Ingres II HOWTO

Pal Domokos

pal.domokos@usa.net

V1.1, 20 June, 2000

Revision History		
Revision V1.1	20 June, 2000	Revised by: pd
Extended with material on the full version of Ingres II 2.0		
Revision V1.01	23 December, 1999	Revised by: pd
Minor fixes		
Revision V1.0	7 November, 1999	Revised by: pd
Original version		

This document helps install the Ingres II Relational Database Management System on Linux. It covers the setup of both the free Software Development Kit and the full version of Ingres. Further sections try to make it easier to start working with Ingres.

Table of Contents

Table of Contents

1. Introduction

1.1. Copyright

1.2. Disclaimer

To put it briefly: there is no warranty about the validity of any other statement in this document. Read and use at your own risk.

Furthermore, I am not an employee of Computer Associates International and I have no official links with CA. This HOWTO is *not* official documentation.

All copyrights are held by their respective owners. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

1.3. New Versions of the HOWTO

The latest version of this HOWTO can always be found on the Linux Documentation Project's site, in various formats:

- HTML − multiple pages
- HTML − multiple pages: tarred and gzipped
- HTML − single page
- PDF
- PostScript − gzipped
- Text
- SGML (DocBook) source − gzipped

The LDP has many mirrors around the world, as listed on http://www.linuxdoc.org/mirrors.html. Some of these mirrors may be out of date, though. Therefore I suggest you check LDP's primary site for new versions.

HOWTOs are also bundled with most Linux distributions. If you are reading this HOWTO from your Linux

CD, also take a look at LDP's main site to check if a newer version exists.

1.4. Credits

I would like to thank for all the feedback I have received so far. I found especially valuable the contributions of Jorgen Heesche (on forms−based development tools), and Gerhard Hofmann (on the automatic startup and shutdown of Ingres).

Last, but not least, my thanks go to CA for making it possible for me to examine the Ingres II 2.0 Enterprise Edition for Linux.

Naturally, I continue to welcome any comments, criticisms and suggestions. Just email me at `<pal.domokos@usa.net>`.

1.5. Audience

This HOWTO aims to help install Ingres II on (Intel) Linux. As always, help is useful for those who need it and can utilize it as well.

Therefore:

- If you are an Ingres pro familiar with Linux then you do not really need to read this HOWTO. Skim through it though if you have time.
- If you have no previous background in relational database management (experience with at least one real RDBMS, not some dBase−like file management system), you do not know UNIX and have just started using Linux, this HOWTO will not make an easy reading for you. Even then, I do not want to persuade you *not* to try to install and use Ingres. Do not give up easy!

If you are not a novice in database management and have some working knowledge of Linux, this HOWTO is for you! We are not going to discuss the basics of relational database management or SQL in this document, neither are we going to elaborate on how to edit text files on Linux. You can find as much information on these topics as you want in numerous places. This HOWTO is not an Ingres guide, either: the Ingres manuals serve that purpose.

The objective of this HOWTO is that the reader can prepare for, then implement the installation of Ingres II on Linux, through simple and understandable steps. It also gives starting points for basic Ingres system and database administration.

I can only hope that the HOWTO reaches its goal.

2. Ingres

In this section the Ingres II Relational Database Management System is introduced and you come to know how to get it.

2.1. University Ingres and Commercial Ingres

Let us start with an important fact: there are two different types of Ingres. The original one, which was designed and developed in the seventies by a research group led by Michael Stonebraker at the University of California, Berkeley, was the first open source relational database management system: it was free to use and distribute, source code included. In fact, it *is* still free software, although its development stopped in 1989. Its last version (version 8.9) made it into some Linux distributions as well. If you are interested in it, you can download it from, say, the SuSE site. The packages are:

- The main software
- The tools

In 1979, with the foundation of Relational Technology, the career of Commercial Ingres started. Since 1995 it has been distributed by Computer Associates. Its latest version is Ingres II 2.0. This HOWTO deals with the installation of this type of Ingres.

2.2. The Software Development Kit

Ingres, being commercial software, is not free to use. However, CA, like most RDBMS vendors, offers a free version of it (the Software Development Kit) to everyone who is interested in learning Ingres. The SDK has two variants, one for Windows NT and one for Linux. These variants are not quite the same as far as the included components are concerned. Obviously, we are engaged in installing the SDK for Linux here. This contains the following elements:

- Intelligent DBMS: the database engine.
- Internet Commerce Enabled (ICE): Ingres' propriatery CGI solution to connect a database to the Web.
- Enhanced Security: the tool supporting mandatory access control.
- C2 Security Auditing: the possibility of C2 level auditing.
- Terminal Monitors: forms−based and command line SQL interfaces.
- Querying and Reporting Tools: forms−based querying, report−writing and report−running tools plus a forms editor.
- Querying and Reporting Runtime: like the previous one, but without the forms editor.
- Vision Pro: integrated, forms−based development environment with a code generator.
- Embedded SQL Precompilers: precompilers for embedding SQL statements in 3GL applications. Supported languages are: C, C++, COBOL, and Fortran.

You can order a free copy of the Ingres SDK CD on http://www.cai.com/registration/cd_ingres.htm.

Remember that you are not allowed to use the SDK in a business environment. It is for evaluating Ingres and prototyping applications only.

The SDK CD contains both the Windows NT and the Linux versions of the Software Development Kit. You can find the Linux files in the following directories:

- `/doc`: the manuals in PDF format, together with the Linux version of Acrobat Reader (`linux-ar-40.tar.gz`). The installer will not copy the documentation to hard disk. These manuals are also available on http://www.cai.com/products/ingres/documentation_set.htm. I will reference some of them later in this document.
- `/int_lnx`: this directory contains `ingres.tar`, the tarball to be installed. `ingres.tar` can be installed directly from the CD or you can copy it to hard disk first.

Do not forget to read the `Readme` file in the root directory on the CD.

2.3. The Beta Version

The freshest beta version of the SDK can always be downloaded from http://www.cai.com/products/betas/ingres_linux/ingres_linux.htm.

> **Note:** At the time of writing, the version of the downloadable beta is 2.5. The next revision of the HOWTO will cover the installation of this version, too. The 2.0 beta is still available on ftp://ftp.cai.com/pub/marketing/ingres/ingresII9808libc6.tar.

2.4. The Ingres II Full Edition

In February 2000 Computer Associates announced the general availability of Ingres II 2.0 for Linux. Besides the components found in the SDK, the full edition contains more modules, such as:

- Net: this component makes possible for Ingres utilities and user applications to access databases residing on different installations.
- Replicator: support for replication functions.
- Star: for handling distributed databases.
- Enterprise Access: communication with different database management systems and other, non−relational data sources (used to be called Gateways).
- Protocol Bridge: for communicating with clients on different types of networks.
- Spatial Object Library: for handling two−dimensional spatial objects.

