

# MIMER

## Installation Guide for UNIX<sup>®</sup>

**Version 7.3**

Copyright

©

1996

Sysdeco

Mimer

AB

MIMER version 7.3 Installation Guide for UNIX

December, 1996

Copyright © 1996 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.se>

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.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	
1.1	Organization of the documentation .....	1-1
1.2	Documentation objectives .....	1-2
1.3	Conventions .....	1-2
1.4	Important terms .....	1-3
1.4.1	MIMER structure terms .....	1-3
1.4.2	MIMER administrative terms .....	1-4
1.5	Acronyms and trademarks .....	1-4
<b>2</b>	<b>THE UNIX ENVIRONMENT</b>	
2.1	The MIMER system setup .....	2-1
2.1.1	Administrative parts .....	2-1
2.1.2	Installation prerequisites .....	2-2
2.2	UNIX users for administration .....	2-3
2.3	Single-user database administration .....	2-3
2.4	End user setup .....	2-3
2.5	Directories and files in the distribution .....	2-4
2.6	UNIX tools needed to install MIMER .....	2-4
2.7	UNIX disk resources needed to install MIMER .....	2-4
2.8	Ownership and access privileges .....	2-5
2.9	Inter Process Communication .....	2-6
2.9.1	Communication levels in MIMER .....	2-6
2.9.2	Example system IPC allocation .....	2-7
2.10	The MIMER servers .....	2-8
2.11	Kernel resources needed by the MIMER multi-user system .....	2-9
2.11.1	Allocation of resources .....	2-9
2.11.2	Example resource calculation .....	2-10
2.11.3	Kernel IPC configuration .....	2-12
2.11.4	Other kernel configuration .....	2-13
2.12	Integration with the UNIX system administration .....	2-14
2.13	Database lookup - /etc/sqlhosts .....	2-14
2.13.1	DEFAULT section in /etc/sqlhosts .....	2-16
2.13.2	LOCAL section in /etc/sqlhosts .....	2-16
2.13.3	REMOTE section in /etc/sqlhosts .....	2-17
2.14	Database network server .....	2-17
2.15	Use of raw devices .....	2-18
2.15.1	Raw device files in MIMER .....	2-18
2.16	Upgrading MIMER .....	2-19
2.17	Parallel MIMER versions .....	2-19
<b>3</b>	<b>INSTALLATION</b>	
3.1	Check the UNIX environment .....	3-1
3.2	Create UNIX users .....	3-1

3.3	Load the distribution .....	3-2
3.4	Install the MIMER system .....	3-2
3.5	Install the MRS key .....	3-3
3.6	Install the /etc/sqlhosts file .....	3-4
3.7	Finishing the installation .....	3-4
<b>4</b>	<b>ADDITIONAL STEPS</b>	
4.1	Re-configuring the UNIX kernel.....	4-1
4.2	Integrating system administration .....	4-2
4.2.1	Automatic startup .....	4-2
4.2.2	Automatic dbcheck .....	4-3
4.2.3	Automatic shutdown.....	4-4
4.3	Increasing priority of the MIMER I/O processes .....	4-4
4.4	Setting up the system as database server .....	4-5
4.4.1	Parallel database servers .....	4-6
4.4.2	Error situations.....	4-6
4.5	Employ use of raw device files.....	4-7
4.5.1	Creating a raw device databank file.....	4-7
4.5.2	Reset use of raw device databank file.....	4-7
4.6	Databank file extend or pre-allocation.....	4-8
<b>5</b>	<b>INSTALLATION SUMMARY</b>	
5.1	Summary of the superuser steps .....	5-1
5.2	Continued installation .....	5-1

# 1 INTRODUCTION

This guide describes the installation of the MIMER relational database system version 7.3 on a computer running the UNIX operating system. This guide is intended for the UNIX system administrator and covers the steps that need to be performed in superuser mode.

Additional steps should be performed by the MIMER system administrator before MIMER can be started. These actions are described in the *MIMER Administration Guide for UNIX*.

## 1.1 Organization of the documentation

The documentation describing installation and maintenance of MIMER on a UNIX system is divided into two guides:

### **MIMER Installation Guide for UNIX:**

This guide is intended to help the UNIX system administrator prepare the UNIX system and install MIMER.

### **MIMER Administration Guide for UNIX:**

This guide supports the MIMER system administrator in configuring and maintaining the MIMER software and database systems.

The following document is a guide for users of the MIMER system:

### **MIMER User's Guide for UNIX:**

This guide is a supplement to the general MIMER reference manuals, and contains UNIX specific information, such as file specifications. This guide is intended for users of a MIMER system under UNIX.

The following document contains machine dependent information:

### **MIMER Release Notes for UNIX:**

This document includes machine-specific comments on the specific implementation of MIMER, such as keypad layouts, and commands specific to that machine.

## 1.2 Documentation objectives

This document describes the installation of the MIMER relational database management system, version 7.3, on computers with the UNIX operating system. It covers those steps which must be performed by the UNIX System Administrator who has superuser privileges or the equivalent. This manual assumes that the installer has a working knowledge of system management in a UNIX environment, and is able to perform the following tasks:

- unpack a distribution received from FTP-site or mount and manage removable media such as D8-cartridge, DAT/DDS data cartridge or cassette streamer, used to distribute the MIMER software
- reconfigure the UNIX kernel according to the guidelines offered here
- create new UNIX users and their home directories.

The information in this document is given mainly in general terms, but in specific areas, e.g. where real life examples are used, the details may differ from your system, relevant information for your computer can be found in the UNIX system administration or configuration guides provided by the hardware supplier (or possibly in the *MIMER Release Notes for UNIX*).

## 1.3 Conventions

The following terms and conventions are used in this guide:

~username	Following the normal csh syntax, the tilde (~) represents the name of the UNIX home directory for the given user name.
environment variable	A shell variable that can be assigned a string value. See your UNIX documentation for a more thorough discussion.
group id	The numerical identification codes used by UNIX to identify a group of users when checking access privileges to files or other UNIX resources.
user id (uid)	The numerical identification codes used to identify a specific login name. The identification number is assigned when the account is created on the particular UNIX system and is used by UNIX when checking access privileges to files or other UNIX resources.
path name	A series of directory names separated by “/” characters and ending in a directory or file name.
absolute path name	A path name beginning with the “/” character, which means “from the root directory”, when used as the first character.

search path	A series of path names which are tried one after the other when looking for a specific file.
superuser	The user who has unrestricted access privileges. This user is normally responsible for performing UNIX administration tasks and has a login name called root.

## 1.4 Important terms

The following terms are important in the discussion presented in this document.

### 1.4.1 MIMER structure terms

database	A database is a collection of tables, databanks, idents, domains, etc. All of these objects are defined in the MIMER data dictionary, SYSDB. A UNIX system may contain several MIMER databases operating simultaneously, each with its own SYSDB, but no information may be shared between different MIMER databases. Each database has a dedicated UNIX user account for administration.
databank	A databank corresponds to one UNIX file or raw device. A databank may contain several tables.
table	Tables (or relations) hold all information in the relational database. A table may not be split over several databanks, but a databank can hold several tables.
shadow	A MIMER databank may have one or several shadows. A shadow is a copy of the original (master) databank and is continuously updated by MIMER/DB. If the master databank is lost, it is possible to continue operations from the shadow databank without stopping the multi-user system.

For other terms such as idents and domains, see the *MIMER System Management Handbook*.

