

MySQL and Linux/Unix

MySQL and Linux/Unix

Abstract

This is the MySQL Linux extract from the MySQL 5.5 Reference Manual.

Document generated on: 2012-11-14 (revision: 33073)

Table of Contents

Preface and Legal Notices	v
1. Installing MySQL from Generic Binaries on Unix/Linux	1
2. Installing MySQL on Linux	5
2.1. Installing MySQL from RPM Packages on Linux	5
2.2. Installing MySQL on Linux using Native Package Manager	9
3. Installing MySQL on Solaris and OpenSolaris	15
3.1. Installing MySQL on Solaris using a Solaris <code>PKG</code>	16
3.2. Installing MySQL on OpenSolaris using IPS	16
4. Installing MySQL on FreeBSD	19
5. Unix Postinstallation Procedures	21
5.1. Problems Running <code>mysql_install_db</code>	25
5.2. Starting and Stopping MySQL Automatically	27
5.3. Starting and Troubleshooting the MySQL Server	30

Preface and Legal Notices

This is the MySQL Linux extract from the MySQL 5.5 Reference Manual.

Legal Notices

Copyright © 1997, 2012, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. MySQL is a trademark of Oracle Corporation and/or its affiliates, and shall not be used without Oracle's express written authorization. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle or as specifically provided below. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

For more information on the terms of this license, or for details on how the MySQL documentation is built and produced, please visit [MySQL Contact & Questions](#).

For additional licensing information, including licenses for third-party libraries used by MySQL products, see [Preface and Legal Notices](#).

For help with using MySQL, please visit either the [MySQL Forums](#) or [MySQL Mailing Lists](#) where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the [MySQL Documentation Library](#).

Chapter 1. Installing MySQL from Generic Binaries on Unix/Linux

Oracle provides a set of binary distributions of MySQL. These include binary distributions in the form of compressed `tar` files (files with a `.tar.gz` extension) for a number of platforms, as well as binaries in platform-specific package formats for selected platforms.

This section covers the installation of MySQL from a compressed `tar` file binary distribution. For other platform-specific package formats, see the other platform-specific sections. For example, for Windows distributions, see [Installing MySQL on Microsoft Windows](#).

To obtain MySQL, see [How to Get MySQL](#).

MySQL compressed `tar` file binary distributions have names of the form `mysql-VERSION-OS.tar.gz`, where `VERSION` is a number (for example, `5.5.30`), and `OS` indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

To install MySQL from a compressed `tar` file binary distribution, your system must have GNU `gunzip` to uncompress the distribution and a reasonable `tar` to unpack it. If your `tar` program supports the `z` option, it can both uncompress and unpack the file.

GNU `tar` is known to work. The standard `tar` provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU `tar`, or if available, use a preinstalled version of GNU `tar`. Usually this is available as `gnutar`, `gtar`, or as `tar` within a GNU or Free Software directory, such as `/usr/sfw/bin` or `/usr/local/bin`. GNU `tar` is available from <http://www.gnu.org/software/tar/>.

Warning

If you have previously installed MySQL using your operating system native package management system, such as `yum` or `apt-get`, you may experience problems installing using a native binary. Make sure your previous MySQL previous installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check the existence of configuration files such as `/etc/my.cnf` or the `/etc/mysql` directory have been deleted.

If you run into problems and need to file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

On Unix, to install a compressed `tar` file binary distribution, unpack it at the installation location you choose (typically `/usr/local/mysql`). This creates the directories shown in the following table.

Table 1.1. MySQL Installation Layout for Generic Unix/Linux Binary Package

Directory	Contents of Directory
<code>bin</code>	Client programs and the <code>mysqld</code> server
<code>data</code>	Log files, databases
<code>docs</code>	Manual in Info format
<code>man</code>	Unix manual pages
<code>include</code>	Include (header) files
<code>lib</code>	Libraries

Directory	Contents of Directory
<code>scripts</code>	<code>mysql_install_db</code>
<code>share</code>	Miscellaneous support files, including error messages, sample configuration files, SQL for database installation
<code>sql-bench</code>	Benchmarks

Debug versions of the `mysqld` binary are available as `mysqld-debug`. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. For more information on compiling from source, see [Installing MySQL from Source](#).

To install and use a MySQL binary distribution, the basic command sequence looks like this:

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

A more detailed version of the preceding description for installing a binary distribution follows.

Note

This procedure assumes that you have `root` (administrator) access to your system. Alternatively, you can prefix each command using the `sudo` (Linux) or `pfexec` (OpenSolaris) command.

The procedure does not set up any passwords for MySQL accounts. After following the procedure, proceed to [Postinstallation Setup and Testing](#).

Create a `mysql` User and Group

If your system does not already have a user and group for `mysqld` to run as, you may need to create one. The following commands add the `mysql` group and the `mysql` user. You might want to call the user and group something else instead of `mysql`. If so, substitute the appropriate name in the following instructions. The syntax for `useradd` and `groupadd` may differ slightly on different versions of Unix, or they may have different names such as `adduser` and `addgroup`.

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
```

Note

Because the user is required only for ownership purposes, not login purposes, the `useradd` command uses the `-r` option to create a user that does not have login permissions to your server host. Omit this option to permit logins for the user (or if your `useradd` does not support the option).

Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it. The example here unpacks the distribution under `/usr/local`. The instructions, therefore, assume that you have permission to create files and directories in `/usr/local`. If that directory is protected, you must perform the installation as `root`.

```
shell> cd /usr/local
```

Obtain a distribution file using the instructions in [How to Get MySQL](#). For a given release, binary distributions for all platforms are built from the same MySQL source distribution.

Unpack the distribution, which creates the installation directory. Then create a symbolic link to that directory. `tar` can uncompress and unpack the distribution if it has `z` option support:

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
```

The `tar` command creates a directory named `mysql-VERSION-OS`. The `ln` command makes a symbolic link to that directory. This enables you to refer more easily to the installation directory as `/usr/local/mysql`.