The CD, besides the `/doc` and `/int_lnx` directories that are common with the SDK, contains `install.sh`, the general Ingres installer and its files. More on `install.sh` later, in subsection Starting the Installation Program.

2.5. The Unicenter TNG Framework

At last, let me note that the Linux version of CA's Unicenter TNG Framework also includes Ingres as its embedded database management system. For this reason, knowing Ingres may come in handy when using Unicenter, too. You can order a free Unicenter TNG Framework CD on

- http://www.cai.com/registration/tng_framework_linux/index.htm for RedHat, or

* http://www.cai.com/registration/tng_framework_linux/suse_linux.htm for SuSE.

3. System Requirements

In this section you will see what hardware and software requirements must be met before you can install Ingres. The ingres user, owner of the installation, makes a debut, too.

3.1. Hardware

The minimal hardware capable of running Ingres is:

- 486x33 processor, Pentium recommended.
- 16 Mb RAM, with 32 Mb swap space (64 Mb RAM recommended).
- 200 Mb disk space if you install everything (150 Mb will do for the SDK). You do not need to have this space in one file system: we will discuss the possibilities in the section Preparing for the Installation.

  Note: This is the *minimum* recommended configuration. Ingres, like most other RDBMSs, is a fairly resource−hungry application. While your development system will probably run beatifully on a 166 MHz Pentium with 64 Mb RAM, a live system with potentially many concurrent users would require more iron.

3.2. Software

The following software must be present for Ingres to run:

- kernel 2.0.34 or higher.
- libcrypt.so − this library is not included in every Linux distribution. If this is the case with your system, check your distribution's Web site: they must have it somewhere.
- uncompress − certain Linux distributions (such as Caldera's Open Linux 2.2) do not contain the ncompress package. Again, check your distribution's Web site if you do not have it.

Working glibc versions:

glibc	SDK	Full Version
glibc 2.07 (eg RedHat 5.2)	Yes.	No.
glibc 2.1 (eg RedHat 6.0)	Yes but you need the RedHat compatibility packages and an Ingres patch to be able to use the forms−based development tools. See Forms−Based Development Tools for details.	Yes.
glibc 2.1.1, 2.1.2 (eg RedHat 6.1)	No.	Yes.
glibc 2.1.3 (eg RedHat 6.2)	See glibc 2.1.	Yes.

If you are unsure of the version of your glibc, check the `/lib` directory:

```
# ls -l /lib/libc*so
```

The output should be something like:

```
-rwxr-xr-x  ...  /lib/libc-2.1.3.so
```

The version of my glibc is apparently 2.1.3.

> **Note:** There is no guarantee that if your system meets the above requirements you will be able to install Ingres on it. Sticking to a distribution that is explicitly mentioned in the release notes of your Ingres version is the best way to avoid installation problems.

3.3. Kernel Parameters

The default settings of the Linux kernel are adequate for a development Ingres environment. For a live system, however, probably to increase the size of the database cache(s), you may want to change the built–in value of the *SHMMAX* parameter. This parameter sets the maximum size of a shared memory segment. By default, it is 32 Mb which allows for a somewhat lesser buffer cache.

You have two choices to change the value of *SHMMAX*:

As root, simply **echo** the new value into `/proc/sys/kernel/shmmax`:

```
#echo 83886080 > /proc/sys/kernel/shmmax
```

In the example above, we set the value of *SHMMAX* to 80 Mb. The change takes effect immediately but after a reboot, the original value is restored.

The other possibility is to change *SHMMAX*'s default value in the kernel source (the relevant header file is `/usr/src/linux/include/asm/shmparam.h` if you have installed the source). In this case, you may also have to modify other parameters in the file, then rebuild the kernel. I suggest you do it only if you know what you are doing. For information on how to configure and compile the kernel see The Linux Kernel HOWTO by Brian Ward.

3.4. The ingres User and II_SYSTEM

We need an account called ingres to install and run Ingres. He will own the installed software and only he can perform system management tasks such as starting and stopping Ingres.

The ingres user may belong to any group. In the following example, we will create a separate group for him.

The verified (therefore, recommended) shell for the ingres user is **bash**. All examples in this paper apply to this shell. If you use some other shell (which is probably just as fine), take into account the differences in syntax.

The binaries, shared libraries, configuration files and other files which make up the Ingres software, will be located in a tree structure after installation. You will set the root of this tree via the shell variable `II_SYSTEM` in the environment of the ingres user (to be exact, the root directory will be `$II_SYSTEM/ingres`).

If you plan to install the whole software, either the SDK, or the full version, make sure you have the following free space under `$II_SYSTEM/ingres`:

SDK	Full Version
70 Mb	90 Mb

10 Mb extra free space is needed during installation.

> **Tip:** If this is the first time you install Ingres (I hope you start with the SDK, not a live system), I suggest you keep the whole installation (the Ingres software, databases, backups, sort areas, etc.) in one place so that you can find every component easily. If you have at least 150–200 Mb free space under `$II_SYSTEM/ingres` and you do not plan to create large databases (at least, not for some time), your system will work without problems. Should you at any later time run out of space, you will always have the possibility to relocate some of your databases to other partitions.

In the following, I will assume that `II_SYSTEM` is set to `/opt`.

Logging in as root, execute the tasks mentioned above:

```
# useradd -d /opt/ingres -s /bin/bash ingres
# chmod 755 /opt/ingres
# passwd ingres
```

The **useradd** command creates a group with the same name as the new user if you do not specify the group on the command line. It also creates the user's home directory.

We set the home directory of ingres to `/opt/ingres` (`$II_SYSTEM/ingres`). This is not mandatory but convenient.

Finally, append the following lines to the `.bashrc` file of ingres:

```
umask 022
export II_SYSTEM=/opt
export PATH=$II_SYSTEM/ingres/bin:$II_SYSTEM/ingres/utility:$PATH
export LD_LIBRARY_PATH=/lib:/usr/lib:$II_SYSTEM/ingres/lib
export ING_EDIT=/bin/vi
if [ -n "$DISPLAY" ]
then
        export TERM_INGRES=vt100fx
else
        export TERM_INGRES=vt100f
fi
```

ING_EDIT sets the editor that can be called from Ingres utilities or application programs. Naturally, you can use any editor, not just **vi**. You must, however, specify the whole access path to the program. (If you stick to **vi**, check if it is under /bin: it may be somewhere else in your system.)

> **Note:** If the EDITOR shell variable is set, it overrides the value of ING_EDIT.

Setting TERM_INGRES is necessary for the terminal to work properly. Forms−based Ingres utilities, such as the installer itself, and also applications created with traditional Ingres development tools (ABF, Vision) make heavy use of function keys. The .bashrc above sets TERM_INGRES according to the terminal type (X, or VT100−like).

These settings must be included in the .bashrc file of every Ingres user.

4. Preparing for the Installation

This is the longest section and so it should be: after careful planning the installation itself should be an easy task.