### 1.4.2 MIMER administrative terms

- MIMERADM**    **The MIMER software administrator.** This is a specific UNIX user, created to own and administer the libraries, executable files, data files, etc. distributed with MIMER. Only one user is to be created for this purpose, and, for example, could be given the UNIX user name **mimer7**, which will be used in examples shown in this document.
- ~MIMERADM**    The HOME directory of the MIMERADM user. The installation path **/opt/products/mimer7** will be used in examples shown in this document.
- MULTIADM**    **The MIMER multi-user database administrator.** This is a specific UNIX user, created to own and administer a specific database. One user should be created for each MIMER multi-user system to be installed. Suitable UNIX user names are **multi1**, **multi2** etc. Note that the user name given to the MULTIADM user will become the main database name, although aliases may be created for each database. Also note that this user must not be the MIMERADM user. The names mentioned above will be used in examples shown in this document.
- ~MULTIADM**    The HOME directory of the MULTIADM user. The database paths **/d1/multi1** and **/d2/multi2** will be used in examples shown in this document.

## 1.5 Acronyms and trademarks

- MRS**            MIMER Release and Security.
- IPC**            Inter Process Communication.
- NIS**            Network Information System.
- UNIX**            UNIX is a trademark registered by X/Open Company.

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

## 2 THE UNIX ENVIRONMENT

This Chapter describes the operations which must be carried out before the installation can be performed. These operations can also be performed if the installation needs to be extended in any way, e.g. if a MIMER multi-user system runs out of UNIX system resources.

This Chapter also describes the UNIX kernel resources required by the MIMER system, and gives a brief description of the communication mechanism that uses them.

Other UNIX system facilities involved, such as raw device files, system administration, etc., are also considered.

Chapter 3 actually demonstrates the operations described here in Chapter 2, and Chapter 4 describes and demonstrates additional steps, also described in this Chapter.

### 2.1 The MIMER system setup

#### 2.1.1 Administrative parts

A complete MIMER system basically consists of three main parts, each administered from separate instances, and they must all fit into the UNIX system configuration. These are:

- The installed MIMER software, administered by the MIMERADM user
- MIMER multi-user database system(s), each administered by a MULTIADM user
- MIMER single-user database systems, each administered by any user with access to the tools available in the `~MIMERADM/bin` directory

### 2.1.2 Installation prerequisites

Before the installation begins, decisions and estimates should be made concerning the following (described later on in this Chapter):

- Where to install the distributed MIMER software. Disk requirements and setup (including user name selection) for the corresponding MIMERADM user
- The number of MIMER multi-user systems, i.e. production and test databases, to be installed. Disk requirements, and setup (including user name selection) for the corresponding MULTIADM users
- Disk space requirements and considerations such as which disk partitions to use, security and restrictions (UNIX user groups, etc.)
- Databank storage, i.e. using separate disks for selected databank files to achieve maximum disk-I/O performance (TRANSDB) and optimal data consistency at disk failure (LOGDB)
- Planning, and specifying in the `/etc/sqlhosts` file, the location of local and remote databases
- Integration of the MIMER multi-user systems with the UNIX system administration concerning computer startup/shutdown and backup/restore procedures
- Security, including password management and grouping
- MRS-key management. The number of users authorised to use the MIMER database systems on the site, systems running different MRS-keys, additional modules, etc
- User environment update for MIMER users. Globally, or user specific, definition of environment variables, e.g. PATH, TERM, MIMER\_DATABASE, MIMER\_HOME, MIMER\_KEYFILE and the variable for locating shared libraries
- The number of MIMER single-user systems to be installed (perhaps none). Disk space requirements on HOME partition
- The strategy if this installation is an upgrade of an earlier version of MIMER, e.g. should the previous version of the MIMER system be run in parallel during the installation or not?

## 2.2 UNIX users for administration

The MIMER installation begins with the creation of the administration users, MIMERADM and MULTIADM, within the UNIX environment. These users are required for the proper operation of the MIMER system. Their role is to own and administer different parts of the MIMER system.

These users should only be used for the administration of a MIMER system, and should not be used for other purposes, such as running MIMER applications.

Since anybody can become MIMERADM or MULTIADM by logging on to the corresponding user-accounts, it is very important that these accounts are assigned password, that these password are kept secret and that they are changed regularly. A user logging on as MULTIADM, for example, has all privileges to manipulate and destroy the underlying files of the database owned by the MULTIADM (but not to access the actual data).

## 2.3 Single-user database administration

A MIMER single-user system is owned and administered by any ordinary UNIX user who is permitted to run the **singleinstall** script. It is not necessary to define a separate administration user for single-user MIMER systems. See the *MIMER User's Guide for UNIX* for information on **singleinstall**.

## 2.4 End user setup

Users of a MIMER multi-user system should not be given any special privileges, other than ordinary grouping and access modes. Different environment variables, described in the *MIMER User's Guide for UNIX*, can be set locally for each user or globally for all users logging in to the UNIX operating system.

For security reasons, you may want to place each MULTIADM in a separate UNIX group (i.e. defined in **/etc/group**). Access to a specific multi-user system can then be controlled by allowing group access to the directory containing the multi-user system databank files, e.g. **~MULTIADM/multi**. This specific directory file is used to construct a unique shared key. The key can only be created if the directory file can be accessed by the user, which is why the group access can be used as a "key" to the MIMER multi-user system (see the *MIMER Administration Guide for UNIX* for more details).

## 2.5 Directories and files in the distribution

The root of the distributed directory structure should become the home directory for the UNIX user chosen to be the MIMERADM. The `~MIMERADM` directory will then contain the following sub directories:

<code>~MIMERADM/.adm</code>	Contains scripts and additional files for internal MIMER system administration
<code>~MIMERADM/bin</code>	Contains all the executable programs and scripts
<code>~MIMERADM/dat</code>	Contains examples, documentation, and different data files used and referenced by the MIMER system
<code>~MIMERADM/lib</code>	Contains all the libraries and parameter files
<code>~MIMERADM/modules</code>	Contains a sub-directory for each module, containing the specific files, object code and makefiles, used when creating the executable programs for the module

## 2.6 UNIX tools needed to install MIMER

The MIMER system is distributed with the complete set of executable programs, ready to use.

If these are to be rebuilt for some reason, a C compiler has to be available on the system. Ordinary common UNIX development tools such as a make program and a library archiver will also be needed. See the *MIMER Release Notes for UNIX* for detailed information.

## 2.7 UNIX disk resources needed to install MIMER

Maximum disk space requirements for the `~MIMERADM` directory will vary considerably depending on several factors. For example, a computer whose CPU employs the RISC concept requires more disk space.

Figures for the actual installation resources required initially are provided in the machine specific *MIMER Release Notes for UNIX*.

For the MULTIADM environments the disk consumption depends upon the amount of data to be stored in the database.

## 2.8 Ownership and access privileges

It is intended that the MIMER software can be installed, administered and used within a secure UNIX environment, taking advantage of the safeguards provided by the UNIX system. It is therefore important that the measures below are followed:

- No UNIX users should be left without a password
- The various user home directories should be set up with the proper user IDs and group IDs
- These home directories should not be accessible to inappropriate user IDs who share access to the system

All file attributes and access privileges for the files and directories that reside under `~MIMERADM`, are set up automatically during installation. The ownership and access privileges for the MIMER file structure are very important for the proper operation of the system and should not be changed. Changes to these and the consequences of doing so are not supported.

