

Then make the example program:

```
$ export MIMER_HOME=/opt/mimer811A
$ make
.
.
.
```

To test the example program, a personal single-user database (including the hotel example data environment) can be created:

```

$ mimsdbgen
SYSADM password: SYSADM
Repeat password: SYSADM
.
.
.
$ export MIMER_DATABASE=SINGLE
$ export MIMER_MODE=SINGLE
$ mimexampldb
SYSADM password: SYSADM
.
.
.
$ ./example
INFORMATION_SCHEMA          COLUMNS          VIEW
INFORMATION_SCHEMA          SERVER_INFO        VIEW
INFORMATION_SCHEMA          SQL_LANGUAGES     VIEW
INFORMATION_SCHEMA          TABLES           VIEW
MIMER                        ACCCOL            BASE TABLE
MIMER                        ACCESS            BASE TABLE
MIMER                        ACCESS_PRIVS      VIEW
MIMER                        BACKUP            BASE TABLE
MIMER                        BACKUPS           VIEW
MIMER                        CODE              BASE TABLE
MIMER                        COLUMN            BASE TABLE
MIMER                        COLUMNS         VIEW
MIMER                        COLUMN_DEFAULTS  VIEW
MIMER                        COLUMN_PRIVS     VIEW
MIMER                        COMMENT          BASE TABLE
MIMER                        COMMENTS        VIEW
.
.
.
$

```

7.1.2 Dynamic SQL program, DSQLSAMP

The dsql.ec file demonstrates how a C program can be constructed, using dynamic SQL syntax conforming to international SQL standards.

The dsql.ec file contains a collection of routines that define a simple but convenient API for dynamic SQL, which can be called from other C programs.

The dsqlsamp.c file contains a program that calls the routines in dsql.ec. The dsqlsamp program allows the user to enter a SQL statement that will be executed directly, similar to the bsql program (see the *MIMER/SQL User's Manual*).

The program can be compiled by using the distributed example makefile:

```
$ mkdir dsq1          # Do everything in a sub directory
$ cd dsq1
$ cp /opt/mimer811A/examples/ex_makefile ./makefile
$ cp /opt/mimer811A/examples/dsql* .
```

Update the makefile so that the MYPROG symbol is set to “dsqlsamp” and the MYFUNCS symbol is set to “dsq1.o” (avoid trailing spaces).

```
MYPROG = dsq1samp
MYFUNCS = dsq1.o
```

Then make the dsqlsamp program and execute it:

```
$ export MIMER_HOME=/opt/mimer811A
$ make
.
.
.
$ ./dsq1samp

*** Welcome to Mimer Dynamic SQL Sample Program ***

Database:hotel
Username:sysadm
Password:5y5adm

- End an SQL statement with ';'
- Exit program by giving an empty command

SQL>select * from mimer.databank;

*** 4 rows selected ***

DBANKID SEQNO DATABANK CREATOR TYPE
===== =====
1      0      SYSDB      MIMER   L
2      0      TRANSDB    MIMER   T
3      0      LOGDB      MIMER   L
4      0      SQLDB      MIMER   W

SQL>

Quit? (y/n) y

*** Exit from Mimer Dynamic SQL Sample Program ***

$
```

7.1.3 BLOB program, BLOBSAMP

The blobsamp.ec file contains a program that demonstrates the use of the VARCHAR data type to store BLOB's (Binary Large Object's). The program copies the contents of any file into the database as a BLOB, it then performs a SELECT to retrieve it again and compares the result with the original file. The file name should be specified to the program as an input parameter and the program assumes that the user has access to a databank where a table can be created.