4.1. Ingres Environment Variables

You will use Ingres environment variables to determine where to put further elements (besides the software itself) of the Ingres installation. These variables, unlike `II_SYSTEM`, are not shell variables but rather parameters of Ingres stored in a file. Some of them can be changed at any time after the installation, but altering the value of others requires a whole re−install. Later you will see which of them are of this "stable" nature.

During installation, you can choose between setting these variables manually or letting the installer set them to their default values (Express Install option).

In the following, we will take the relevant Ingres environment variables one by one and see what each of them is good for. It may help if you put their planned values on paper. You can find an Installation Worksheet in the *Getting Started Guide* which you can print out and use for this purpose.

4.2. II_LOG_FILE and II_DUAL_LOG

Ingres uses an installation−wide transaction log file to record information on all changes made to any database. This information broadly consists of:

- *Before images* of updated or deleted rows. These are necessary for rolling back uncommitted transactions, should it be required *(undo)*.
- The changes made to database objects. Recording them makes it possible to *redo* committed transactions after a system crash if the new data had not been written to the database prior the crash.

The transaction log resides in the `II_LOG_FILE/ingres/log` directory, where `II_LOG_FILE` is an Ingres environment variable. The name of the log file is `ingres_log`.

Express Install creates a log file of the minimum possible size, 16 Mb. Such a log file may not be large enough even in a development system. If you have free disk space and choose manual install (in which case you can specify the size of the log), set it to something much larger.

Both the location and the size of the log file can be changed at any time after installation. The method of doing this is described in the *System Reference Guide*.

You also have to decide if you want dual logging (mirroring the transaction log). If the log gets corrupted for any reason, Ingres stops and you have to recover your databases from backup. Therefore, in a live system, it is almost compulsory either to have some type of RAID protection of the log or to have it mirrored by Ingres. If you use dual logging, the copy of the log file can be found under `II_DUAL_LOG/ingres/log`. Its name is `dual_log`.

In a development or test environment, mirroring the log is not always necessary.

4.3. Database Locations

There can be any number of databases in an Ingres installation. A database, on the other hand, consists of files of different types. These are:

- Control file: it stores certain basic information about the database. You can see this information with the **infodb** command after you have completed the installation.
- Data files: every system table, user table, and also every index goes in a separate file.
- Checkpoint files: checkpoint is the term Ingres uses for a database backup. A backup can consist of more than file.
- Dump files: online backups are possible in Ingres, that is, the database may be in use while the backup program is running. For this reason, the database may change while it is being checkpointed. Ingres, so that it can restore the database to the state it was in at the *beginning* of the backup, saves the before images of those data blocks (pages) that have changed during the backup process. These pages are saved in the dump files.
- Journal files: from time to time, Ingres writes the records of committed transactions from the log file to journal files (at least, this is the default behavior: journalling may be set off at the database or table level). The frequency of the journalling activity is a tunable function of the amount of information that is written to the transaction log. Journalling protects the installation against media failures: if the disk containing the database crashes, you can restore the last (just before the failure occurred) committed state of the database using a backup (checkpoint) of the database and the journals created after that checkpoint was taken. If you lose the log disk, you can restore the last committed state the database was in at the time the last journal file was written.
- Work files: Ingres, if it needs to sort large volumes of data, creates temporary files on disk.

The files constituting a database reside in different directories, according to their types. These directories are specified indirectly, by means of Ingres *locations*.

There are five location types:

- Data location: place for data files of a database. A database can have more than one data location (adding data locations to a database is called *extending* the database). However, every database has a *primary* data location. The system tables and the control file always reside in the primary location. When creating a table, if you do not specify the location(s) to put it in, it will be placed in the primary data location of the database.
- Checkpoint location: by default, backups are created here. Not necessarily, however: the **ckpdb** command allows you to specify an arbitrary place for the backup, this way you can checkpoint a database directly to tape as well.
- Dump location: for dump files.
- Journal location: this is where the journal files for a database reside.
- Work location: Ingres creates temporary sort files in this location. Just like with data locations, a database may have more than one work location. If this is the case, Ingres, by default, uses all of them for each sort operation.

Let us see how these locations work in practice. Say we have a database, called test, with the following locations:

- `DATALOC1`: data location --> `/opt`

* `CKPLOC`: checkpoint location --> `/opt`
* `DMPLOC`: dump location --> `/opt`
* `JRNLLOC`: journal location --> `/opt`
* `WORKLOC1`: work location --> `/opt`

Every location of the test database points to the `/opt` directory. Elements of the database go in these directories:

* data files: `/opt/ingres/data/default/test`
* checkpoint files: `/opt/ingres/ckp/default/test`
* dump files: `/opt/ingres/dmp/default/test`
* journal files: `/opt/ingres/jnl/default/test`
* temporary files: `/opt/ingres/work/default/test`

Let us suppose now, that we *extend* the database to the following locations:

* `DATALOC2`: data location --> `/opt`
* `DATALOC3`: data location --> `/disk2`
* `WORKLOC2`: work location --> `/disk2`

The database is effectively extended to the following directories:

* data files: `/disk2/ingres/data/default/test`
* temporary files: `/disk2/ingres/work/default/test`

`DATALOC2` points to `/opt`, just like `DATALOC1`. Tables to be created in location `DATALOC2` will go to `/opt/ingres/data/default/test`, the same directory where tables created in location `DATALOC1` reside.

As is apparent from the example, we could have created just one location for `DATALOC1`, `DATALOC2`, `CKPLOC`, `DMPLOC`, `JRNLLOC`, and `WORKLOC1`.

Summarizing the main points about locations:

* Every location points to the root of a directory tree.
* More than one location can point to the same directory.
* A location can be used for storing different types of files.
* Databases can share locations. You can see from the example why this is true: the name of the database becomes part of the directory tree, hence files of different databases never mix.

4.4. The iidbdb Database

Every Ingres installation has a master database called iidbdb. Ingres stores information about users, locations and user databases in this database. iidbdb is created by the installer.

You have to set the locations for iidbdb during installation. These locations are stored in the following Ingres environment variables:

* `II_DATABASE`: data location

- II_CHECKPOINT: checkpoint location
- II_DUMP: dump location
- II_JOURNAL: journal location
- II_WORK: work location

These variables determine the default locations for every user database as well, if you do not override them when creating those databases. See Creating and Destroying Databases for more information.

Warning
Changing the value of II_DATABASE, II_CHECKPOINT, II_DUMP, II_JOURNAL, or II_WORK requires a complete re–install of Ingres.

Let us see these variables one by one.

4.5. II_DATABASE

II_DATABASE determines the data location of iidbdb. Its default value is $II_SYSTEM (in case of a manual install you can enter a different value for II_DATABASE, while Express Install inevitably sets it to $II_SYSTEM).