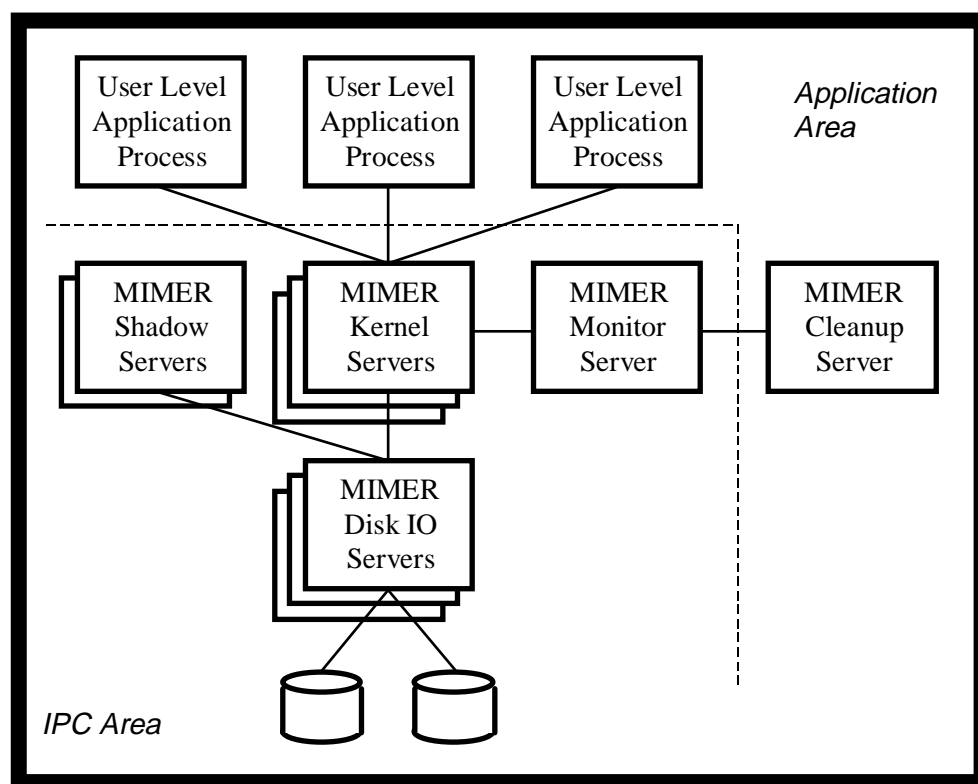
## 2.9 Inter Process Communication

The communication and process synchronization between the communication levels described below is performed by the UNIX SystemV standard primitives for IPC.

### 2.9.1 Communication levels in MIMER

In a multi-user environment, the MIMER/DB software is divided into two parts, with a user level linked separately to each user program and a system level residing in a number of processes.

The illustration below shows the interface between the user program and the MIMER multi-user system:



The code used by the MIMER system level server processes is shared (i.e. re-entrant), so that increasing the number of processes does not result in significantly increased requirements for primary memory.

IPC between user processes and the multi-user system level processes uses the IPC primitives for message queues, shared memory and semaphores.

Data transfer between the MIMER users and the system level processes is done via a shared memory area created by the user process, but is owned by the MULTIADM for that particular MIMER multi-user system.

The MIMER multi-user system bufferpool is a shared memory area created when the multi-user system is started up. This area is also owned by the MULTIADM for that MIMER multi-user system. Access to this area is negotiated among the various MIMER system level processes. No other users may access this area.

The communication between the separate MIMER system level processes is managed with semaphores and a shared memory area owned by the MULTIADM for the actual MIMER multi-user system.

In single-user MIMER installations, all MIMER operations take place in the user process, and no IPC facilities are needed.

### 2.9.2 Example system IPC allocation

The IPC resources a multi-user system allocates can be listed using the `ipcs` command. The indicated owner of an IPC resource can be used to bind it to a specific multi-user system. All IPC resources owned by a specific MULTIADM can be assumed to belong to the multi-user system controlled by that MULTIADM.

The following example shows a multi-user system owned by the MULTIADM **multi1**, having one connected user named **steve**:

```
$ ipcs -c
IPC status from /dev/kmem as of Thu Sep 15 16:01:03 1993
T      ID      KEY          MODE          OWNER        GROUP        CREATOR      CGROUP
Message Queues:
q      0 0x3c3c3284 -Rrw--w--w-   root         root         root         root
q      1 0x3e3c3284 --rw-r--r--   root         root         root         root
q     2853 0x6138501b -Rrw--w--w-   multi1       mimer        multi1       mimer
Shared Memory:
m      13 0x4d453600 --rw-rw-rw-   root         root         multi2       mimer
m     9102 0x5838501b --rw-----   multi1       mimer        multi1       mimer
m     7403 0x5938501b --rw-----   multi1       mimer        multi1       mimer
m     4904 0x00000000 --rw-----   multi1       mimer        multi1       mimer
m     9405 0x00000000 --rw-----   multi1       mimer        steve        mimer
Semaphores:
s      0 0x01090522 --ra-r--r--   root         root         root         root
s      1 0x054baa58 --ra-ra-ra-   root         root         root         root
s      6 0x53453600 --ra-ra-ra-   root         root         multi2       mimer
s     8203 0x7a38501b --ra-----   multi1       mimer        multi1       mimer
s     4004 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4005 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4006 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4007 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4008 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4009 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4010 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4011 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4012 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4013 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4014 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4015 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s     4016 0x00000000 --ra-----   multi1       mimer        multi1       mimer
s    13705 0x00000000 --ra-----   multi1       mimer        steve        mimer
```

The example reveals that the **multi1** MULTIADM owns a message queue with id 2853, four shared memory segments 9102, 7403, 4904 and 9405, and 15 sets of semaphores. Of these IPC resources, the **multi1** multi-user system has created all except the 9405 shared memory and the 13705 semaphore set which was created by user **steve** when he connected to the **multi1** multi-user system.

For MRS reasons the semaphore with id 6 and the shared memory with id 13 was created by the first multi-user system started on the computer, in this case **multi2**.

## 2.10 The MIMER servers

A running MIMER multi-user system with a MULTIADM named **multi1** may look like this (default configuration):

```
$ ps -fmulti1
  UID   PID  PPID  C   STIME   TTY   TIME COMMAND
multi1 13164 13152  0   Aug 31   ?     0:03 mim_kernel 3
multi1 13153 13152  0   Aug 31   ?     0:00 mim_io 0
multi1 13154 13152  0   Aug 31   ?     0:00 mim_io 1
multi1 13166 13152  0   Aug 31   ?     0:00 mim_shadow 1
multi1 13162 13152  0   Aug 31   ?     0:03 mim_io 5
multi1 13157 13152  0   Aug 31   ?     0:00 mim_io 4
multi1 13163 13152  0   Aug 31   ?     0:00 mim_shadow 0
multi1 13165 13152  0   Aug 31   ?     0:03 mim_cleanup 30
multi1 13160 13152  0   Aug 31   ?     0:03 mim_kernel 2
multi1 13159 13152  0   Aug 31   ?     0:03 mim_kernel 1
multi1 13158 13152  0   Aug 31   ?     0:03 mim_kernel 0
multi1 13156 13152  0   Aug 31   ?     0:00 mim_io 3
multi1 13152 13152  0   Aug 31   ?     0:03 mim_monitor 32 30 200
      /dl/multi1/multi 1 6 4 2 0 2000 10 1024 128 32
multi1 13155 13152  0   Aug 31   ?     0:00 mim_io 2
```

The default server configuration for a MIMER multi-user system, as illustrated in the ps-listing above, is suitable for most applications and sites. However, for large installations running hundreds of users, additional servers are likely to be required.