The program can be compiled by using the distributed example makefile:

```
$ mkdir blobsamp          # Do everything in a sub directory
$ cd blobsamp
$ cp /opt/mimer811A/examples/ex_makefile ./makefile
$ cp /opt/mimer811A/examples/blobsamp.ec .
```

Update the makefile so that the MYPROG symbol is set to "blobsamp" (avoid trailing spaces).

```
MYPROG = blobsamp
```

Then make the blobsamp program and execute it:

```
$ export MIMER_HOME=/opt/mimer811A
$ make
.
.
.
$ ./blobsamp
```

```
Usage: blobsamp username password filename
```

When the blobsamp program is executed without any parameters, the usage text is printed, as can be seen above. The parameters "username" and "password" indicate that a database is presumed to be available and "filename" is the file to be stored in the database. In the example below the mimsdbgen program is used to create a personal single user database for the test. The bsql program will be used to create a databank for the test to be able to create a table for storage. A copy of the /etc/passwd file will be used as the file to be stored and retrieved:

```
$ mimsdbgen
SYSADM password: 5y5adm
Repeat password: 5y5adm
.
.
.
$ export MIMER_MODE=SINGLE
$ export MIMER_DATABASE=SINGLE
```

\$ **bsql -s**

```

MMMMM      MMMMM  MMMMM  MMMMM      MMMMM  MMMMMMMMMMMM  MMMMMMMMM
MMMMMMM    MMMMMM  MMMMM  MMMMMMM  MMMMMM  MMMMMMMMMMMM  MMMMMMMMMMM
MMMMMMM  MMMMMM  MMM    MMMMMM  MMMMMM  MMM  MMM  MMM  MMM
MMMMMMMMMMMMMMMM  MMM  MMMMMMMMMMMMMMMM  MMMMM  MMMMMMMM
MMM  MMMMM  MMM  MMM  MMM  MMMM  MMMM  MMM  MMM  MMM  MMM  MMM
MMMM  MMM  MMMM  MMMMM  MMMM  MMM  MMMM  MMMMMMMMMMMM  MMMM  MMMM
MMMM  M  MMMM  MMMMM  MMMM  M  MMMM  MMMMMMMMMMMM  MMMM  MMMM
    
```

(C) Copyright Sysdeco Mimer AB 1987-1998. All rights reserved.

M I M E R / B S Q L
Version 8.1.1A

Username: **sysadm**
Password: **5y5adm**

SQL>**create databank blob of 10 pages in 'b.dbf' with log option;**
SQL>**exit;**

```

*-----*
!
! Session time   0 00:00:59  !
!
*-----*
    
```

\$ **cp /etc/passwd passwd.copy**
\$ **blobsamp SYSADM 5y5adm passwd.copy**
compare ok!
\$

7.1.4 Programs calling PSM, FREQCALL/WAKECALL

The `wakecall.ec` and `freqcall.ec` files contain program code that demonstrates how to embed calls to stored procedures. The procedures used are included in the example database which must be installed in order to execute the programs successfully, see Section 3.10 of the *MIMER System Management Handbook*.

The programs can be compiled by using the distributed example makefile. The example makefile, as supplied, is structured to make one single main program. Therefore, two separate makefiles are used to simplify this instruction:

```
$ mkdir psmsamp          # Do everything in a sub directory
$ cd psmsamp
$ cp /opt/mimer811A/examples/ex_makefile ./makefile1
$ cp /opt/mimer811A/examples/ex_makefile ./makefile2
$ cp /opt/mimer811A/examples/freqcall.ec .
$ cp /opt/mimer811A/examples/wakecall.ec .
```

Update the makefiles so that the MYPROG symbol is set to “freqcall” in `makefile1` and “wakecall” in `makefile2` (avoid trailing spaces).

```
MYPROG = freqcall
MYPROG = wakecall
```

Then, make the `freqcall` and `wakecall` programs:

```
$ make -f makefile1
.
.
.
$ make -f makefile2
.
.
.
```

Finally, create a personal single-user database for test purposes, including the hotel data environment, and execute the `freqcall` and `wakecall` programs:

```
$ mimsdbgen
SYSADM password: 5y5adm
Repeat password: 5y5adm
.
.
.
$ export MIMER_DATABASE=SINGLE
$ export MIMER_MODE=SINGLE
$ mimexampldb
SYSADM password: 5y5adm
.
.
.
$ ./freqcall
6 available rooms.
$ ./wakecall
0 rooms
$
```

7.2 ODBC example programs

7.2.1 Dynamic SQL program, OSQLSAMP

To be able to compile the ODBC example program, an ODBC Driver Manager SDK is needed. This package should include the necessary header files, such as `sql.h` and `sql.h`, or corresponding.