The size of iidbdb after the installation is somewhat more than 5 Mb. It can only grow significantly if you create hundreds of Ingres users, databases or locations.

4.6. II_CHECKPOINT

II_CHECKPOINT contains the value for the checkpoint location of iidbdb. By default, it is also set to $II_SYSTEM.

The size of a checkpoint is just about the same as that of the database itself (at least until you modify the template file of the checkpoint program: it is possible, as you will see in Backup and Recovery). The installer takes the first checkpoint of iidbdb.

If you plan to place checkpoints of user databases under II_CHECKPOINT then you have to provide for more space here.

A further factor that must be taken into account is how long you want to keep backups. When starting the checkpoint program, you can request the deletion of older backups if you do not have too much free space.

4.7. II_DUMP

II_DUMP determines the dump location of the iidbdb database. By default, its value equals to that of II_CHECKPOINT.

By the end of the installation process, II_DUMP will contain a very small amount of data. If you always create checkpoints off−line then you will not need much space here.

4.8. II_JOURNAL

II_JOURNAL contains the value for the journal location of the iidbdb database. Its default value is the same as II_CHECKPOINT's.

The first checkpoint, taken by the installer causes the first, small journal file to appear here. If you do not use different journal locations for user databases then the necessary amount of free space under II_JOURNAL depends on three factors:

- Whether you want Ingres to journal at all. If you take checkpoints of your databases regularly and do not mind losing the changes you have made to them since the latest checkpoint, you may switch off journalling. Naturally, running a live system without journalling is usually not acceptable.
- If journalling is switched on, then the growth rate of the journal area is determined by the volume of changes made to the databases. Frequent, large updates require quite a bit of space under II_JOURNAL.
- The third factor is, how long you wish to keep old journal files. If, when taking a checkpoint, you instruct **ckpdb** to delete the old checkpoints, then previous journal files will be removed as well.

4.9. II_WORK

II_WORK determines the work location of the iidbdb database. It also defaults to II_CHECKPOINT.

The problem of sizing the work location only arises if II_WORK serves as a work location for user databases as well. It is next to impossible to estimate the temporary disk space that will be needed here; however, having the size of the largest table multiplied by three should work as a starting point.

Remember that a database can have more than one work location. If the original location turns out too small, you can always extend the database to further work locations.

4.10. Other Ingres Environment Variables

Besides the Ingres environment variables that determine locations there are a couple more that you have to set during installation (or have Express Install set them to their default value). These are:

- II_INSTALLATION: a two−character code, identifying the installation. Every Ingres installation on a machine must have its own, unique, installation code. The default value for II_INSTALLATION is II. Once set, it cannot be changed.
- II_NUM_OF_PROCESSORS: number of processors in the machine. By default, it is 1. If you set it to a higher value, Ingres will use spin−locks when accessing the database cache. If you do not know what spin−locks are, do not bother. The point is to set II_NUM_OF_PROCESSORS to 2 if you have a multiprocessor machine. Its value can be changed at any time later.
- II_CHARSET: this variable determines the code set of all character data stored in all databases you will create in the installation. Its default value is ISO88591. Perhaps it is not surprising that changing it to a different value after installation may corrupt data stored in your existing databases.

Since the iidbdb database is created by the installer program, you should not choose Express Install if `ISO88591` does not suit you.

- `II_TIMEZONE_NAME`: name of the time zone, by default `NA-PACIFIC`. During manual install you can select its value from a list of valid codes. Ingres stores all date and time values in GMT and adjusts them according to `II_TIMEZONE_NAME` when communicating with the client. Therefore, if you set `II_TIMEZONE_NAME` to a different value, you will see all date–time values in the databases change. For this reason, set this variable to its final value before creating the first user database.

The (manual) installer prompts you for the value of two further parameters which are not Ingres environment variables:

- Expected number of concurrent users in the system: this is `32` by default. Based on this number, the installer sets the value of a number of other parameters, such as the size of the database cache. These derived parameters can later be adjusted.
- SQL–92 compatible databases: by default, Ingres databases differ from the SQL–92 standard in some ways. For example, object names not protected by single or double quotes are converted to lower case rather than upper case. You can find the other differences in the *SQL Reference Guide*.

After you have made up your mind on the values of all installation parameters, you know whether the default values for those variables that cannot be changed after installation are acceptable to you. If they are, you can choose Express Install.

5. The Installation Process

In this section, the actual installation of Ingres takes place.

5.1. Starting the Installation Program

In the following I will presume that you install directly from the CD which is mounted under `/cdrom`. Depending on whether you install the SDK or the full version of Ingres, you have to start the installation differently.

For the SDK:

1. Log in as *ingres*.
2. **cd** to `$II_SYSTEM/ingres` if it is not your home directory.
3. Unpack the `install` subdirectory from the tar file.
4. Start the **ingbuild** program.

```
$ cd $II_SYSTEM/ingres
$ tar xf /cdrom/int_lnx/ingres.tar install
$ install/ingbuild
```

For the full version:

1. Log in as *root*.
2. Start the installer.

```
# /cdrom/install.sh
```

In this latter case, you have to let the installer know the owner of the installation (ingres), and the value of `II_SYSTEM`. After that, **install.sh** starts **ingbuild** for you.

From this point on, the installation process is the same for both options.

On the starting screen of **ingbuild** you have to specify the path to the tar file and select the type of install: Custom or Package. I suggest you go for Custom Install to be able to choose exactly those elements you want to install.

After choosing Custom Install, a table on the next screen shows all components of your Ingres version together with their respective sizes. Because of common parts in different components, the sizes added up indicate much more disk space than is really needed for the installation.

By default every component is set to be installed. If you want to exclude some of them, write `"No"` in their `"Install?"` field.

You had previously decided if the default values for the "stable" Ingres environment variables were acceptable for your installation. If this is the case, you can choose Express Install here. Remember that you can alter the value of `II_LOG_FILE` as well as the size of the transaction log at any time later.

> **Tip:** If this is your first Ingres install, you have the necessary space under

$II_SYSTEM/ingres and the "stable" parameters' default values are OK, I suggest you choose Express Install.

Therefore, let us see this option first.

5.2. Express Install

In the case of Express Install, the installer executes the following tasks:

* It untars all chosen components from the ingres.tar file to the $II_SYSTEM/ingres/install/tmp directory.
* Checks the integrity of the components.
* Puts the components in appropriate subdirectories under $II_SYSTEM/ingres.
* Sets the Ingres environment variables to their default values.
* Starts Ingres.
* Creates the iidbdb database.
* Takes a checkpoint of iidbdb.
* Stops Ingres.
* Sets up those components that require this (ABF, Enhanced Security, etc).

If the installation process went OK, the program tells you that every installed component is ready to use. In the table on the screen the "Install?" column shows "Ready" for every selected component.

Ingres is installed on your machine. Jump to Completing the Initial Configuration.