The MIMER servers that must be present in a running MIMER multi-user system are described below:

The **mim\_monitor** server is the parent to all the other servers in the multi-user system. It monitors the existence of the other processes.

The **mim\_kernel** servers receive and service the users database requests. Usually a number of parallel, co-operating copies of the server exist. The number of kernel servers can be altered to improve the responsiveness of the system.

The **mim\_shadow** servers monitor the use of the shared bufferpool, where database pages are held in memory while being read or modified, and initiate output of modified pages depending on the amount of activity in the system. The **mim\_shadow** servers smoothe out the amount of I/O activity over time since a page may be modified a number of times before it is actually physically written to secondary storage. All modifications are immediately secured to the transaction databank during transaction processing. Therefore, database modifications can be reconstructed even if the system fails before modified pages have been written to disk.

The **mim\_io** servers perform all I/O operations on the databank files.

The **mim\_cleanup** server is used for releasing allocated resources in the event of abnormal termination of user processes. This process is activated repeatedly at a time interval chosen when the multi-user system is started.

Response time and overall performance of the database system are usually improved if more than one **mim\_kernel** and **mim\_io** server is started. However, if there is insufficient primary memory available, excessive virtual memory paging may occur if too many servers are started.

## 2.11 Kernel resources needed by the MIMER multi-user system

The kernel has a limited amount of resources. If the resources are exhausted when needed by a running multi-user system, an error message is produced. This can happen if a large number of multi-user systems and users are started, larger than planned for when the MIMER installation was prepared. To avoid repeated problems of this kind, the UNIX kernel should be updated to provide more resources.

For most installations, the UNIX kernel will need to be reconfigured to provide the IPC facilities used by the MIMER multi-user system. As kernel reconfiguration often requires a reboot of the computer with the new kernel, be sure to save the old one (in case the new one does not function correctly). Having an up-to-date full system backup is always recommended when performing system administration changes such as modifying the kernel, especially if it is not changed very often.

### 2.11.1 Allocation of resources

As mentioned earlier, each MIMER multi-user system started requires a quantity of shared memory (below shortened to **shm**), a number of semaphores (**sem**) and one message queue (**msg**), which are all allocated in the UNIX kernel memory space. The following paragraphs describe the existing objects and the resources required for each MIMER multi-user system. These objects are commonly referred to as IPC objects.

Since any given user may start several MIMER user level processes (e.g. use the MIMER "COMMAND" command and start up another MIMER session), you should estimate generously when allocating the kernel limits on shm and sem resources. A good approximation is to allocate twice the number of MIMER user level processes.

The UNIX system ability to keep several simultaneously attached shm segments is a kernel resource parameter that affects the performance at times of peak usage. The limit should be set above the number of presumed concurrent users of a MIMER multi-user system.

The MIMER bufferpool should be set as large as possible for performance reasons, especially in production-critical systems. The amount of memory that can be dedicated for this purpose should be investigated, and the shm size limit must reflect the imposed maximum limit.

Each MIMER multi-user system uses the UNIX kernel message queue feature. The message size is controlled via a parameter defining the number of bytes in a message segment on the queue. To get the MIMER system to work properly, at least eight bytes should be set up, but this value may vary from machine to machine.

The following table describes the IPC objects in a MIMER multi-user system:

Type	Name	Amount	Attached processes	Growth
shm	ccs	500Kbytes	MIMER system level. Monitor-, kernel- and shadow-servers	None. There is always exactly one for each multi- user system started
shm	bpool	*)	MIMER system level. Monitor-, kernel-, Disk I/O- and shadow-servers	None. There is always exactly one for each multi- user system started
shm	ssm	16Kbytes per user	MIMER user level. MIMER system level kernel servers	One for every started user-level process
shm	mrs	1Kbytes	MIMER system level	None. There is always exactly one for the node
sem	enq	2 per server	MIMER system level. Kernel-, monitor-, shadow- and Disk I/O- servers	Approximately two for every started multi-user system level server
sem	ssm	1 per user	MIMER user level. MIMER system level kernel servers	One for every started user-level process
sem	mrs	1	MIMER system level	None. There is always exactly one for the node
msq	ssm	1	MIMER user level. MIMER system level kernel servers	None. There is always exactly one for each multi- user system started

\*) The size depends on the startup figures of the three MIMER bufferpool sections: the 64Kbytes section, the 16Kbytes section and the 2Kbytes section.

### 2.11.2 Example resource calculation

When performing calculations like this, the results should not be used as the optimal values to use. Instead they should be treated as approximate lower limits, and the value finally installed can be set considerably higher. E.g. other software products may have competing requirements on these resources. There is usually no gain in setting kernel limits as low as possible.

Failure to properly consider the resource requirements will probably lead to problems at a later stage, e.g. when the system is put into full production and the requirements for kernel resources reach a new peak.

The following table illustrates a resource calculation for an example installation (the example system configuration used is that listed by the **ps** command in the previous section). The MIMER multi-user system example configuration is:

```

1  monitor server
4  kernel servers
6  Disk I/O servers
2  shadow servers
1  cleanup server
1024 2K bufferpool pages
128 16K bufferpool pages
32 64K bufferpool pages
64 2K Cleanup buffer pages
25  users

```

The cleanup server can be treated as a user application, and as discussed earlier in this section, the number of ordinary users should be doubled, thus the value 51 (1+25+25) is used below.

The IPC used for **mrs** will be included for this system.

The system described above would be using, approximately, the following amounts of IPC resources:

- **Shared memory**

	shm	Bytes	Segments
system level:	ccs	500000	1
	bpool (1024*2048 + 128*16384 + 32*65536 + 64*2048)	6422528	1
	mrs	1024	1
user level:	ssm (16384*51 users)	835584	20
TOTAL:		7759136	23

Conclusion: The largest shared memory segment would fit within 6,5Mbytes. Total amount of shared memory used would be approximately 8Mbytes. The number of shared memory segments used would be less than 25.

- **Semaphores**

	sem	Number of semaphores
system level:	enq (4 kernel 6 Disk I/O 2 shadow 1 monitor)* 2	26
	mrs	1
user level:	ssm (51 users)	51
TOTAL:		78

Conclusion: About 80 semaphores will be required, which also can be used as an approximate figure on number of semaphore sets.

- **Message queues**

One message queue will be used.

The UNIX kernel configuration file should now be updated, and then a new kernel is created.

---

**Note:** Some UNIX implementations extend kernel limits dynamically and other have applications to aid the operation. See the *MIMER Release Notes for UNIX* for details.

---

### 2.11.3 Kernel IPC configuration

There are several parameters within the UNIX kernel that need to be updated to achieve a kernel setup corresponding to the values obtained from the preceding example calculation. The following parameters are generally available for SystemV IPC (for a specific set of configuration parameters on your machine, see the local UNIX system administration (configuration) manuals, or in some cases the machine specific *MIMER Release Notes for UNIX*):

mesg	Message queue feature. 0=disabled; 1=enabled
msgmni	Maximum number of message queues on the system
msgssz	Size of a message segment
sema	Semaphore feature. 0=disabled; 1=enabled
semmni	Maximum number of semaphore sets in the system
semmns	Maximum number of semaphores in the system
semmsl	Maximum number of semaphores per set

semopm	Maximum number of semaphore operations that can be executed per <b>semop()</b> system call. (This parameter is available instead of the <b>semmsl</b> parameter on some UNIX systems)
shmem	Shared memory feature. 0=disabled; 1=enabled
shmall	Maximum space for shared memory
shmmax	Maximum size in bytes of a shared memory segment
shmmni	Maximum number of shared memory segments in the system
shmseg	Maximum number of attached shared memory segments per process

#### 2.11.4 Other kernel configuration

Due to the size of the program in the MIMER/PGD module, the maximum process size will have to be increased in most cases. The value needed is computer dependent and can be determined only when the program has been linked. This figure will be larger for RISC machines. If it is likely that the PGD module is going to be used, an approximate value of 5Mbytes should be set.