If your `tar` does not have `z` option support, use `gunzip` to unpack the distribution and `tar` to unpack it. Replace the preceding `tar` command with the following alternative command to uncompress and extract the distribution:

```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

Perform Postinstallation Setup

The remainder of the installation process involves setting up the configuration file, creating the core databases, and starting the MySQL server. For instructions, see [Postinstallation Setup and Testing](#).

Note

The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in [Postinstallation Setup and Testing](#).

Chapter 2. Installing MySQL on Linux

Table of Contents

2.1. Installing MySQL from RPM Packages on Linux	5
2.2. Installing MySQL on Linux using Native Package Manager	9

Linux supports a number of different solutions for installing MySQL. The recommended method is to use one of the distributions from Oracle. If you choose this method, there are three options available:

- Installing from a generic binary package in `.tar.gz` format. See [Chapter 1, *Installing MySQL from Generic Binaries on Unix/Linux*](#) for more information.
- Extracting and compiling MySQL from a source distribution. For detailed instructions, see [Installing MySQL from Source](#).
- Installing using a pre-compiled RPM package. For more information on using the RPM solution, see [Section 2.1, “Installing MySQL from RPM Packages on Linux”](#).
- Installing using Oracle's Unbreakable Linux Network (ULN). For more information, see [Installing MySQL Using Unbreakable Linux Network \(ULN\)](#).

As an alternative, you can use the native package manager within your Linux distribution to automatically download and install MySQL for you. Native package installations can take of the download and dependencies required to run MySQL, but the MySQL version will often be some way behind the currently available release. You will also normally be unable to install developmental releases, as these are not usually made available in the native repository. For more information on using the native package installers, see [Section 2.2, “Installing MySQL on Linux using Native Package Manager”](#).

Note

For many Linux installations, you will want to set up MySQL to be started automatically when your machine starts. Many of the native package installations perform this operation for you, but for source, binary and RPM solutions you may need to set this up separately. The required script, `mysql.server`, can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. You can install it as `/etc/init.d/mysql` for automatic MySQL startup and shutdown. See [Section 5.2, “Starting and Stopping MySQL Automatically”](#).

2.1. Installing MySQL from RPM Packages on Linux

The recommended way to install MySQL on RPM-based Linux distributions is by using the RPM packages. The RPMs that we provide to the community should work on all versions of Linux that support RPM packages and use `glibc` 2.3. To obtain RPM packages, see [How to Get MySQL](#).

For non-RPM Linux distributions, you can install MySQL using a `.tar.gz` package. See [Chapter 1, *Installing MySQL from Generic Binaries on Unix/Linux*](#).

Installations created from our Linux RPM distributions result in files under the following system directories.

Table 2.1. MySQL Installation Layout for Linux RPM

Directory	Contents of Directory
<code>/usr/bin</code>	Client programs and scripts

Directory	Contents of Directory
<code>/usr/sbin</code>	The <code>mysqld</code> server
<code>/var/lib/mysql</code>	Log files, databases
<code>/usr/share/info</code>	Manual in Info format
<code>/usr/share/man</code>	Unix manual pages
<code>/usr/include/mysql</code>	Include (header) files
<code>/usr/lib/mysql</code>	Libraries
<code>/usr/share/mysql</code>	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation
<code>/usr/share/sql-bench</code>	Benchmarks

Note

RPM distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by us in features, capabilities, and conventions (including communication setup), and that the instructions in this manual do not necessarily apply to installing them. The vendor's instructions should be consulted instead. Because of these differences, RPM packages built by us check whether such RPMs built by other vendors are installed. If so, the RPM does not install and produces a message explaining this.

In most cases, you need to install only the `MySQL-server` and `MySQL-client` packages to get a functional MySQL installation. The other packages are not required for a standard installation.

RPMs for MySQL Cluster. Standard MySQL server RPMs built by MySQL do not provide support for the `NDBCLUSTER` storage engine.

Important

When upgrading a MySQL Cluster RPM installation, you must upgrade *all* installed RPMs, including the `Server` and `Client` RPMs.

For more information about installing MySQL Cluster from RPMs, see [MySQL Cluster Installation](#).

For upgrades, if your installation was originally produced by installing multiple RPM packages, it is best to upgrade all the packages, not just some. For example, if you previously installed the server and client RPMs, do not upgrade just the server RPM.

If you get a dependency failure when trying to install MySQL packages (for example, `error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...`), you should also install the `MySQL-shared-compat` package, which includes the shared libraries for older releases for backward compatibility.

The RPM packages shown in the following list are available. The names shown here use a suffix of `.glibc23.i386.rpm`, but particular packages can have different suffixes, described later.

- `MySQL-server-VERSION.glibc23.i386.rpm`

The MySQL server. You need this unless you only want to connect to a MySQL server running on another machine.

- `MySQL-client-VERSION.glibc23.i386.rpm`

The standard MySQL client programs. You probably always want to install this package.

- `MySQL-devel-VERSION.glibc23.i386.rpm`

The libraries and include files that are needed if you want to compile other MySQL clients, such as the Perl modules.

- `MySQL-shared-VERSION.glibc23.i386.rpm`

This package contains the shared libraries (`libmysqlclient.so*`) that certain languages and applications need to dynamically load and use MySQL. It contains single-threaded and thread-safe libraries. Prior to MySQL 5.5.6, if you install this package, do not install the `MySQL-shared-compat` package.

- `MySQL-shared-compat-VERSION.glibc23.i386.rpm`

This package includes the shared libraries for older releases. It contains single-threaded and thread-safe libraries. Install this package if you have applications installed that are dynamically linked against older versions of MySQL but you want to upgrade to the current version without breaking the library dependencies. Before MySQL 5.5.6, `MySQL-shared-compat` also includes the libraries for the current release, so if you install it, you should not also install `MySQL-shared`. As of 5.5.6, `MySQL-shared-compat` does not include the current library version, so there is no conflict.

As of MySQL 5.5.23, the `MySQL-shared-compat` RPM package enables users of Red Hat-provided `mysql-*5.1` RPM packages to migrate to Oracle-provided `MySQL-*5.5` packages. `MySQL-shared-compat` replaces the Red Hat `mysql-libs` package by replacing `libmysqlclient.so` files of the latter package, thus satisfying dependencies of other packages on `mysql-libs`. This change affects only users of Red Hat (or Red Hat-compatible) RPM packages. Nothing is different for users of Oracle RPM packages.

- `MySQL-embedded-VERSION.glibc23.i386.rpm`

The embedded MySQL server library.

- `MySQL-test-VERSION.glibc23.i386.rpm`

This package includes the MySQL test suite.

- `MySQL-VERSION.src.rpm`

This contains the source code for all of the previous packages. It can also be used to rebuild the RPMs on other architectures (for example, Alpha or SPARC).

The suffix of RPM package names (following the `VERSION` value) has the following syntax:

```
.PLATFORM.CPU.rpm
```

The `PLATFORM` and `CPU` values indicate the type of system for which the package is built. `PLATFORM` indicates the platform and `CPU` indicates the processor type or family.

All packages are dynamically linked against `glibc` 2.3. The `PLATFORM` value indicates whether the package is platform independent or intended for a specific platform, as shown in the following table.

Table 2.2. MySQL Linux Installation Packages

<code>PLATFORM</code> Value	Intended Use
<code>glibc23</code>	Platform independent, should run on any Linux distribution that supports <code>glibc</code> 2.3

PLATFORM Value	Intended Use
<code>rhel4, rhel5</code>	Red Hat Enterprise Linux 4 or 5
<code>el6</code>	Enterprise Linux 6
<code>sles10, sles11</code>	SuSE Linux Enterprise Server 10 or 11

In MySQL 5.5, only `glibc23` packages are available currently.

The `CPU` value indicates the processor type or family for which the package is built.

Table 2.3. MySQL Installation Packages for Linux CPU Identifier

CPU Value	Intended Processor Type or Family
<code>i386, i586, i686</code>	Pentium processor or better, 32 bit
<code>x86_64</code>	64-bit x86 processor
<code>ia64</code>	Itanium (IA-64) processor

To see all files in an RPM package (for example, a `MySQL-server` RPM), run a command like this:

```
shell> rpm -qpl MySQL-server-VERSION.glibc23.i386.rpm
```

To perform a standard minimal installation, install the server and client RPMs:

```
shell> rpm -i MySQL-server-VERSION.glibc23.i386.rpm
shell> rpm -i MySQL-client-VERSION.glibc23.i386.rpm
```

To install only the client programs, install just the client RPM:

```
shell> rpm -i MySQL-client-VERSION.glibc23.i386.rpm
```

RPM provides a feature to verify the integrity and authenticity of packages before installing them. If you would like to learn more about this feature, see [Verifying Package Integrity Using MD5 Checksums or GnuPG](#).

The server RPM places data under the `/var/lib/mysql` directory. The RPM also creates a login account for a user named `mysql` (if one does not exist) to use for running the MySQL server, and creates the appropriate entries in `/etc/init.d/` to start the server automatically at boot time. (This means that if you have performed a previous installation and have made changes to its startup script, you may want to make a copy of the script so that you do not lose it when you install a newer RPM.) See [Section 5.2, “Starting and Stopping MySQL Automatically”](#), for more information on how MySQL can be started automatically on system startup.

In MySQL 5.5.5 and later, during a new installation, the server boot scripts are installed, but the MySQL server is not started at the end of the installation, since the status of the server during an unattended installation is not known.

In MySQL 5.5.5 and later, during an upgrade installation using the RPM packages, if the MySQL server is running when the upgrade occurs, the MySQL server is stopped, the upgrade occurs, and the MySQL server is restarted. If the MySQL server is not already running when the RPM upgrade occurs, the MySQL server is not started at the end of the installation.

If something goes wrong, you can find more information in the binary installation section. See [Chapter 1, Installing MySQL from Generic Binaries on Unix/Linux](#).

Note

The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in [Postinstallation Setup and Testing](#).

During RPM installation, a user named `mysql` and a group named `mysql` are created on the system. This is done using the `useradd`, `groupadd`, and `usermod` commands. Those commands require appropriate administrative privileges, which is required for locally managed users and groups (as listed in the `/etc/passwd` and `/etc/group` files) by the RPM installation process being run by `root`.

If you log in as the `mysql` user, you may find that MySQL displays “Invalid (old?) table or database name” errors that mention `.mysqlgui`, `lost+found`, `.mysqlgui`, `.bash_history`, `.fonts.cache-1`, `.lesshtst`, `.mysql_history`, `.profile`, `.viminfo`, and similar files created by MySQL or operating system utilities. You can safely ignore these error messages or remove the files or directories that cause them if you do not need them.

For nonlocal user management (LDAP, NIS, and so forth), the administrative tools may require additional authentication (such as a password), and will fail if the installing user does not provide this authentication. Even if they fail, the RPM installation will not abort but succeed, and this is intentional. If they failed, some of the intended transfer of ownership may be missing, and it is recommended that the system administrator then manually ensures some appropriate user and group exists and manually transfers ownership following the actions in the RPM spec file.

2.2. Installing MySQL on Linux using Native Package Manager

Many Linux distributions include a version of the MySQL server, client tools, and development components into the standard package management system built into distributions such as Fedora, Debian, Ubuntu, and Gentoo. This section provides basic instructions for installing MySQL using these systems.

Important

Native package installations can take care of the download and dependencies required to run MySQL, but the MySQL version will often be some way behind the currently available release. You will also normally be unable to install developmental releases, as these are not usually made available in the native repository.

Distribution specific instructions are shown below:

- **Red Hat Linux, Fedora, CentOS**

For Red Hat and similar distributions, the MySQL distribution is divided into a number of separate packages, `mysql` for the client tools, `mysql-server` for the server and associated tools, and `mysql-libs` for the libraries. The libraries are required if you want to provide connectivity from different languages and environments such as Perl, Python and others.

To install, use the `yum` command to specify the packages that you want to install. For example:

```
root-shell> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
--> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
--> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
--> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
```

```
--> Running transaction check
---> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
--> Finished Dependency Resolution
Dependencies Resolved

=====
Package                Arch          Version           Repository        Size
=====
Installing:
mysql                  x86_64        5.1.48-2.fc13     updates           889 k
mysql-libs             x86_64        5.1.48-2.fc13     updates           1.2 M
mysql-server          x86_64        5.1.48-2.fc13     updates           8.1 M
Installing for dependencies:
perl-DBD-MySQL        x86_64        4.017-1.fc13     updates           136 k
Transaction Summary
=====
Install      4 Package(s)
Upgrade      0 Package(s)
Total download size: 10 M
Installed size: 30 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
Processing delta metadata
Package(s) data still to download: 10 M
(1/4): mysql-5.1.48-2.fc13.x86_64.rpm | 889 kB    00:04
(2/4): mysql-libs-5.1.48-2.fc13.x86_64.rpm | 1.2 MB    00:06
(3/4): mysql-server-5.1.48-2.fc13.x86_64.rpm | 8.1 MB    00:40
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm | 136 kB    00:00
-----
Total                                     201 kB/s | 10 MB    00:52
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing      : mysql-libs-5.1.48-2.fc13.x86_64                1/4
  Installing      : mysql-5.1.48-2.fc13.x86_64                    2/4
  Installing      : perl-DBD-MySQL-4.017-1.fc13.x86_64            3/4
  Installing      : mysql-server-5.1.48-2.fc13.x86_64             4/4
Installed:
mysql.x86_64 0:5.1.48-2.fc13          mysql-libs.x86_64 0:5.1.48-2.fc13
mysql-server.x86_64 0:5.1.48-2.fc13
Dependency Installed:
perl-DBD-MySQL.x86_64 0:4.017-1.fc13
Complete!
```

MySQL and the MySQL server should now be installed. A sample configuration file is installed into `/etc/my.cnf`. An init script, to start and stop the server, will have been installed into `/etc/init.d/mysql`. To start the MySQL server use `service`:

```
root-shell> service mysqld start
```

To enable the server to be started and stopped automatically during boot, use `chkconfig`:

```
root-shell> chkconfig --levels 235 mysqld on
```

Which enables the MySQL server to be started (and stopped) automatically at the specified the run levels.

The database tables will have been automatically created for you, if they do not already exist. You should, however, run `mysql_secure_installation` to set the root passwords on your server.

- **Debian, Ubuntu, Kubuntu**

On Debian and related distributions, there are two packages, `mysql-client` and `mysql-server`, for the client and server components respectively. You should specify an explicit version, for example `mysql-client-5.1`, to ensure that you install the version of MySQL that you want.

To download and install, including any dependencies, use the `apt-get` command, specifying the packages that you want to install.

Note

Before installing, make sure that you update your `apt-get` index files to ensure you are downloading the latest available version.

A sample installation of the MySQL packages might look like this (some sections trimmed for clarity):

```
root-shell> apt-get install mysql-client-5.1 mysql-server-5.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.28-11 linux-headers-2.6.28-11-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-common postfix
Suggested packages:
  dbshell libipc-sharedcache-perl tinycat procmail postfix-mysql postfix-pgsql
  postfix-ldap postfix-pcre sasl2-bin resolvconf postfix-cdb
The following NEW packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-client-5.1 mysql-common mysql-server-5.1 postfix
0 upgraded, 13 newly installed, 0 to remove and 182 not upgraded.
Need to get 1907kB/25.3MB of archives.
After this operation, 59.5MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get: 1 http://gb.archive.ubuntu.com jaunty-updates/main mysql-common 5.1.30really5.0.75-0ubuntu10.5 [63.
Get: 2 http://gb.archive.ubuntu.com jaunty-updates/main libmysqlclient15off 5.1.30really5.0.75-0ubuntu10
Fetched 1907kB in 9s (205kB/s)
Preconfiguring packages ...
Selecting previously deselected package mysql-common.
(Reading database ... 121260 files and directories currently installed.)
...
Processing 1 added doc-base file(s)...
Registering documents with scrollkeeper...
Setting up libnet-daemon-perl (0.43-1) ...
Setting up libplrpc-perl (0.2020-1) ...
Setting up libdbi-perl (1.607-1) ...
Setting up libmysqlclient15off (5.1.30really5.0.75-0ubuntu10.5) ...
Setting up libdbd-mysql-perl (4.008-1) ...
Setting up libmysqlclient16 (5.1.31-1ubuntu2) ...
Setting up mysql-client-5.1 (5.1.31-1ubuntu2) ...
Setting up mysql-server-5.1 (5.1.31-1ubuntu2) ...
  * Stopping MySQL database server mysqld
    ...done.
100825 11:46:15 InnoDB: Started; log sequence number 0 46409
100825 11:46:15 InnoDB: Starting shutdown...
100825 11:46:17 InnoDB: Shutdown completed; log sequence number 0 46409
100825 11:46:17 [Warning] Forcing shutdown of 1 plugins
  * Starting MySQL database server mysqld
    ...done.
  * Checking for corrupt, not cleanly closed and upgrade needing tables.
...
Processing triggers for libc6 ...
```