5.3. Manual Install

If you choose Install rather than Express Install, the installer untars ingres.tar, checks the integrity of the components and puts them in their respective directories. Then it asks you if you want to setup these components now.

If you decide to do the setup later, the installer stops. In the table containing the components the "Install?" column shows "Not Set Up" for every selected component. You can run **ingbuild** again at any time and choose one of the options Setup All or Setup to set up all or one of the components. A component cannot be used until it has been set up.

Let us see what happens if you choose to set up the components.

First, you have to set up the DBMS server. On the screens to follow you will see a fair amount of explanatory text about the parameters we have covered earlier.

During the setup phase, the installer prompts you for the values of the Ingres environment variables and the other necessary parameters:

* II_INSTALLATION.
* II_DATABASE.
* II_CHECKPOINT: if you set it to the same value as II_DATABASE, the installer warns you of the

dangers of losing a database and its backup at the same time. You have to repeat `II_CHECKPOINT`'s value for the program to accept it.
- `II_JOURNAL`.
- `II_DUMP`.
- `II_WORK`.
- `II_LOG_FILE`: the installer reminds you of Ingres' capability of mirroring the transaction log. Naturally, it only makes sense if the mirrored log file is on a different disk than the primary log file. The installer asks you if you want to *disable* dual logging. Then you have to specify the size of the log (16 Mb by default, make it bigger if you have free disk space as I suggested earlier). After this you have to tell the installer where to put the primary log file, and, if you did not switch off dual logging, the dual log file (`II_DUAL_LOG`).
- `II_NUM_OF_PROCESSORS`.
- `II_TIMEZONE_NAME`.
- `II_CHARSET`.
- Expected number of concurrent users.
- SQL−92 compatible databases.

At every prompt, enter the appropriate parameter's previously decided value.

The installer may also ask you about other components you have chosen to install. Accept the defaults for these.

Full Version
If you requested the installation of Net, make **ingbuild** set it up. Do not bother setting an *installation password*, unless you know what it is: we will complete Net's configuration later, in its own section (Ingres/Net).

5.4. Completing the Initial Configuration

After an Express Install (but perhaps after a manual install as well), you may want to change the values of some of the Ingres environment variables. You will see how to do this here. Stay logged in as ingres.

You can view the current values of Ingres environment variables with the **ingprenv** command:

```
$ ingprenv
```

You can change the value of any variable with the **ingsetenv** command:

```
$ ingsetenv II_TIMEZONE_NAME GMT1
```

In the example we set `II_TIMEZONE_NAME` to `GMT1` (Greenwich Mean Time + 1 hour), which happens to be the time zone Hungary is placed in. You can find all possible values for `II_TIMEZONE_NAME` in the file `$II_SYSTEM/ingres/files/tz.crs` (look for the lists beginning with the word `VALID`).

You can change the value of any other Ingres environment variable in a similar way. **ingprenv** and **ingsetenv** do not require a running Ingres server.

The *System Reference Guide* contains the description of every Ingres environment variable. Let me mention two of those that we have not covered yet.

`II_DATE_FORMAT` determines the display format of dates. By default, its value is `US` which provides the format `dd-mmm-yy`.

> **Note:** The setting of `II_DATE_FORMAT` has no effect on the way dates are stored in the database. Ingres always stores full date values, century included. Hence, you can change the setting of `II_DATE_FORMAT` without the risk of corrupting data. In order to avoid Y2K problems in your applications, you should use a date format that contains the century, such as `MULTINATIONAL4` (`dd/mm/yyyy`) or `FINLAND` (`yyyy-mm-dd`). The latter seems especially proper under Linux :-)

Another Ingres environment variable that has a good chance to be changed from its default value is `II_MONEY_FORMAT`. This one is responsible for how values of money type are displayed.

> **Note:** Just like with dates, the value of `II_MONEY_FORMAT` has no impact on the storage format of money columns.

`II_MONEY_FORMAT` consists of two parts: the first part tells whether the currency sign precedes the amount (`L` = Leading or `T` = Trailing), the second part describes the currency itself (`$`, `DM`, `Ft`, etc.: this part is a string of maximum 4 characters). The two parts are separated by a colon. `II_MONEY_FORMAT` defaults to `L:$`.

Only the ingres user is allowed to use **ingsetenv**, since these Ingres environment variables apply to the whole installation. However, some Ingres environment variables (including `II_DATE_FORMAT` and `II_MONEY_FORMAT`) can be overridden in the users' shell, via Linux variables of the same name. You can check the *System Reference Guide* about which other variables fall into this category.

Warning
Be careful: Ingres will not prevent you from changing the value of any Ingres environment variable, including `II_DATABASE`, `II_CHECKPOINT`, `II_CHARSET`, etc. (the "stable" parameters as we saw earlier). However, if you change one of these, prepare for the nastiest possible consequences, the mildest one of which is that Ingres will not run.

You can find information on how to setup Net and ICE in separate sections (Ingres/Net, and ICE, respectively).

5.5. Re-installation

If you want to re-install Ingres for any reason, remember to do the following first:

- Backup everything you need from `$II_SYSTEM/ingres` (user databases, source code of applications stored there, etc.). Also backup any other databases you want to keep that are stored in different locations. You can use the **unloaddb** utility for creating portable copies of databases. On **unloaddb** see the *System Reference Guide*.
- Stop Ingres.

- Remove everything under `$II_SYSTEM/ingres`. Also remove the contents of every other location where you stored any part of any database.

Warning
Databases that are not completely removed can cause problems when you try to re−create them.

5.6. Command Line Install (SDK)

For installing the SDK, you can run **ingbuild** in batch mode as well. The easiest way to do an Express Install is to start **ingbuild** in the following way (logged in as ingres):

```
$ cd $II_SYSTEM/ingres
$ tar xf /cdrom/int_lnx/ingres.tar install
$ install/ingbuild −express /cdrom/int_lnx/ingres.tar
```

In this case a regular Express Install takes place without having to press another key.

5.7. Client Installation (Full Version)

If you have the full Ingres version, you may want to set up a client installation. If your application will run on a different machine than the database server, all you need on the application server is a client Ingres installation.

For a client install, choose PackageInstall in **ingbuild**, then mark "Ingres Networked Client" to be installed. After the installation has been finished, go to section Ingres/Net to set up Net.

5.8. The Installer's Log

No matter which type of install you have chosen (Express or Manual), you can find all of **ingbuild**'s messages in `$II_SYSTEM/ingres/files/install.log`. I suggest you check this file after an Express Install to see what happened during the installation process. On the other hand, if **ingbuild** stops with an error message, also check this log for possible clues to the cause of the error.