Other large modules are the MIMER/SHD module (the SH-editor program) and a MIMER/QL module linked together with RG-, QL-, QF- and SH/FM-interfaces. For these programs the limit must be set to at least 3Mbytes.

Generally the **maxspace** UNIX kernel parameter is available to allow large program modules.

The following UNIX kernel parameters may be tuned for your specific system (for specific configuration parameters on your machine see the local UNIX system configuration manuals, or in some cases the machine specific *MIMER Release Notes for UNIX*):

maxspace	Maximum amount of user address space in bytes
ulimit	File size limit for a process
nproc	Maximum number of active processes in the system
nflocks	Maximum number of file locks in the system
nfiles	Maximum number of open files for a process
max_files	Maximum number of open files in the system

## 2.12 Integration with the UNIX system administration

When a MIMER multi-user system is started it creates a number of UNIX processes as described earlier in this guide. These processes can only be created if the computer is in multi-user mode. If the MIMER processes are running and the computer is switched to single-user mode, the MIMER processes are killed. The databanks which are open at this time will not be closed properly. At the next databank open operation, a databank check will be automatically performed on the contents of these databanks, the time taken for this depends on the size of the databank files involved (a number of large files can take some time to process). It is recommended that you stop the MIMER multi-user system before switching to single-user mode to ensure proper databank closure. The MIMER shutdown procedure can be invoked from a UNIX command script that is run every time the UNIX system is switched to single-user mode (please consult your UNIX system manuals for more information on how your system works).

The MIMER multi-user system can be started again when the computer is returned to multi-user mode. Normally this is done interactively by logging in as the MULTIADM user for each multi-user system and executing the startup menu option, but this can also be done from a script, e.g. in the UNIX system's multi-user mode initialization procedure. Most UNIX systems use a table (**/etc/inittab**) and a script (**/etc/rc**) to initialize multi-user mode processes (please consult your UNIX system manuals for more information on how your system works).

If the database system was brought down incorrectly, the **dbopen** utility should be invoked to open all the databanks defined in the system and check the integrity of any that were not correctly closed. In addition, any pending transactions against a databank will be completed. These operations can take some time, especially if there are large databanks that have not been closed correctly. By running **dbopen** as soon as the multi-user system has been started, these operations can be performed immediately instead of being postponed until the first user opens a databank. The **dbopen** program is further described in the *MIMER System Management Handbook*.

## 2.13 Database lookup - /etc/sqlhosts

For a user to access a MIMER database, the MIMER\_DATABASE environment variable should be defined to point to the name of a particular database, listed in the **/etc/sqlhosts** file. The **/etc/sqlhosts** file lists all databases that are accessible to a MIMER application. Each database has a unique name. Database names may, in general, be up to 64 characters long.

The databases can be either local or remote (remote databases are accessed by using a client/server interface, invisible to the application).

Please make sure that all UNIX users have read access to the **/etc/sqlhosts** file. A UNIX user who cannot read the file will not be able to start any MIMER/DB application.

When MIMER is installed a sample database host lookup file can be found in `~MIMERADM/dat/ex_sqlhosts`. This file can be copied to `/etc/sqlhosts` for further modifications to fit the database environment. The example file looks like the following:

```
-----
--
-- sqlhosts
--
-- This file contains a list of all accessible local and remote
-- databases on a node.
--
-- In a VAX/VMS environment the file is placed in the directory MIMLIB7.
-- On UNIX and PC the file is stored in /etc.
--
-- On VAX/VMS the database can be specified by setting the logical name
-- MIMER_DATABASE to the name of the database. On UNIX and PC an
-- environment symbol MIMER_DATABASE is set to the name of the database.
--
-----
--
-- DEFAULT is the database used if MIMER_DATABASE is not found.
-- Can be a defined local or remote database name.
--
--      DATABASE
--      -----
DEFAULT:
      SINGLE
-----
--
-- LOCAL is a list of all local databases on the current node.
--
--      DATABASE      MULTISYSTEM DIRECTORY
--      -----
LOCAL:
      SINGLE          .
      single          .
--
-- VAX/VMS examples
--
--      DATABASE      MULTISYSTEM DIRECTORY
--      -----
--      PRODUCTION    DISK:<DIRECTORY>
--
-- UNIX examples
--
--      DATABASE      MULTISYSTEM DIRECTORY
--      -----
--      multil         /usr/multil/multi
--      HOTELDB        /usr/multil/multi
--      muldev         /usr/muldev/multi:/opt/db_dir
-----
--
-- REMOTE is a list of all remote databases. The following examples are
-- the same on both VAX/VMS and UNIX. The only difference is that a
-- UNIX system does not support DECNET.
--
-- Interface and service may be defaulted. Default is chosen by entering
-- ''. Default depends on protocol and the machine used.
--
--      DATABASE      NODE      PROTOCOL INTERFACE      SERVICE
--      -----
REMOTE:
--
-- DECNET examples:
--      INTERFACE is currently ignored. $QIO is always used.
--
--      SERVICE corresponds to NCP object or command procedure used by
--      network server.
```

```

--
--      DATABASE          NODE      PROTOCOL INTERFACE      SERVICE
--      -----          -
--      PRODUCTION        ODEN      DECNET   ''             MIMER
--      INVENTORY         INV       DECNET   ''             ''
--
-- TCP/IP examples
-- INTERFACE is the TCP/IP product used on VAX/VMS. On UNIX this
-- signifies which network interface to use, such as socket
-- interface, X/OPENS transport interface (xti) or OSI transport
-- layer interface (tli).
--
-- SERVICE corresponds to the port number used in TCP/IP. The port
-- number may either be a number or a name of a service stored in
-- /etc/services.
--
--      DATABASE          NODE      PROTOCOL INTERFACE      SERVICE
--      -----          -
--      VAX_TEST          oden     tcp      wollongong      mimer
--      DIAB_DEMO         ds90     tcp      tli              1360
--      HP_TEST           hp       tcp      xti              mimer
--      SUN_SERVER        solen    tcp      socket           mimer
--      IBM_DB            rs6000   tcp      ''               mimer
-----

```

### 2.13.1 DEFAULT section in /etc/sqlhosts

The DEFAULT section contains a single line that specifies the default database which should be used by an application that does not explicitly specify a database. Any user may override this default by setting the environment variable MIMER\_DATABASE to translate to the desired database (see the *MIMER User's Guide for UNIX* for more information).

The default database may be either local or remote and a specification for it must appear in the appropriate section of the sqlhosts file.

### 2.13.2 LOCAL section in /etc/sqlhosts

The LOCAL section contains a list of databases that reside on the local machine (i.e. the system databank file, **sysdb7.dbf**, is on a disk on the local machine). Each line under the "LOCAL" keyword should contain two fields, separated by one or more blanks or tab characters. The first field specifies the database name. The second field specifies the search path specification where the databank files for that database can be found. The first pathname in the search path, the primary pathname, must be the **~MULTIADM/multi** directory, which is where the system databank file must reside.