```
ldconfig deferred processing now taking place
```

Note

The `apt-get` command will install a number of packages, including the MySQL server, in order to provide the typical tools and application environment. This can mean that you install a large number of packages in addition to the main MySQL package.

During installation, the initial database will be created, and you will be prompted for the MySQL root password (and confirmation). A configuration file will have been created in `/etc/mysql/my.cnf`. An init script will have been created in `/etc/init.d/mysql`.

The server will already be started. You can manually start and stop the server using:

```
root-shell> service mysql [start|stop]
```

The service will automatically be added to the 2, 3 and 4 run levels, with stop scripts in the single, shutdown and restart levels.

• Gentoo Linux

As a source-based distribution, installing MySQL on Gentoo involves downloading the source, patching the Gentoo specifics, and then compiling the MySQL server and installing it. This process is handled automatically by the `emerge` command. Depending on the version of MySQL that you want to install, you may need to unmask the specific version that you want for your chosen platform.

The MySQL server and client tools are provided within a single package, `dev-db/mysql`. You can obtain a list of the versions available to install by looking at the portage directory for the package:

```
root-shell> ls /usr/portage/dev-db/mysql/mysql-5.1*
mysql-5.1.39-r1.ebuild
mysql-5.1.44-r1.ebuild
mysql-5.1.44-r2.ebuild
mysql-5.1.44-r3.ebuild
mysql-5.1.44.ebuild
mysql-5.1.45-r1.ebuild
mysql-5.1.45.ebuild
mysql-5.1.46.ebuild
```

To install a specific MySQL version, you must specify the entire atom. For example:

```
root-shell> emerge =dev-db/mysql-5.1.46
```

A simpler alternative is to use the `virtual/mysql-5.1` package, which will install the latest version:

```
root-shell> emerge =virtual/mysql-5.1
```

If the package is masked (because it is not tested or certified for the current platform), use the `ACCEPT_KEYWORDS` environment variable. For example:

```
root-shell> ACCEPT_KEYWORDS="~x86" emerge =virtual/mysql-5.1
```

After installation, you should create a new database using `mysql_install_db`, and set the password for the root user on MySQL. You can use the configuration interface to set the password and create the initial database:

```
root-shell> emerge --config =dev-db/mysql-5.1.46
```

A sample configuration file will have been created for you in `/etc/mysql/my.cnf`, and an init script will have been created in `/etc/init.d/mysql`.

To enable MySQL to start automatically at the normal (default) run levels, you can use:

```
root-shell> rc-update add mysql default
```

Chapter 3. Installing MySQL on Solaris and OpenSolaris

Table of Contents

3.1. Installing MySQL on Solaris using a Solaris <code>PKG</code>	16
3.2. Installing MySQL on OpenSolaris using IPS	16

MySQL on Solaris and OpenSolaris is available in a number of different formats.

- For information on installing using the native Solaris `PKG` format, see [Section 3.1, “Installing MySQL on Solaris using a Solaris `PKG`”](#).
- On OpenSolaris, the standard package repositories include MySQL packages specially built for OpenSolaris that include entries for the Service Management Framework (SMF) to enable control of the installation using the SMF administration commands. For more information, see [Section 3.2, “Installing MySQL on OpenSolaris using IPS”](#).
- To use a standard `tar` binary installation, use the notes provided in [Chapter 1, *Installing MySQL from Generic Binaries on Unix/Linux*](#). Check the notes and hints at the end of this section for Solaris specific notes that you may need before or after installation.

To obtain a binary MySQL distribution for Solaris in tarball or `PKG` format, <http://dev.mysql.com/downloads/mysql/5.5.html>.

Additional notes to be aware of when installing and using MySQL on Solaris:

- If you want to use MySQL with the `mysql` user and group, use the `groupadd` and `useradd` commands:

```
groupadd mysql
useradd -g mysql mysql
```

- If you install MySQL using a binary tarball distribution on Solaris, you may run into trouble even before you get the MySQL distribution unpacked, as the Solaris `tar` cannot handle long file names. This means that you may see errors when you try to unpack MySQL.

If this occurs, you must use GNU `tar` (`gtar`) to unpack the distribution. In Solaris 10 and OpenSolaris `gtar` is normally located in `/usr/sfw/bin/gtar`, but may not be included in the default path definition.

- When using Solaris 10 for `x86_64`, you should mount any file systems on which you intend to store `InnoDB` files with the `forcedirectio` option. (By default mounting is done without this option.) Failing to do so will cause a significant drop in performance when using the `InnoDB` storage engine on this platform.
- If you would like MySQL to start automatically, you can copy `support-files/mysql.server` to `/etc/init.d` and create a symbolic link to it named `/etc/rc3.d/S99mysql.server`.
- If too many processes try to connect very rapidly to `mysqld`, you should see this error in the MySQL log:

```
Error in accept: Protocol error
```

You might try starting the server with the `--back_log=50` option as a workaround for this.

- To configure the generation of core files on Solaris you should use the `coreadm` command. Because of the security implications of generating a core on a `setuid()` application, by default, Solaris does not support core files on `setuid()` programs. However, you can modify this behavior using `coreadm`.

If you enable `setuid()` core files for the current user, they will be generated using the mode 600 and owned by the superuser.

3.1. Installing MySQL on Solaris using a Solaris [PKG](#)

You can install MySQL on Solaris and OpenSolaris using a binary package using the native Solaris PKG format instead of the binary tarball distribution.

To use this package, download the corresponding `mysql-VERSION-solaris10-PLATFORM.pkg.gz` file, then decompress it. For example:

```
shell> gunzip mysql-5.5.30-solaris10-x86_64.pkg.gz
```

To install a new package, use `pkgadd` and follow the onscreen prompts. You must have root privileges to perform this operation:

```
shell> pkgadd -d mysql-5.5.30-solaris10-x86_64.pkg
The following packages are available:
  1  mysql      MySQL Community Server (GPL)
      (i86pc) 5.5.30
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

The [PKG](#) installer installs all of the files and tools needed, and then initializes your database if one does not exist. To complete the installation, you should set the root password for MySQL as provided in the instructions at the end of the installation. Alternatively, you can run the `mysql_secure_installation` script that comes with the installation.

The default installation directory is `/opt/mysql`. You can only change the root path of the installation when using `pkgadd`, which can be used to install MySQL in a different Solaris zone. If you need to install in a specific directory, use the binary `tar` file.