5.9. Checking the Installation

After you have installed and configured Ingres, it is time to check if it works properly. Supposing you are still logged in as ingres, start the Ingres system:

```
$ ingstart
```

Create a new database:

```
$ createdb test
```

Start the command line SQL interface:

```
$ sql test
```

Create a table, insert a row into it and query the table's contents:

```
create table t1 (col1 char(10));
insert into t1 values ('abcde');
select * from t1;
\g
```

If everything went OK, you should see something like the following:

```
$ sql test
INGRES TERMINAL MONITOR Copyright (c) 1981, 1998 Computer Associates Intl, Inc.
Ingres Linux Version II 2.0/9808 (lnx.us5/95)libc6 login
Sun Oct 3 03:43:54 1999

continue
* create table t1 (col1 char(10));
* insert into t1 values ('abcde');
* select * from t1;
* \g
Executing . . .

(1 row)


col1

abcde

(1 row)
continue
*
```

You can leave **sql** with the command **\q**.

6. Basic System and Database Administration

In this section I outline some of the basic tasks of the Ingres system administrator and the Ingres database administrator. You will also see what tools Ingres provides to perform these tasks. In the following I suppose you are logged in as ingres.

6.1. Starting and Stopping Ingres

You have already seen how to start Ingres:

```
$ ingstart
```

To stop Ingres, use the **ingstop** command:

```
$ ingstop
```

ingstop only stops Ingres if the are no active user sessions. If you want to stop the system regardless of user sessions, use the following form:

```
$ ingstop -force
```

In this case, after you have killed Ingres, check if it released all shared memory segments and semaphores it had used:

```
$ ipcs -a
```

If you see shared memory segments or semaphores in **ipcs**'s output that are still attached to the ingres user, release them with Ingres' **ipcclean** utility:

```
$ ipcclean
```

Warning
Take care: forcing Ingres to stop might make your databases inconsistent.

6.2. New Ingres Users and Locations

In order for any user to have access to the Ingres installation, you have to define them as Ingres users with the **accessdb** utility.

Start **accessdb**:

```
$ accessdb
```

Select the Users option, then Create.

Here, enter the name of the user. You do not have to modify permissions.

Save, then End, and End.

You can also use **accessdb** to create new locations, change their types or extend databases to new locations. The usage of **accessdb** is covered in the *System Reference Guide* and in the *Database Administrator's Guide*.

As an alternative to **accessdb**, you can maintain users and locations by running SQL commands on iidbdb (**create user**, **create location**, etc.). The syntax of these commands can be found in the *SQL Reference Guide*.

Warning
Since the ingres user has unlimited power of changing and possibly destroying any element of an Ingres installation, it is highly advisable that you only use this account for carrying out administrative tasks. Create another Linux user and set its environment to that of ingres. Register it as an Ingres user via **accessdb** and use this account for everyday work.

6.3. Creating and Destroying Databases

In subsection Checking the Installation you created a new database. You did not specify any options in the

```
$ createdb test
```

command. Therefore the values stored in `II_DATABASE`, `II_CHECKPOINT`, etc., became locations for the test database. You could have specified each location explicitly:

```
$ createdb test -d<data location> -c<checkpoint location> -j<journal location>
-b<dump location> -w<work location>
```

You can remove a database with the **destroydb** command:

```
$ destroydb test
```

Warning
Be careful, because Ingres will not prompt you before destroying the database.

6.4. Collation Sequences

The collation sequence determines which of any two character strings should be considered less than the other. In Ingres, every database can have its own sort order. You can specify the collation sequence when creating the database:

```
createdb test −lhun
```

If you omit the `−l` parameter, the database will have the default collation sequence which is determined by the implicit sort order of the code set of the Ingres installation (`II_CHARSET`).

If you want to use your own collation sequence (it is `hun` in the example above), you have to create a definition file first. The structure of this file must obey to simple rules by which you specify the absolute or relative ordering of letters and/or strings in your language. This file must then be compiled by the **aducompile** utility for Ingres to be able to use it.

The Spanish collation sequence and the collation based on the DEC Multinational Character Set are available both in source (`spanish.dsc` and `multi.dsc`), and compiled form (`spanish` and `multi`).

You specify these collation sequences in the following way:

```
createdb test −lspanish
```

or

```
createdb test −lmulti
```

The compiled definition files for a collation sequence must be in the `$II_SYSTEM/ingres/file/collation` directory. The syntax rules of the definition files can be found in the *System Reference Guide*. It may also be useful to examine the definition files for the Spanish and the DEC Multinational collations.

6.5. Backup and Recovery

You can back up an Ingres database or certain tables in it with the **ckpdb** utility. The following command backs up the test database:

```
$ ckpdb test
```

> **Note:** Checkpoints can be taken online.

Restoring a database can be done with the **rollforwarddb** command:

```
$ rollforwarddb test
```

By default, **rollforwarddb**, using the latest checkpoint and all journal files created since that checkpoint, restores the database to its last committed state. However, you can specify a point in time to restore the database to the state it was in at that time. You can go back as far as 16 checkpoints (Ingres stores data for the last 16 checkpoints in the control file of the database).

Both **ckpdb** and **rollforwarddb** accept many parameters. You can read more about these commands in the *System Reference Guide*. Besides, you should read Michael Leo's paper on Ingres backup and recovery at http://www.naiua.org/papers/backup99.zip.

Both **ckpdb** and **rollforwarddb** use a template file (`$II_SYSTEM/ingres/files/cktmpl.def`). By modifying this file, you can customize the Linux commands that do the physical backup and restore of the data files. Consult the *Database Administrator's Guide* for the syntax of this file.

6.6. Configuring Ingres

Most Ingres parameters can be set via the **cbf** utility. This is the program by which you can specify the number of DBMS servers, the sizes of different caches and a lot of other variables. The usage of **cbf** is detailed in the *System Reference Guide*.

6.7. Monitoring Ingres

You can use the **ipm** utility to monitor a running Ingres system (Visual DBA only runs on Win32). With **ipm**, you can monitor and manage user sessions, and also the locking and logging subsystems.

6.8. Message Files

The Ingres message files reside in the `$II_SYSTEM/ingres/files` directory. The most important of these is `errlog.log`. Should any problems arise during the running of Ingres, this is the file to check first.

7. Ingres/Net

Ingres/Net is not part of the SDK. You only get it with the full version of Ingres. It allows applications (Ingres utilities and user programs alike) to access Ingres databases on other installations (usually on different machines as well). On the machine where the application runs, a client Ingres installation must be set up. We covered the installation of the client in subsection Client Installation. (Naturally, the client can also be a full Ingres installation.)

In this section you will see how to set up Net on both the client and server to provide remote access to the DBMS server. For a complete description of Ingres/Net I suggest you consult the *Ingres/Net User Guide*.

Before starting with Net, however, we need some information on how Ingres authenticates its users.

7.1. User Authentication

We saw earlier that only valid Ingres users can access an Ingres installation. Ingres keeps information on its users in the iidbdb database. But how does Ingres authenticate users?

In case of local access, the answer is simple: Ingres asks the operating system who the user is.