A database name alias can be created by copying the line for an existing database and simply giving it another name (by editing the first field of the line).

The database named SINGLE in the example file has specified a single dot (".") as the directory for the **sysdb7.dbf** file. This allows any user to create his own single-user database environment in his local directory structure. By setting the default directory to where the **sysdb7.dbf** file is, the database can be accessed in single-user mode by specifying the database SINGLE.

### 2.13.3 REMOTE section in /etc/sqlhosts

The REMOTE section lists all databases that are found on other machines. Access to these databases is provided by using TCP/IP to establish a client/server connection to the remote machine. Provided the MIMER client/server installation is setup correctly (see Section 4.5), access to remote machines is performed transparently from the point of view of the application programs.

Both single-user mode and multi-user mode applications may connect to remote databases that are served by multi-user systems on the remote machine.

Each entry in the REMOTE section contains up to five fields, separated by spaces and/or tab characters.

The **database** field specifies the name of the remote database.

The **node** field should specify the TCP/IP node name of the server machine. If no name-server facilities are available, the IP number may be specified here instead.

The **protocol** field should specify "tcp" since TCP/IP should be used to create the client/server connection.

The **interface** field specifies the type of the TCP/IP interface. Currently, only 'socket' is implemented (which also is the default). You can specify " (two single apostrophes) to denote the default.

The **service** field specifies which server port to connect to. Default is "mimer". The TCP/IP port number 1360 has been reserved by Sysdeco Mimer AB for MIMER client/server communication.

The physical location of a remote database must be given in the sqlhosts file on the remote machine. Thus in the preceding example, the sqlhosts file on node **solen** should contain a LOCAL entry for the database named **SUN\_SERVER**.

## 2.14 Database network server

If the machine is to be used as a remote database server, it must be configured accordingly. The network service **mimer** should be added to the system and the inetd facility must be configured to start the MIMER network interface program.

See Section 4.4, or the local UNIX system manuals, for the file formats for **services** and **inetd.conf**.

## 2.15 Use of raw devices

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

Device files like these may be created and used for databank storage to bypass the normal UNIX file system, which uses index pages to locate the file blocks. This will improve file I/O performance on especially large and/or very frequently used databanks. The improvement varies between different UNIX implementations and ranges from no difference in speed at all up to several times faster. Of course the effect also depends on several other factors such as transaction rate, etc.

---

**Note!** It is essential that a raw device file is correctly defined, since the normal UNIX file handling safeguards do not apply. Verify the backup procedures and take a full back-up copy before using a raw device file.

---

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 of the disk before reformatting !*

### 2.15.1 Raw device files in MIMER

The raw device file interface is primarily intended for MIMER multi-user systems in heavy production environments where performance must be the maximum attainable. However, the functionality also works in a single-user environment.

Use of the raw device interface is 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.

It should be noted that if a raw device file partition becomes full, the file cannot be extended. In the worst case, the disk may have to be reformatted in order to provide space for a larger raw device file, i.e. a larger disk partition provided. Under all circumstances the databank file has to be moved to a disk location with space enough to hold the complete partition. This implies that a raw device file should not be used on a production system unless it is verified that the databank will not need to grow beyond the file size limits in the near future.

## 2.16 Upgrading MIMER

When a working MIMER installation is upgraded, the system databank SYSDB (and other databanks) must **not** be re-generated. SYSDB contains the data dictionary for the system, without which the system cannot be used.

See the *MIMER Administration Guide for UNIX* and the *MIMER Release Notes for UNIX* for detailed information when upgrading MIMER.

## 2.17 Parallel MIMER versions

When a new release of MIMER is available it may be convenient to run two MIMER versions in parallel, e.g. if the current version cannot be stopped while the new version is being setup. Parallel versions can be created as long as new administrators (UNIX users) and user paths are installed and maintained correctly for the additional system. Note that the `/etc/sqlhosts` file for database lookup must be updated to reflect the situation.

The use of parallel MIMER versions is not generally recommended except for test and development work in setting up a new MIMER version. Beware of creating data inconsistencies when using parallel copies of a database!



## 3 INSTALLATION

This chapter describes the steps that the superuser should follow to install a MIMER system.

### 3.1 Check the UNIX environment

Once the environment has been set up, the installation procedure can take place. Before loading the software on to the disk, check the disk space available.

Log on as the superuser (root) and check disk space by entering the following:

```
$ su
# df
```

The size of the distributed software is less than 100 Mb.

### 3.2 Create UNIX users

Create the UNIX users needed for the MIMER installation, as described earlier in this document (one MIMERADM and one or more MULTIADMs). In general terms the `/etc/passwd` file should be updated, but some UNIX implementations have their own System Administration utilities for the purpose.

The users should be given the Bourne Shell command interpreter.

Specific `.profile` files will be installed for the users automatically, by the MIMER installation.

Make sure the users have access through the directory tree structure down to their home directories.

Remember to inform the person responsible for the MIMER system about the default passwords for the MIMERADM and MULTIADM users, and where to find the `multiinstall` script, i.e. the `~MIMERADM/bin` path.

The users should be installed in a NIS folder if one exists.

For detailed information on how to create a UNIX user account, see the UNIX documentation for your computer.

### 3.3 Load the distribution

The distribution is in UNIX **tar** format. It can be distributed on removable media, such as streamer, or taken off the FTP-site for example. Usually if the distribution is received over the network it must be uncompressed, using the UNIX **uncompress** command, before installation can take place.

When unpacking, using the **tar** command, a directory named **dist** will be created. This directory will be renamed to the MIMERADM users home directory. If such a home directory already exists, it can be removed (as far as the MIMER installation is concerned). In the following installation example the MIMER software is to be installed under the MIMERADM user **mimer7**, who has been created with the home directory **/opt/products/mimer7**. The distribution read from a tape, via the device **/dev/rmt1**, but it could equally have been directly read from a tar file. The **mimer7** directory already existed in the example below. It could have been created automatically by the System Administration utility when the user was created or it contains an existing MIMER software installation. The following commands are run from the superuser (root):

```
# cd /opt/products
# tar xvf /dev/rmt1
# mv mimer7 mimer7.old_home
# mv dist mimer7
```

At a later stage, e.g. when the new system is verified, the old system can be deleted (if not used):

```
# rm -rf mimer7.old_home
```

### 3.4 Install the MIMER system

To make the MIMER system behave properly, the created environment needs to be initially modified for site-specific characteristics. The following tasks are performed when executing the **.install** script:

- A verification of the distribution is made
- All files in the distribution are given the correct ownership and access mode. Both external integrity and internal availability depend on the proper definition of privileges
- Some files are copied to their official locations
- Terminal definitions are added to the UNIX system terminfo database
- The MIMERADM .profile file is installed.

The script will prompt for the **user** and **group** names set for the MIMERADM user. Run the script from the superuser (root):

```
# cd /opt/products/mimer7
# ./install
```

When executing the script on a previously installed system, the initial state will be reinstated for the tasks discussed in this section.

### 3.5 Install the MRS key

The MRS system is an internal mechanism to enable and disable MIMER modules. Only licensed modules can be executed. The MRS-key also holds information about the total number of simultaneous MIMER users permitted on the actual machine.

The MRS system reads an installation dependent key located in the file **/mimkey7**. The key enables the authorised parts of the installed MIMER system and is provided by the MIMER Agent responsible for the distribution.