The `osql.c` file contains a collection of routines that define a simple but convenient API for dynamic SQL, using ODBC according to international standards. These routines can be called from other C programs, in this case `osqlsamp.c`.

The `osqlsamp.c` (same as `dsqlsamp.c`) file contains a program that calls the routines in `osql.c`. The `osqlsamp` program allows the user to enter a SQL statement that will be executed directly, similar to the `bsql` program (see the *MIMER/SQL User's Manual*).

The program can be compiled by using the distributed example makefile:

```
$ mkdir osql          # Do everything in a sub directory
$ cd osql
$ cp /opt/mimer811A/examples/ex_makefile ./makefile
$ cp /opt/mimer811A/examples/osql.c .
$ cp /opt/mimer811A/examples/dsql.h .
$ cp /opt/mimer811A/examples/dsqlsamp.c osqlsamp.c
```

In the following example the ODBC Driver Manager is presumed to be installed under `/usr/local`.

Update the following macros in the makefile: The `INCLUDE` symbol should be updated to include the search path for the ODBC Driver Manager include files. The `MYPROG` symbol should be set to the main program name, in this case “`osqlsamp`”. The `MYFUNCS` symbol should hold the name of the object file(s) for the subroutines, in this case “`osql.o`” (avoid trailing spaces). The `ODBCLIB` symbol should be updated to include the search path for the ODBC Driver Manager library and the name of that library.

```
INCLUDE = -I$(MIMINC) -I. $(EXTEND) $(EXTEND_SHL) -I/usr/local/include
MYPROG = osqlsamp
MYFUNCS = osql.o
ODBCLIB = -L/usr/local/lib -liodbc $(CLIBS) $(LIBC)
```

Note! The names of ODBC Driver Manager libraries and include files may vary between different products. These names must match the makefile and the C-source header include statements, respectively.

Then make the `osqlsamp` program (the environment variable `MIMER_HOME` must be correctly defined for the makefile to work properly):

```
$ export MIMER_HOME=/opt/mimer811A
$ make
.
.
.
```

To be able to execute the `osqlsamp` program the `.odbc.ini` file must be installed in the user HOME directory and updated according to the following example (for details, see documentation for the ODBC Driver Manager in use). The `.odbc.ini` file should include lines as in the following example in order to access a MIMER database:

```
[ODBC Data Sources]
hotel=MIMER database

[hotel]
Driver = /opt/mimer811A/lib/libmimer.so

[default]
Driver = /opt/mimer811A/lib/libmimer.so
Database = hotel
```

Note! If the ODBC Data Source name is not equal to the MIMER database name the “Database” parameter should be used in the `.odbc.ini` file, as shown for the “default” ODBC Data Source above.

Then the `osqlsamp` program can be executed. To succeed with the dynamic library load during execution the `LD_LIBRARY_PATH` environment variable, or corresponding, usually must be set to include the path to the ODBC Driver Manager library.

```
$ export LD_LIBRARY_PATH="/usr/local/lib:$LD_LIBRARY_PATH"
$ osqlsamp
```

```
*** Welcome to Mimer Dynamic SQL Sample Program ***

Data source name [default]: hotel
Username []: sysadm
Password []:
  - End an SQL statement with ';'
  - Exit program by giving an empty command

SQL>select * from mimer.databank;

*** 4 rows selected ***

DBANKID SEQNO DATABANK CREATOR TYPE
===== =====
1         0     SYSDB     MIMER   L
2         0     TRANSDB    MIMER   T
3         0     LOGDB     MIMER   L
4         0     SQLDB     MIMER   W

SQL>

Quit? (y/n) y

*** Exit from Mimer Dynamic SQL Sample Program ***

$
```

To access a database on a different node, i.e. client/server, a database registered in the REMOTE section of the `/etc/sqlhosts` file should be defined as “Database” in the `.odbc.ini` file. No other `.odbc.ini` file options should be used for this purpose.