The [pkg](#) installer copies a suitable startup script for MySQL into `/etc/init.d/mysql`. To enable MySQL to startup and shutdown automatically, you should create a link between this file and the init script directories. For example, to ensure safe startup and shutdown of MySQL you could use the following commands to add the right links:

```
shell> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
shell> ln /etc/init.d/mysql /etc/rc0.d/K02mysql
```

To remove MySQL, the installed package name is `mysql`. You can use this in combination with the `pkgrm` command to remove the installation.

To upgrade when using the Solaris package file format, you must remove the existing installation before installing the updated package. Removal of the package does not delete the existing database information, only the server, binaries and support files. The typical upgrade sequence is therefore:

```
shell> mysqladmin shutdown
shell> pkgrm mysql
shell> pkgadd -d mysql-5.5.30-solaris10-x86_64.pkg
shell> mysql_upgrade
shell> mysqld_safe &
```

You should check the notes in [Upgrading or Downgrading MySQL](#) before performing any upgrade.

3.2. Installing MySQL on OpenSolaris using IPS

OpenSolaris includes standard packages for MySQL in the core repository. The MySQL packages are based on a specific release of MySQL and updated periodically. For the latest release you must use either

the native Solaris [PKG](#), [tar](#), or source installations. The native OpenSolaris packages include SMF files so that you can easily control your MySQL installation, including automatic startup and recovery, using the native service management tools.

To install MySQL on OpenSolaris, use the [pkg](#) command. You will need to be logged in as root, or use the [pfexec](#) tool, as shown in the example below:

```
shell> pfexec pkg install SUNWmysql55
```

The package set installs three individual packages, [SUNWmysql55lib](#), which contains the MySQL client libraries; [SUNWmysql55r](#) which contains the root components, including SMF and configuration files; and [SUNWmysql55u](#) which contains the scripts, binary tools and other files. You can install these packages individually if you only need the corresponding components.

The MySQL files are installed into [/usr/mysql](#) which symbolic links for the sub directories ([bin](#), [lib](#), etc.) to a version specific directory. For MySQL 5.5, the full installation is located in [/usr/mysql/5.5](#). The default data directory is [/var/mysql/5.5/data](#). The configuration file is installed in [/etc/mysql/5.5/my.cnf](#). This layout permits multiple versions of MySQL to be installed, without overwriting the data and binaries from other versions.

Once installed, you must run [mysql_install_db](#) to initialize the database, and use the [mysql_secure_installation](#) to secure your installation.

Using SMF to manage your MySQL installation

Once installed, you can start and stop your MySQL server using the installed SMF configuration. The service name is [mysql](#), or if you have multiple versions installed, you should use the full version name, for example [mysql:version_55](#). To start and enable MySQL to be started at boot time:

```
shell> svcadm enable mysql
```

To disable MySQL from starting during boot time, and shut the MySQL server down if it is running, use:

```
shell> svcadm disable mysql
```

To restart MySQL, for example after a configuration file changes, use the [restart](#) option:

```
shell> svcadm restart mysql
```

You can also use SMF to configure the data directory and enable full 64-bit mode. For example, to set the data directory used by MySQL:

```
shell> svccfg
svc:> select mysql:version_55
svc:/application/database/mysql:version_55> setprop mysql/data=/data0/mysql
```

By default, the 32-bit binaries are used. To enable the 64-bit server on 64-bit platforms, set the [enable_64bit](#) parameter. For example:

```
svc:/application/database/mysql:version_55> setprop mysql/enable_64bit=1
```

You need to refresh the SMF after settings these options:

```
shell> svcadm refresh mysql
```

Chapter 4. Installing MySQL on FreeBSD

This section provides information about installing MySQL on variants of FreeBSD Unix.

You can install MySQL on FreeBSD by using the binary distribution provided by Oracle. For more information, see [Chapter 1, *Installing MySQL from Generic Binaries on Unix/Linux*](#).

The easiest (and preferred) way to install MySQL is to use the `mysql-server` and `mysql-client` ports available at <http://www.freebsd.org/>. Using these ports gives you the following benefits:

- A working MySQL with all optimizations enabled that are known to work on your version of FreeBSD.
- Automatic configuration and build.
- Startup scripts installed in `/usr/local/etc/rc.d`.
- The ability to use `pkg_info -L` to see which files are installed.
- The ability to use `pkg_delete` to remove MySQL if you no longer want it on your machine.

The MySQL build process requires GNU make (`gmake`) to work. If GNU `make` is not available, you must install it first before compiling MySQL.

To install using the ports system:

```
# cd /usr/ports/databases/mysql51-server
# make
...
# cd /usr/ports/databases/mysql51-client
# make
...
```

The standard port installation places the server into `/usr/local/libexec/mysqld`, with the startup script for the MySQL server placed in `/usr/local/etc/rc.d/mysql-server`.

Some additional notes on the BSD implementation:

- To remove MySQL after installation using the ports system:

```
# cd /usr/ports/databases/mysql51-server
# make deinstall
...
# cd /usr/ports/databases/mysql51-client
# make deinstall
...
```

- If you get problems with the current date in MySQL, setting the `TZ` variable should help. See [Environment Variables](#).

Chapter 5. Unix Postinstallation Procedures

Table of Contents

5.1. Problems Running <code>mysql_install_db</code>	25
5.2. Starting and Stopping MySQL Automatically	27
5.3. Starting and Troubleshooting the MySQL Server	30

After installing MySQL on Unix, you must initialize the grant tables, start the server, and make sure that the server works satisfactorily. You may also wish to arrange for the server to be started and stopped automatically when your system starts and stops. You should also assign passwords to the accounts in the grant tables.

On Unix, the grant tables are set up by the `mysql_install_db` program. For some installation methods, this program is run for you automatically if an existing database cannot be found.

- If you install MySQL on Linux using RPM distributions, the server RPM runs `mysql_install_db`.
- Using the native packaging system on many platforms, including Debian Linux, Ubuntu Linux, Gentoo Linux and others, the `mysql_install_db` command is run for you.
- If you install MySQL on Mac OS X using a PKG distribution, the installer runs `mysql_install_db`.

For other platforms and installation types, including generic binary and source installs, you will need to run `mysql_install_db` yourself.

The following procedure describes how to initialize the grant tables (if that has not previously been done) and start the server. It also suggests some commands that you can use to test whether the server is accessible and working properly. For information about starting and stopping the server automatically, see [Section 5.2, “Starting and Stopping MySQL Automatically”](#).

After you complete the procedure and have the server running, you should assign passwords to the accounts created by `mysql_install_db` and perhaps restrict access to test databases. For instructions, see [Securing the Initial MySQL Accounts](#).

In the examples shown here, the server runs under the user ID of the `mysql` login account. This assumes that such an account exists. Either create the account if it does not exist, or substitute the name of a different existing login account that you plan to use for running the server. For information about creating the account, see [Creating a `mysql` System User and Group](#), in [Chapter 1, *Installing MySQL from Generic Binaries on Unix/Linux*](#).

1. Change location into the top-level directory of your MySQL installation, represented here by `BASEDIR`:

```
shell> cd BASEDIR
```

`BASEDIR` is the installation directory for your MySQL instance. It is likely to be something like `/usr/local/mysql` or `/usr/local`. The following steps assume that you have changed location to this directory.

You will find several files and subdirectories in the `BASEDIR` directory. The most important for installation purposes are the `bin` and `scripts` subdirectories:

- The `bin` directory contains client programs and the server. You should add the full path name of this directory to your `PATH` environment variable so that your shell finds the MySQL programs properly. See [Environment Variables](#).