There is an exception to this rule: certain users may be granted the privilige to *impersonate* other Ingres users when starting an Ingres utility or application. That is why it is not necessary for every Ingres user to have an OS account. This privilege, however, can only be granted as all−or−none: if you give it to somebody, he/she will be able to impersonate *any* other Ingres user, including the ingres account. Therefore, never grant it to anyone.

Leaving the authentication of users to the operating system works fine for local access. But what about users who want to use the database from a remote machine? They do not log in to the machine the database resides on (the *server*), therefore the server's operating system will not authenticate them (they may not even have an OS account on the server machine).

There are two possible ways Ingres can authenticate these users. We will cover them in the next two subsections.

7.2. Login Account Passwords

The first solution to the remote user authentication problem is to require that the client provides a local (to the server machine) user name and password. Then the Ingres server authenticates these through standard OS facilities, just like the operating system would do with real local accounts.

In this case, you do not have to set anything in Net on the server. The only thing you will need is the **ingvalidpw** Ingres utility. It will check (by using the `getspnam` and `crypt` OS functions) if the user's name and password are valid on the server machine. On how to install **ingvalidpw**, see subsection ingvalidpw.

7.3. Installation Passwords

The other way of authenticating remote users is that the server accepts their user ID *on the client machine*. In this case, the remote users do not have to be known to the OS on the server.

How will the server validate clients in this case? It is obvious that we need some kind of authentication: anybody can create an ingres account on a client machine, then he/she could connect to the installation as the ingres super–user.

This is where the *installation password* comes in: you set an installation password on the server. You then set this password on the client machines *for those accounts* that you want to allow to access the server under their name on the client.

The Ingres server can then authenticate the client by simply checking its installation password.

7.4. ingvalidpw

As **ingbuild** apparently does not bother installing **ingvalidpw**, you have to build it yourself.

Login as root, set the environment as that of ingres, then simply type

```
# mkvalidpw
```

This script builds and installs **ingvalidpw**.

7.5. Setting up the Client

You will use the **netutil** utility to set up Net on the client side, and, in the case of installation passwords, also on the server. Let us see the client side first. Log in as the account you want to grant access to, or ingres, if you want to set up general access. Then type:

```
$ netutil
```

You can see three tables on **netutil**'s screen. Let us see what fields each of them contains:

- Virtual Node Name: this is the name by which you identify the remote Ingres installation, similarly as you would define an ODBC data source name. The name is of local scope and has nothing to do either with the server machine's name or the remote installation's code.
- Login/Password Data: one or two entries of the following:
    - Type: can be `Global`, or `Private`. If `Private`, the entry will only pertain to the current account. If `Global`, it will be used for all users on the client machine, except for those with a `Private` entry.
    - Login: the user account on the server machine. In case of an installation password, it should be `*`.

    - Password: the password on the server machine (the above user's password, or the installation password).
- Connection Data: at least one entry of the following:
    - Type: can be `Global`, or `Private`. The same applies as in Login/Password Data.
    - Network Address: the server machine's address.
    - Protocol: the network protocol. On Linux, it is probably `tcp_ip`.
    - Listen Address: listen address of the communication server as set up by **cbf**. By default, it is the same as the installation code.

7.6. Setting up the Server

If you want to use an installation password, you have to configure Net on the server, too. In **netutil**, create a virtual node with the following data:

- Virtual Node Name: must be the machine's name.
- Login/Password Data
    - Type: `Global`.
    - Login: `*`.
    - Password: enter the installation password.
- Connection Data: you do not have to enter any data here.

7.7. Using Net

After you have configured Net with **netutil** on the client, and, if necessary, on the server, use **netutil**'s Test menu option to see if the connection works. If it does, you can access a remote database in the following manner (let us suppose the name of the database is test, the virtual node name for the remote Ingres installation is ingserv1):

```
$ sql ingserv1::test
```

8. ICE (Internet Commerce Enabled)

ICE is Ingres' proprietary gateway to the Web. Basically, it is a CGI program that can talk to an Ingres server through the native Ingres API. ICE supports a couple of macro commands which you can embed in HTML documents. When rendering such a document, ICE first executes the macros then outputs the resulting web page.

On other platforms you can configure ICE as a server extension to the Spyglass web server which is bundled with Ingres. The Linux version of Ingres does not include Spyglass. Therefore, in this section I will show you how to setup ICE as a standalone CGI program under Apache, the world's most popular web server.

You need the **ingvalidpw** program for ICE to work. See subsection ingvalidpw on how to install it.

8.1. Configuring Apache

Building, installing and configuring Apache is beyond the scope of this HOWTO. (You had better learn Apache if you are putting your databases on the Web, with ICE or otherwise.)

I suggest to download the newest stable version of Apache in source and build it yourself for maximum flexibility. I also suggest you keep a separate Apache installation just for ICE.

In this subsection I will only cover those parameters of Apache that are important from ICE's point of view.

Things to watch out for:

- The installed software should be owned by the ingres user. This is not strictly necessary but will make things easier.
- Compile the mod_env module into the server, preferably statically (do not use DSOs unless you have to: they make Apache slower).

After you have compiled and installed Apache, set the following parameters in `httpd.conf`:

```
Port 8000 -- must be greater than 1023
User ingres -- all server processes run as ingres
Group ingres -- the ingres user's group
PassEnv II_SYSTEM
PassEnv LD_LIBRARY_PATH
```

The last two lines must be added to `httpd.conf`. These variables will be passed from the environment of the ingres user to the environment of CGI programs started by Apache (specifically **iceinst** and **ice**, the two executables of ICE).

8.2. ICE Setup

Now you can configure ICE and its Tutorials. You can do this with a browser and the **iceinst** program. Let us suppose that your CGI directory is `/opt/ingres/apache/cgi-bin` and Apache is listening on port 8000. Let the name of your machine be ingserv1. Then you can start **iceinst** in the following manner:

```
$ iceinst -d/opt/ingres/apache/cgi-bin -u/cgi-bin -shttp://ingserv1:8000
-b/opt/netscape/netscape
```

Option `-d` is the full path to the CGI directory, `-u` is this directory's address within the site, `-s` is the Internet address of the server, while `-b` is the full path to the browser. If you omit option `-b` and write `-remote` instead, then **iceinst** will not try to start the browser. You can configure ICE from another machine then, directing your browser to http://ingserv1/cgi–bin/iceinst.

First the program asks for the value of `II_SYSTEM`. Then you should visit every screen and set all parameters presented on them. Have **iceinst** install the *Dynamic SQL Tutorial* and the *Macro Processor Tutorial* as well. These show the usage of ICE via applications and a database (icedb by default).

It is important to create a directory under Apache's `DocumentRoot` where ICE can store the output it creates for clients' requests. ICE will not start until you create this directory and specify its name in **iceinst**.

After you have completed every form, choose the Install option. If you have set everything properly, the configuration of ICE and the installation of the tutorials take place. ICE is ready to use.