A DATA TYPES USED IN MIMER

A.1 Internal MIMER representation

MIMER uses only two data representations internally: numeric and character. These two types are used to store all data in the database.

The numeric data type stores numbers in packed BCD format with a maximum precision of 45 digits. Every number stored has an exponent with a range of -999 to +999.

The character data type uses the ISO 8859-1 standard character set, also known as Latin 1. This standard specifies graphic characters and the representation of each character.

On most platforms, including UNIX, the ISO 8859-1 character set is used by MIMER to transfer data to and from the application, i.e. the same character set as is used internally.

Terminals used with a MIMER database should be set up to use the ISO 8859-1 character set.

The ISO 8859-1 character set is presented in Appendix B of the *MIMER/SQL Reference Manual*.

A.2 External data types supported by MIMER

Any value stored in the database may be read into host language variables as described in Appendix A of the *MIMER/SQL Programmer's Manual*. MIMER will perform all the necessary conversions and will signal an error if the value to be converted is not compatible with the destination type.

B USING MIMER7 APPLICATIONS WITH MIMER8

General database upgrade information is provided in the *MIMER Release Notes*. This Appendix describes the specific task of using a MIMER7 application with a MIMER8 database server on UNIX.

If a MIMER7 application is to be used with a MIMER8 database server, there are three main approaches to getting communication established:

- Connect to the database using the client/server interface, i.e. from a client node to a MIMER8 server node.
- Connect to the database using the client/server interface locally, i.e. the client and server nodes are the same.
- Re-link MIMER7 applications to include the shared library for MIMER8 and the compatibility library.

B.1 Client-server access

MIMER7 applications can access a MIMER8 database server through the client/server interface, i.e. from a client node running MIMER7 to a server node running MIMER8, without any modification to the application, licenses, etc.

B.2 Local client/server access

It is possible to use the client/server interface locally, i.e. having a remote database definition pointing to a local database server. This is the recommended access method when using MIMER7 tools, such as `ql` and `pg`, towards a MIMER8 database.

To achieve this, the `/etc/sqlhosts` file must be updated in a specific way using a 6th parameter in the remote definition, as shown in the following example (current node is “startrek”, i.e. where the MIMER8 database “m8server” runs):

```
--
-- =====
DEFAULT:
--
-- Database
-----
SINGLE
-----
LOCAL:
--
-- Database          Path
-----
SINGLE
m8server          /database/m8server:/mnt1/m8server:/mnt2/m8server
-----
REMOTE:
--
-- Database          Node          Protocol Interface Service
-----
example_database    server_nodename  ''          ''          1360
m8access          startrek       ''          ''          mimer      m8server
```

To access the “m8server” database, the MIMER7 application must connect to the database “m8access”.

In this case the “startrek” node must have both a MIMER7 license and a MIMER8 license, where the application depends upon `/mimkey7` and the database server depends upon `/mimkey8`, which must include a specific license in order to be able to accept client/server connections.

B.3 Re-link applications

For optimal performance, the MIMER7 application should be re-linked.

MIMER8 is distributed as a shared library which implies that an application that was previously statically linked must now employ dynamic load. The example makefiles supplied in MIMER7 and MIMER8 provide useful information about loader switches, etc.

For an application that depends upon old low level modules, such as the runtime parts of MIMER/FM, MIMER/SH and MIMER/PG, a compatibility library is provided for inclusion.

In the following example “cc” is used to link a MIMER/PG application from version 7. The shared library libmimer.so and the archive compat.a are taken from a MIMER8 distribution:

```
cc -o pgappl pgappl.o \  
/home/mimer7/lib/pgfml.a \  
/home/mimer7/lib/pgl.a \  
/home/mimer7/lib/lru.a \  
/home/mimer7/lib/fmlib.a \  
/opt/mimer811A/lib/libmimer.so \  
/opt/mimer811A/lib/compat.a -lm -lcurses
```

Now, when referring to the previous section, the “m8server” database can be accessed directly by the re-linked application.

In this case, only a MIMER8 license is needed (/mimkey8), but it must include the necessary licenses for the entire application, e.g. MIMER/PG, MIMER/FM, etc.