-
- The `scripts` directory contains the `mysql_install_db` script used to initialize the `mysql` database containing the grant tables that store the server access permissions.
2. If necessary, ensure that the distribution contents are accessible to `mysql`. If you installed the distribution as `mysql`, no further action is required. If you installed the distribution as `root`, its contents will be owned by `root`. Change its ownership to `mysql` by executing the following commands as `root` in the installation directory. The first command changes the owner attribute of the files to the `mysql` user. The second changes the group attribute to the `mysql` group.

```
shell> chown -R mysql .
shell> chgrp -R mysql .
```

3. If necessary, run the `mysql_install_db` program to set up the initial MySQL grant tables containing the privileges that determine how users are permitted to connect to the server. You will need to do this if you used a distribution type for which the installation procedure does not run the program for you.

```
shell> scripts/mysql_install_db --user=mysql
```

Typically, `mysql_install_db` needs to be run only the first time you install MySQL, so you can skip this step if you are upgrading an existing installation. However, `mysql_install_db` does not overwrite any existing privilege tables, so it should be safe to run in any circumstances.

It might be necessary to specify other options such as `--basedir` or `--datadir` if `mysql_install_db` does not identify the correct locations for the installation directory or data directory. For example:

```
shell> scripts/mysql_install_db --user=mysql \
      --basedir=/opt/mysql/mysql \
      --datadir=/opt/mysql/mysql/data
```

The `mysql_install_db` script creates the server's data directory with `mysql` as the owner. Under the data directory, it creates directories for the `mysql` database that holds the grant tables and the `test` database that you can use to test MySQL. The script also creates privilege table entries for `root` and anonymous-user accounts. The accounts have no passwords initially. [Securing the Initial MySQL Accounts](#), describes the initial privileges. Briefly, these privileges permit the MySQL `root` user to do anything, and permit anybody to create or use databases with a name of `test` or starting with `test_`. See [The MySQL Access Privilege System](#), for a complete listing and description of the grant tables.

It is important to make sure that the database directories and files are owned by the `mysql` login account so that the server has read and write access to them when you run it later. To ensure this if you run `mysql_install_db` as `root`, include the `--user` option as shown. Otherwise, you should execute the script while logged in as `mysql`, in which case you can omit the `--user` option from the command.

If you do not want to have the `test` database, you can remove it after starting the server, using the instructions in [Securing the Initial MySQL Accounts](#).

If you have trouble with `mysql_install_db` at this point, see [Section 5.1, "Problems Running mysql_install_db"](#).

4. Most of the MySQL installation can be owned by `root` if you like. The exception is that the data directory must be owned by `mysql`. To accomplish this, run the following commands as `root` in the installation directory:

```
shell> chown -R root .
shell> chown -R mysql data
```

5. If the plugin directory (the directory named by the `plugin_dir` system variable) is writable by the server, it may be possible for a user to write executable code to a file in the directory using `SELECT ... INTO DUMPFILE`. This can be prevented by making `plugin_dir` read only to the server or by setting `--secure-file-priv` to a directory where `SELECT` writes can be made safely.
6. If you installed MySQL using a source distribution, you may want to optionally copy one of the provided configuration files from the `support-files` directory into your `/etc` directory. There are different sample configuration files for different use cases, server types, and CPU and RAM configurations. If you want to use one of these standard files, you should copy it to `/etc/my.cnf`, or `/etc/mysql/my.cnf` and edit and check the configuration before starting your MySQL server for the first time.

If you do not copy one of the standard configuration files, the MySQL server will be started with the default settings.

If you want MySQL to start automatically when you boot your machine, you can copy `support-files/mysql.server` to the location where your system has its startup files. More information can be found in the `mysql.server` script itself, and in [Section 5.2, “Starting and Stopping MySQL Automatically”](#).

7. Start the MySQL server:

```
shell> bin/mysqld_safe --user=mysql &
```

It is important that the MySQL server be run using an unprivileged (non-`root`) login account. To ensure this if you run `mysqld_safe` as `root`, include the `--user` option as shown. Otherwise, you should execute the script while logged in as `mysql`, in which case you can omit the `--user` option from the command.

For further instructions for running MySQL as an unprivileged user, see [How to Run MySQL as a Normal User](#).

If the command fails immediately and prints `mysqld ended`, look for information in the error log (which by default is the `host_name.err` file in the data directory).

If you neglected to create the grant tables by running `mysql_install_db` before proceeding to this step, the following message appears in the error log file when you start the server:

```
mysqld: Can't find file: 'host.frm'
```

This error also occurs if you run `mysql_install_db` as `root` without the `--user` option. Remove the `data` directory and run `mysql_install_db` with the `--user` option as described previously.

If you have other problems starting the server, see [Section 5.3, “Starting and Troubleshooting the MySQL Server”](#). For more information about `mysqld_safe`, see [mysqld_safe — MySQL Server Startup Script](#).

8. Use `mysqladmin` to verify that the server is running. The following commands provide simple tests to check whether the server is up and responding to connections:

```
shell> bin/mysqladmin version
shell> bin/mysqladmin variables
```

The output from `mysqladmin version` varies slightly depending on your platform and version of MySQL, but should be similar to that shown here:

```
shell> bin/mysqladmin version
mysqladmin Ver 14.12 Distrib 5.5.30, for pc-linux-gnu on i686
...
```

```

Server version      5.5.30
Protocol version    10
Connection          Localhost via UNIX socket
UNIX socket         /var/lib/mysql/mysql.sock
Uptime:             14 days 5 hours 5 min 21 sec
Threads: 1  Questions: 366  Slow queries: 0
Opens: 0  Flush tables: 1  Open tables: 19
Queries per second avg: 0.000

```

To see what else you can do with `mysqladmin`, invoke it with the `--help` option.

9. Verify that you can shut down the server:

```
shell> bin/mysqladmin -u root shutdown
```

10. Verify that you can start the server again. Do this by using `mysqld_safe` or by invoking `mysqld` directly. For example:

```
shell> bin/mysqld_safe --user=mysql &
```

If `mysqld_safe` fails, see [Section 5.3, “Starting and Troubleshooting the MySQL Server”](#).

11. Run some simple tests to verify that you can retrieve information from the server. The output should be similar to what is shown here:

```

shell> bin/mysqlshow
+-----+
|      Databases      |
+-----+
| information_schema |
| mysql              |
| test               |
+-----+
shell> bin/mysqlshow mysql
Database: mysql
+-----+
|      Tables      |
+-----+
| columns_priv |
| db            |
| event        |
| func         |
| help_category |
| help_keyword  |
| help_relation |
| help_topic    |
| host         |
| plugin       |
| proc         |
| procs_priv   |
| servers      |
| tables_priv  |
| time_zone    |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user         |
+-----+
shell> bin/mysql -e "SELECT Host,Db,User FROM db" mysql
+-----+
| host | db   | user |
+-----+
| %    | test |      |
| %    | test_% |    |

```

12. There is a benchmark suite in the `sql-bench` directory (under the MySQL installation directory) that you can use to compare how MySQL performs on different platforms. The benchmark suite is written in Perl. It requires the Perl DBI module that provides a database-independent interface to the various databases, and some other additional Perl modules:

```
DBI
DBD::mysql
Data::Dumper
Data::ShowTable
```

These modules can be obtained from CPAN (<http://www.cpan.org/>). See also [Installing Perl on Unix](#).