If you want to store several license files in the same root file system, you may use alternative file names for the key file by defining the environment variable **MIMER\_KEYFILE** to point at a specific MRS file. This is useful if there are several machines that share the same root file system. If the **MIMER\_KEYFILE** environment variable is undefined or points to an unreadable file, the file **/mimkey7** will be used.

To be able to create your key, the MIMER Agent responsible needs the node name for your machine. Perform the following command:

```
# uname -n
```

Deliver the result exactly as given on screen, and the MRS-key can be generated for your site.

When the MRS-key is received, do the following:

1. Create or edit the **/mimkey7** file using any text editor to enter the key data according to the instructions provided with the key. No line may contain more than 80 characters. No space characters are allowed.

```
# vi /mimkey7
```

2. Make the file readable to all MIMER users.

```
# chmod 444 /mimkey7
```

When the licence is extended to include further modules, new key data will be supplied. Simply overwrite the old data in the key file with the new key.

### 3.6 Install the `/etc/sqlhosts` file

There is an example `sqlhosts` file provided in `~MIMERADM/dat/ex_sqlhosts`. It is recommended that you copy the file to `/etc/sqlhosts` and modify it for your own requirements.

1. Copy the distributed template file example `sqlhosts` file to `/etc/sqlhosts`:  

```
# cp /opt/products/mimer7/dat/ex_sqlhosts /etc/sqlhosts
```
2. Edit the `/etc/sqlhosts` file to reflect the site configuration (this should be done in consultation with the MIMER administrator):

```
# vi /etc/sqlhosts
```

3. Make the file readable to all MIMER users.

```
# chmod 444 /etc/sqlhosts
```

### 3.7 Finishing the installation

Exit from the superuser account and login to the recently created MIMERADM user account to check that a proper login can be performed. A menu will be displayed, where you select "Exit".

If proceeding with the additional steps, described in the next Chapter, login to the superuser (root) account again, and continue.

Otherwise, the person responsible for the MIMER system takes over the installation procedure from here. No other operations need to be done with superuser privilege. The *MIMER Administration Guide for UNIX* describes additional installation steps performed by the MIMERADM and the MULTIADM(s) users.

## 4 ADDITIONAL STEPS

This Chapter describes the additional steps that can be of interest for some MIMER installations on certain occasions, steps that the superuser must perform.

### 4.1 Re-configuring the UNIX kernel

To be able to run a MIMER multi-user system and some of the MIMER modules properly, the UNIX kernel usually has to be updated. When reconfiguring a UNIX kernel, an updated configuration file will be used. Use the information given in the sections concerning the kernel resources needed to locate the system configuration parameters for your computer, and to give them proper values.

The following steps are required, in general terms, to reconfigure a UNIX kernel. For your specific computer see the local UNIX documentation and the machine specific *MIMER Release Notes for UNIX*:

1. Locate the UNIX kernel configuration file and make a copy of it. Modify the UNIX kernel configuration file, by introducing the new kernel parameter settings.
2. Bring the system down to single-user mode.
3. Make a new version of the UNIX kernel, using the edited configuration file. This is performed in a directory that is different from the root directory. You now have a customised UNIX kernel.
4. Save the old working kernel, often called **/unix**. Move the new kernel into the boot location, i.e. to the root directory.
5. Reboot the UNIX system.

## 4.2 Integrating system administration

A MIMER multi-user system can be started and stopped automatically, according to the UNIX system state, by editing the UNIX startup and shutdown scripts.

### 4.2.1 Automatic startup

The start can be arranged by inserting the MIMER multi-user system standalone startup into the **rc** script (or equivalent). The **/etc/rc** script often contains a shell function called **localrc()**, where this kind of operation should be placed; on other systems a specific **/etc/rc.local** may exist for the same purpose.

For example, suppose that there are two MIMER multi-user database systems on the machine, owned by the UNIX users **multi1** and **multi2** respectively. To get those systems started automatically put the following lines into the operating system startup procedure (Bourne shell should be used):

```
echo "Starting the MIMER database multi1"
sh /d1/multi1/bin/dbstartup &
echo "Starting the MIMER database multi2"
sh /d2/multi2/bin/dbstartup &
```

Detailed machine specific information may be found in the *MIMER Release Notes for UNIX*.

### 4.2.2 Automatic dbcheck

As mentioned earlier, all databanks will be automatically checked if the system was brought down incorrectly. The **dbcheck** function is invoked when a databank is accessed the first time after startup. To force all databanks to be checked immediately, when the UNIX system goes into multi-user mode and the MIMER multi-user system is started, the **dbopen** utility should be used. Run **dbopen** immediately after dbstartup and all databanks will be ready to use when a MIMER user logs on. Create a file, named **open\_mimer**, containing the following (the Bourne Shell is used):

```

:
#####
# This open_mimer script will perform dbopen on MIMER
# systems, to initiate required dbcheck operations.
#####
echo `date` > /tmp/dbopenm.log
MIMER_DATABASE=multi1; export MIMER_DATABASE
/opt/products/mimer7/bin/dbopenm <<! >> /tmp/dbopenm.log
username
password
!
MIMER_DATABASE=multi2; export MIMER_DATABASE
/opt/products/mimer7/bin/dbopenm <<! >> /tmp/dbopenm.log
username
password
!
#####
# End of open_mimer script
#####

```

The created command script can be located wherever suitable, e.g. **/usr/local/bin**. It should be run in the background every time the machine is switched to multi-user mode and a MIMER multi-user system is started. Add the following to the UNIX startup file (**/etc/rc**):

```

...
echo "sh /usr/local/bin/open_mimer &" | at now + 2 minutes
...

```

(The delay ensures that MIMER is properly started before dbopen is activated).

The **dbopen** program is an ordinary MIMER multi-user application. It will ask for a MIMER username and a password. Any MIMER user that has **SELECT** privilege on the table **MIMER.DATABANK** may be used for dbopen-login. It is advisable not to use the **SYSADM** user and password for security reasons, which is why a separate user, dedicated for this purpose, should be created and granted the required privileges and no more. See the *MIMER System Management Handbook* for further information.

### 4.2.3 Automatic shutdown

The MIMER multi-user system should be stopped each time the UNIX system enters single user- or system mode. Even if the **dbopen** program helps recover the situation when a multi-user system terminates abnormally, a correct close down of the MIMER multi-user system is preferred, using the standalone multi-user system shutdown command script.

On some systems a sub directory of **/etc**, often called **rc.d**, contains scripts or other executables that should be run at operating system shutdown. To shutdown the MIMER multi-user systems described earlier in the startup example, create the following script file, called **stop\_mimer**, to be put in the **rc.d** directory, or equivalent (Bourne shell is used):

```
:
#####
# This stop_mimer script will close down MIMER
# multi-user systems
#####
echo "Stopping the MIMER database multi1"
sh /d1/multi1/bin/dbshutdown
echo "Stopping the MIMER database multi2"
sh /d2/multi2/bin/dbshutdown
```

Note that the use of **dbshutdown** (**mimstop**) does not guarantee a close down where the databank check at startup is avoided. If users are connected at shutdown they will be forced off the MIMER system, which could lead to a databank check when the system is next started.

Detailed machine-specific information may be found in the *MIMER Release Notes for UNIX*.

## 4.3 Increasing priority of the MIMER I/O processes

To achieve higher performance in a MIMER multi-user database system, the priority of the I/O servers can be increased. This is done by letting these servers run in superuser mode. Do the following:

1. Change directory to the MIMER system bin directory.
 

```
# cd /opt/products/mimer7/bin
```
2. Change owner of the **mim\_io** executable file to root.
 

```
# chown root mim_io
```
3. Set user- and group-ID on execution.
 

```
# chmod 6771 mim_io
```

The multi-user system can now be started and the **mim\_io** servers will run in the high priority superuser mode.