9. Miscellaneous Topics

Further hints to the use of Ingres.

9.1. Automatic Startup and Shutdown

If you want Ingres to start automatically whenever Linux boots and stop when you shutdown or reboot the system, do the following:

Log in as root.

Check if your Linux variant has System V or BSD style **init** (**init**'s **man** page will tell that).

If your system conforms to System V, the `/etc/rc.d/init.d` directory must exist. Create a file there (call it **ingres** or any other name you wish). The file should contain at least the following:

```
#!/bin/sh

case $1 in
  start)
    echo "Starting Ingres"
    su - ingres -c "ingstart"
    ;;

  stop)
    echo "Stopping Ingres"
    su - ingres -c "ingstop"
    ;;

  *)
    echo "Usage: ingres {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Link the file as `K01ingres` to the directories that correspond to the run levels in which Ingres should *stop*:

```
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc0.d/K01ingres
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc1.d/K01ingres
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc6.d/K01ingres
```

Also link it as `S99ingres` to the directories that correspond to the run levels in which Ingres should *start*:

```
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc2.d/S99ingres
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc3.d/S99ingres
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc4.d/S99ingres
# ln -s /etc/rc.d/init.d/ingres /etc/rc.d/rc5.d/S99ingres
```

It is not important to call the links `K01ingres` and `S99ingres`, the point is that the name starting with

`K` should contain a small number (so that Ingres stops early when changing to a lower runlevel) and the name starting with `S` should contain a large number (so that Ingres starts after everything else has started). Naturally, the file names must not clash with names of existing files.

If you have a BSD style **init**, put the following lines into `/etc/rc.d/rc.local`:

```
echo "Starting Ingres"
su - ingres -c "ingstart"
```

This will start Ingres. (As a matter of fact, you can use `/etc/rc.d/rc.local` even if you have a System V style **init**.)

To stop Ingres automatically, create a file in `/etc/shutdown.d` (call it, say, **ingres**) that contains the commands:

```
echo "Stopping Ingres"
su - ingres -c "ingstop"
```

No matter which type your system is, the files you create must be executable files, owned by root.

Naturally, if your system provides a utility for configuring programs to start and stop automatically (such as **chkconfig** in RedHat), use that if you wish.

9.2. ingmenu

The easiest way to access an Ingres database (at least, for beginners) is via the **ingmenu** program. From **ingmenu**, you can reach Ingres' forms−based utilities, by which you can create, update and query tables, create, edit and run reports and ABF or Vision applications. Its usage is:

```
$ ingmenu test
```

Test is the name of the database.

9.3. Circumventing Ingres Net

Without Ingres/Net, in theory it is not possible for Ingres applications to access databases on different machines. However, there exists a method, not supported by CA, by which sometimes you can come around this problem.

Let us suppose your application runs on host ingdev and the database (called test) you would like to update or query resides on host ingserv. Your first task is to find out the port number of the appropriate DBMS server running on ingserv. You can use **ipm** for this purpose: as ingres, start **ipm** on ingserv and choose option Server List. In the list of servers select one that is of type `INGRES` and handles the test database (you have to see either `test` or `ALL` in column `Connecting to Databases`). You find the port number of the DBMS server in the first column. Let us suppose it is 1259.

On machine ingdev, set the shell variable `II_DBMS_SERVER` in the following way:

```
$ export II_DBMS_SERVER='ingserv::1259'
```

Now run the command:

```
$ sql test
```

If it works, you have access to the test database on host ingserv.

This solution is applicable only if both machines are of the same architecture, the same operating system is running on both of them, the character set is the same in both Ingres installations, and so on: I do not know the full list of necessary conditions. Therefore, I cannot guarantee that this trick will work.

On the other hand, if you restart Ingres on host ingserv, the DBMS server process will get a different TCP/IP port, therefore you probably have to automate the fetching of the current port number to the application server. You can use the **show** command of the **iinamu** utility for this purpose. The following command line gives the port number of the DBMS server if there is only one server running:

```
$ echo show | iinamu | grep INGRES | tr -s ' ' '\t' | cut -f4
```

9.4. Forms−Based Development Tools

The Ingres installation includes a sample application, created by ABF, the traditional development tool of Ingres. You can load it with the **abfdemo** command. Unfortunately, the manuals of ABF and Vision cannot be found either on the Ingres CD or on the CA site.

There is a problem with the SDK under glibc 2.1: applications created by ABF or Vision cannot be either compiled or run directly from the database. This problem is solved in the full Ingres version. For the SDK, install the RedHat glibc 2.0 compatibility packages. If you do not have RedHat, download them from the following URLs:

- ftp://ftp.redhat.com/pub/redhat/redhat−6.0/i386/RedHat/RPMS/compat−binutils−5.2−2.9.1.0.23.1.i386.rpm
- ftp://ftp.redhat.com/pub/redhat/redhat−6.0/i386/RedHat/RPMS/compat−egcs−5.2−1.0.3a.1.i386.rpm
- ftp://ftp.redhat.com/pub/redhat/redhat−6.0/i386/RedHat/RPMS/compat−glibc−5.2−2.0.7.1.i386.rpm
- ftp://ftp.redhat.com/pub/redhat/redhat−6.0/i386/RedHat/RPMS/compat−libs−5.2−1.i386.rpm

Besides the compatibility packages, you need an Ingres patch. It was posted on the Ingres newsgroup in September, 1999. I have a copy of it, email me if you wish to install it.

The compatibility packages and the patch probably do not work for all Linux distributions. I only tested them on RedHat and Caldera Open Linux.

9.5. Ingperl and Perl DBI

Previous Perl versions, version 4 included, made Ingres access possible via libraries known as ingperl. You can find information on ingperl at http://www.contrib.andrew.cmu.edu/~lfm/ingperl.html.

In Perl 5 a new, unified database interface, called Perl DBI, appeared. Its site is http://www.symbolstone.org/technology/perl/DBI/index.html.

You can download the Ingres module of DBI from that site.

9.6. Ingres links

I leave you with a few pointers to important Ingres sites:

- http://www.cai.com/ingres/: home page of the Ingres RDBMS on CA's site.
- http://support.cai.com/ingressupp.html: Ingres Technical Support.
- http://www.cai.com/ingres/inquire/: inquire_ingres: Ingres technical magazine.
- http://www.naiua.org: the North American Ingres Users Association's site. Check the FAQ page, and the `/papers` directory.
- news:comp.databases.ingres: the Ingres newsgroup.
- http://www.deja.com/group/comp.databases.ingres: the archived Ingres newsgroup on Deja.
- http://munkora.cs.mu.oz.au/~yuan/Ingres/ingres.html: William Yuan's Ingres Reference Page with lots of Ingres information.
- http://www.mercurie.co.uk/ingres/: Prijesh Patel's Unofficial Ingres Web Page with edited posts from the Ingres newsgroup.

Have fun!