## 4.4 Setting up the system as database server

To set up the MIMER system for networking, the `/etc/services` and `/etc/inetd.conf` files must be updated:

1. Add the following line to the `/etc/services` file, defining the service **mimer** on the port number **1360** (The port number 1360 is reserved by Sysdeco Mimer AB at the Internet Assigned Numbers Authority):

```
mimer 1360/tcp      # MIMER Database Server
```

2. The `/etc/inetd.conf` file need to be updated. The change to this file depends on whether the `netstrvm` program requires that environmental variables are defined in order to work properly, e.g. to locate shared libraries. The machine-specific information can be found in the *MIMER Release Notes for UNIX*.
- 2a. If the `netstrvm` program requires environmental variables, the following method should be used:

Add the following line to the `/etc/inetd.conf` file:

```
mimer stream tcp nowait root /bin/sh sh /etc/s_mimnetstrv7
```

Then create the `/etc/s_mimnetstrv7` file. The following example specifies how the location of shared libraries can be found along the search path defined by `LD_LIBRARY_PATH` and that the installed MIMER software is located in **/opt/products/mimer7**:

```
LD_LIBRARY_PATH="/opt/products/mimer7/lib:/usr/lib:/lib"
export LD_LIBRARY_PATH
/opt/products/mimer7/bin/netstrvm mimer tcp inetd
```

For details on shared library lookup refer to the *MIMER User's Guide for UNIX*.

- 2b. If the `netstrvm` program does **not** require environmental variables, the following method should be used:

Add the following line to the `/etc/inetd.conf` file (the example specifies that the installed MIMER software is located in **/opt/products/mimer7**):

```
mimer stream tcp nowait root /opt/products/mimer7/bin/netstrvm
netstrvm mimer tcp inetd
```

Note that all input must be written on one line, terminated with a return.

3. To employ the new `inetd` configuration, restart the `inetd` daemon. To do this, reboot the system or send the `SIGHUP` signal to the `inetd` process, using the **kill** command. The process id for the `inetd` daemon is found using the **ps** 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
```

The UNIX system is now ready to poll, and start the `netstrvm` program, for remote connections to the MIMER system on the port number defined.

- Verify that the mimer service is listening on the network by running the **netstat** command:

```
# netstat -a | grep mimer
tcp        0      0 *.mimer          *.*          LISTEN
```

#### 4.4.1 Parallel database servers

If the machine supports an older MIMER database server as well as the new one, and they are to be run in parallel, special care must be taken.

First an additional port must be assigned, for the new database server. Then it can be assumed that another MRS-key will be used by the new system, thus the environment variable `MIMER_KEYFILE` must be used. The following steps need to be performed:

- Add the following line to the `/etc/services` file, allocating another port:

```
mimer_2      1361/tcp      # MIMER Database Server
```

- Add the following line to the `/etc/inetd.conf` file:

```
mimer_2 stream tcp nowait root /bin/sh sh /etc/s_mimnetsrv7_2
```

- Then create the `/etc/s_mimnetsrv7_2` file by entering the following lines into it (this example specifies that the new MRS-key resides in the `/mimkey7.newver` file and that the corresponding installed MIMER software is located in `/opt/products/mimer7`):

```
MIMER_KEYFILE=/mimkey7.newver
export MIMER_KEYFILE
/opt/products/mimer7/bin/netsrvm mimer_2 tcp inetd
```

- Set up the `sqlhosts` file on the client to reflect the new port number.

#### 4.4.2 Error situations

If the port definition made in the `inetd.conf` file tries to start a `netsrvm` program of an incompatible version, e.g. `netsrvm` of version 7.2.1 and multi-user system started with version 7.3.1 server programs, the following message will be displayed when the client connects to the port:

```
"Multi-user system not started"
```

If the port definition made in the `inetd.conf` file is set up to use an MRS-key file that denies access to the database server, the following message will be displayed when the client connects to the port:

```
"MIMER/DB fatal error -18502 in function Connect, Handshake message invalid, incompatible protocol"
```

## 4.5 Employ use of raw device files

### 4.5.1 Creating a raw device databank file

When a raw device is to be used for databank storage it must be ensured that no processes other than the ones owned by the MULTIADM user can access the allocated partition.

1. Move the databank file in question, here **frequent.dbf**, to another name. Make sure that the MIMER multi-user system is stopped:

```
# MIMER_DATABASE=multi1; export MIMER_DATABASE
# su multi1 -c "/opt/products/mimer7/bin/mimstat -s"
System directory: /d1/multi1/multi
System state:      stopped
# mv frequent.dbf frequent_ORG.dbf
```

2. Create a soft link to the raw device file that corresponds to the disk partition that is going to be used, here **/dev/rdsk/c0t1d0s0**, and modify the access rights so that the MULTIADM user is the only one that may access both the raw device file and the corresponding regular file device entry to the disk:

```
# ln -s /dev/rdsk/c0t1d0s0 frequent.dbf
# chmod 600 /dev/rdsk/c0t1d0s0
# chmod 600 /dev/dsk/c0t1d0s0
# chown multi1 /dev/rdsk/c0t1d0s0
# chown multi1 /dev/dsk/c0t1d0s0
```

3. Copy the databank file on to the dedicated disk via the link to the specific raw device file called **frequent.dbf**, using the UNIX **cp** command.

```
# cp frequent_ORG.dbf frequent.dbf
```

### 4.5.2 Reset use of raw device databank file

Before removing a raw device file (or a soft link), the databank data should be copied safely to another file, using the UNIX **cp** command.

A soft link to a raw device file, created using the UNIX **ln -s** command, can be removed by the UNIX **unlink** command.

The UNIX system administrator should be informed that the partition is free to use for other purposes. The raw device file under the **/dev** directory should then be "unlocked" (i.e. set to an open mode using the UNIX **chmod** command).

## 4.6 Databank file extend or pre-allocation

In some situations it may be appropriate to pre-allocate a MIMER databank file or to extend an existing MIMER databank file past the global file size limit, set by the UNIX system **ulimit**. The **dbextend** utility can be used to do this. Since superuser privilege is needed to increase the ulimit, this command can only be successfully run by root.

The MIMER databank file is assumed to be a regular file having a size equal to a multiple of 2048 bytes.

Usage:                   **dbextend** *file pages*

Arguments:

<i>file</i>	MIMER databank file name
<i>pages</i>	Number of 2K-pages to add to the file

## 5 INSTALLATION SUMMARY

This chapter is a summary of the installation steps that must be performed with superuser privilege.

### 5.1 Summary of the superuser steps

#### Installation steps:

- Check/create disk space
- Create the MIMERADM and MULTIADM(s) users
- Load the distribution
- Set ownership and access by running the **.install** script
- Install the MRS-key file **/mimkey7**
- Install the **/etc/sqlhosts** file for database connections
- Check that the UNIX kernel is configured

#### Optional steps:

- Re-configure the UNIX kernel
- Integrate the MIMER system with the UNIX system start/stop routines
- Increase the priority of MIMER's I/O-servers
- Add MIMER to the system network configuration
- Install a raw device for MIMER usage
- Pre-allocate or extend a databank file

### 5.2 Continued installation

The continued installation is performed by the person responsible for the MIMER system administration, i.e. not the superuser. The *MIMER Administration Guide for UNIX* describes the installation steps performed by the MIMERADM and the MULTIADM users.