The `sql-bench/Results` directory contains the results from many runs against different databases and platforms. To run all tests, execute these commands:

```
shell> cd sql-bench
shell> perl run-all-tests
```

If you do not have the `sql-bench` directory, you probably installed MySQL using RPM files other than the source RPM. (The source RPM includes the `sql-bench` benchmark directory.) In this case, you must first install the benchmark suite before you can use it. There are separate benchmark RPM files named `mysql-bench-VERSION.i386.rpm` that contain benchmark code and data.

If you have a source distribution, there are also tests in its `tests` subdirectory that you can run. For example, to run `auto_increment.tst`, execute this command from the top-level directory of your source distribution:

```
shell> mysql -vvf test < ./tests/auto_increment.tst
```

The expected result of the test can be found in the `./tests/auto_increment.res` file.

13. At this point, you should have the server running. However, none of the initial MySQL accounts have a password, and the server permits permissive access to test databases. To tighten security, follow the instructions in [Securing the Initial MySQL Accounts](#).

The MySQL 5.5 installation procedure creates time zone tables in the `mysql` database but does not populate them. To do so, use the instructions in [MySQL Server Time Zone Support](#).

To make it more convenient to invoke programs installed in the `bin` directory under the installation directory, you can add that directory to your `PATH` environment variable setting. That enables you to run a program by typing only its name, not its entire path name. See [Setting Environment Variables](#).

You can set up new accounts using the `bin/mysql_setpermission` script if you install the `DBI` and `DBD::mysql` Perl modules. See [mysql_setpermission — Interactively Set Permissions in Grant Tables](#). For Perl module installation instructions, see [Perl Installation Notes](#).

If you would like to use `mysqlaccess` and have the MySQL distribution in some nonstandard location, you must change the location where `mysqlaccess` expects to find the `mysql` client. Edit the `bin/mysqlaccess` script at approximately line 18. Search for a line that looks like this:

```
$MYSQL = '/usr/local/bin/mysql'; # path to mysql executable
```

Change the path to reflect the location where `mysql` actually is stored on your system. If you do not do this, a [Broken pipe](#) error will occur when you run `mysqlaccess`.

5.1. Problems Running `mysql_install_db`

The purpose of the `mysql_install_db` script is to generate new MySQL privilege tables. It does not overwrite existing MySQL privilege tables, and it does not affect any other data.

If you want to re-create your privilege tables, first stop the `mysqld` server if it is running. Then rename the `mysql` directory under the data directory to save it, and then run `mysql_install_db`. Suppose that your current directory is the MySQL installation directory and that `mysql_install_db` is located in the `bin` directory and the data directory is named `data`. To rename the `mysql` database and re-run `mysql_install_db`, use these commands.

```
shell> mv data/mysql data/mysql.old
shell> scripts/mysql_install_db --user=mysql
```

When you run `mysql_install_db`, you might encounter the following problems:

- **`mysql_install_db` fails to install the grant tables**

You may find that `mysql_install_db` fails to install the grant tables and terminates after displaying the following messages:

```
Starting mysqld daemon with databases from XXXXXX
mysqld ended
```

In this case, you should examine the error log file very carefully. The log should be located in the directory `XXXXXX` named by the error message and should indicate why `mysqld` did not start. If you do not understand what happened, include the log when you post a bug report. See [How to Report Bugs or Problems](#).

- **There is a `mysqld` process running**

This indicates that the server is running, in which case the grant tables have probably been created already. If so, there is no need to run `mysql_install_db` at all because it needs to be run only once (when you install MySQL the first time).

- **Installing a second `mysqld` server does not work when one server is running**

This can happen when you have an existing MySQL installation, but want to put a new installation in a different location. For example, you might have a production installation, but you want to create a second installation for testing purposes. Generally the problem that occurs when you try to run a second server is that it tries to use a network interface that is in use by the first server. In this case, you should see one of the following error messages:

```
Can't start server: Bind on TCP/IP port:
Address already in use
Can't start server: Bind on unix socket...
```

For instructions on setting up multiple servers, see [Running Multiple MySQL Instances on One Machine](#).

- **You do not have write access to the `/tmp` directory**

If you do not have write access to create temporary files or a Unix socket file in the default location (the `/tmp` directory) or the `TMP_DIR` environment variable, if it has been set, an error occurs when you run `mysql_install_db` or the `mysqld` server.

You can specify different locations for the temporary directory and Unix socket file by executing these commands prior to starting `mysql_install_db` or `mysqld`, where `some_tmp_dir` is the full path name to some directory for which you have write permission:

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
```



```
shell> export TMPDIR MYSQL_UNIX_PORT
```

Then you should be able to run `mysql_install_db` and start the server with these commands:

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
```

If `mysql_install_db` is located in the `scripts` directory, modify the first command to `scripts/mysql_install_db`.

See [How to Protect or Change the MySQL Unix Socket File](#), and [Environment Variables](#).

There are some alternatives to running the `mysql_install_db` script provided in the MySQL distribution:

- If you want the initial privileges to be different from the standard defaults, you can modify `mysql_install_db` before you run it. However, it is preferable to use `GRANT` and `REVOKE` to change the privileges *after* the grant tables have been set up. In other words, you can run `mysql_install_db`, and then use `mysql -u root mysql` to connect to the server as the MySQL `root` user so that you can issue the necessary `GRANT` and `REVOKE` statements.

If you want to install MySQL on several machines with the same privileges, you can put the `GRANT` and `REVOKE` statements in a file and execute the file as a script using `mysql` after running `mysql_install_db`. For example:

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysql -u root < your_script_file
```

By doing this, you can avoid having to issue the statements manually on each machine.

- It is possible to re-create the grant tables completely after they have previously been created. You might want to do this if you are just learning how to use `GRANT` and `REVOKE` and have made so many modifications after running `mysql_install_db` that you want to wipe out the tables and start over.

To re-create the grant tables, remove all the `.frm`, `.MYI`, and `.MYD` files in the `mysql` database directory. Then run the `mysql_install_db` script again.

- You can start `mysqld` manually using the `--skip-grant-tables` option and add the privilege information yourself using `mysql`:

```
shell> bin/mysqld_safe --user=mysql --skip-grant-tables &
shell> bin/mysql mysql
```

From `mysql`, manually execute the SQL commands contained in `mysql_install_db`. Make sure that you run `mysqladmin flush-privileges` or `mysqladmin reload` afterward to tell the server to reload the grant tables.

Note that by not using `mysql_install_db`, you not only have to populate the grant tables manually, you also have to create them first.

5.2. Starting and Stopping MySQL Automatically

Generally, you start the `mysqld` server in one of these ways:

- Invoke `mysqld` directly. This works on any platform.
- Run the MySQL server as a Windows service. The service can be set to start the server automatically when Windows starts, or as a manual service that you start on request. For instructions, see [Starting MySQL as a Windows Service](#).

- Invoke `mysqld_safe`, which tries to determine the proper options for `mysqld` and then runs it with those options. This script is used on Unix and Unix-like systems. See [mysqld_safe — MySQL Server Startup Script](#).
- Invoke `mysql.server`. This script is used primarily at system startup and shutdown on systems that use System V-style run directories (that is, `/etc/init.d` and run-level specific directories), where it usually is installed under the name `mysql`. The `mysql.server` script starts the server by invoking `mysqld_safe`. See [mysql.server — MySQL Server Startup Script](#).
- On Mac OS X, install a separate MySQL Startup Item package to enable the automatic startup of MySQL on system startup. The Startup Item starts the server by invoking `mysql.server`. See [Installing the MySQL Startup Item](#), for details. A MySQL Preference Pane also provides control for starting and stopping MySQL through the System Preferences, see [Installing and Using the MySQL Preference Pane](#).
- Use the Solaris/OpenSolaris service management framework (SMF) system to initiate and control MySQL startup. For more information, see [Section 3.2, “Installing MySQL on OpenSolaris using IPS”](#).

The `mysqld_safe` and `mysql.server` scripts, Windows server, Solaris/OpenSolaris SMF, and the Mac OS X Startup Item (or MySQL Preference Pane) can be used to start the server manually, or automatically at system startup time. `mysql.server` and the Startup Item also can be used to stop the server.

To start or stop the server manually using the `mysql.server` script, invoke it with `start` or `stop` arguments:

```
shell> mysql.server start
shell> mysql.server stop
```

Before `mysql.server` starts the server, it changes location to the MySQL installation directory, and then invokes `mysqld_safe`. If you want the server to run as some specific user, add an appropriate `user` option to the `[mysqld]` group of the `/etc/my.cnf` option file, as shown later in this section. (It is possible that you will need to edit `mysql.server` if you've installed a binary distribution of MySQL in a nonstandard location. Modify it to change location into the proper directory before it runs `mysqld_safe`. If you do this, your modified version of `mysql.server` may be overwritten if you upgrade MySQL in the future, so you should make a copy of your edited version that you can reinstall.)

`mysql.server stop` stops the server by sending a signal to it. You can also stop the server manually by executing `mysqldadmin shutdown`.

To start and stop MySQL automatically on your server, you need to add start and stop commands to the appropriate places in your `/etc/rc*` files.

If you use the Linux server RPM package (`MySQL-server-VERSION.rpm`), or a native Linux package installation, the `mysql.server` script may be installed in the `/etc/init.d` directory with the name `mysql`. See [Section 2.1, “Installing MySQL from RPM Packages on Linux”](#), for more information on the Linux RPM packages.

Some vendors provide RPM packages that install a startup script under a different name such as `mysqld`.

If you install MySQL from a source distribution or using a binary distribution format that does not install `mysql.server` automatically, you can install it manually. The script can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree.

To install `mysql.server` manually, copy it to the `/etc/init.d` directory with the name `mysql`, and then make it executable. Do this by changing location into the appropriate directory where `mysql.server` is located and executing these commands:

```
shell> cp mysql.server /etc/init.d/mysql
shell> chmod +x /etc/init.d/mysql
```

Note

Older Red Hat systems use the `/etc/rc.d/init.d` directory rather than `/etc/init.d`. Adjust the preceding commands accordingly. Alternatively, first create `/etc/init.d` as a symbolic link that points to `/etc/rc.d/init.d`:

```
shell> cd /etc
shell> ln -s rc.d/init.d .
```

After installing the script, the commands needed to activate it to run at system startup depend on your operating system. On Linux, you can use `chkconfig`:

```
shell> chkconfig --add mysql
```

On some Linux systems, the following command also seems to be necessary to fully enable the `mysql` script:

```
shell> chkconfig --level 345 mysql on
```

On FreeBSD, startup scripts generally should go in `/usr/local/etc/rc.d/`. The `rc(8)` manual page states that scripts in this directory are executed only if their basename matches the `*.sh` shell file name pattern. Any other files or directories present within the directory are silently ignored. In other words, on FreeBSD, you should install the `mysql.server` script as `/usr/local/etc/rc.d/mysql.server.sh` to enable automatic startup.

As an alternative to the preceding setup, some operating systems also use `/etc/rc.local` or `/etc/init.d/boot.local` to start additional services on startup. To start up MySQL using this method, you could append a command like the one following to the appropriate startup file:

```
/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

For other systems, consult your operating system documentation to see how to install startup scripts.

You can add options for `mysql.server` in a global `/etc/my.cnf` file. A typical `/etc/my.cnf` file might look like this:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql
[mysql.server]
basedir=/usr/local/mysql
```

The `mysql.server` script supports the following options: `basedir`, `datadir`, and `pid-file`. If specified, they *must* be placed in an option file, not on the command line. `mysql.server` supports only `start` and `stop` as command-line arguments.

The following table shows which option groups the server and each startup script read from option files.

Table 5.1. MySQL Startup scripts and supported server option groups

Script	Option Groups
<code>mysqld</code>	<code>[mysqld]</code> , <code>[server]</code> , <code>[mysqld-major_version]</code>
<code>mysqld_safe</code>	<code>[mysqld]</code> , <code>[server]</code> , <code>[mysqld_safe]</code>
<code>mysql.server</code>	<code>[mysqld]</code> , <code>[mysql.server]</code> , <code>[server]</code>

`[mysqld-major_version]` means that groups with names like `[mysqld-5.1]` and `[mysqld-5.5]` are read by servers having versions 5.1.x, 5.5.x, and so forth. This feature can be used to specify options that can be read only by servers within a given release series.

For backward compatibility, `mysql.server` also reads the `[mysql_server]` group and `mysqld_safe` also reads the `[safe_mysqld]` group. However, you should update your option files to use the `[mysql.server]` and `[mysqld_safe]` groups instead when using MySQL 5.5.

For more information on MySQL configuration files and their structure and contents, see [Using Option Files](#).

5.3. Starting and Troubleshooting the MySQL Server

This section provides troubleshooting suggestions for problems starting the server on Unix. If you are using Windows, see [Troubleshooting a Microsoft Windows MySQL Server Installation](#).

If you have problems starting the server, here are some things to try:

- Check the error log to see why the server does not start.
- Specify any special options needed by the storage engines you are using.
- Make sure that the server knows where to find the data directory.
- Make sure that the server can access the data directory. The ownership and permissions of the data directory and its contents must be set such that the server can read and modify them.
- Verify that the network interfaces the server wants to use are available.

Some storage engines have options that control their behavior. You can create a `my.cnf` file and specify startup options for the engines that you plan to use. If you are going to use storage engines that support transactional tables ([InnoDB](#), [NDB](#)), be sure that you have them configured the way you want before starting the server:

If you are using [InnoDB](#) tables, see [Configuring InnoDB](#).

Storage engines will use default option values if you specify none, but it is recommended that you review the available options and specify explicit values for those for which the defaults are not appropriate for your installation.

When the `mysqld` server starts, it changes location to the data directory. This is where it expects to find databases and where it expects to write log files. The server also writes the pid (process ID) file in the data directory.

The data directory location is hardwired in when the server is compiled. This is where the server looks for the data directory by default. If the data directory is located somewhere else on your system, the server will not work properly. You can determine what the default path settings are by invoking `mysqld` with the `--verbose` and `--help` options.

If the default locations do not match the MySQL installation layout on your system, you can override them by specifying options to `mysqld` or `mysqld_safe` on the command line or in an option file.

To specify the location of the data directory explicitly, use the `--datadir` option. However, normally you can tell `mysqld` the location of the base directory under which MySQL is installed and it looks for the data directory there. You can do this with the `--basedir` option.

To check the effect of specifying path options, invoke `mysqld` with those options followed by the `--verbose` and `--help` options. For example, if you change location into the directory where `mysqld`

is installed and then run the following command, it shows the effect of starting the server with a base directory of `/usr/local`:

```
shell> ./mysqld --basedir=/usr/local --verbose --help
```

You can specify other options such as `--datadir` as well, but `--verbose` and `--help` must be the last options.

Once you determine the path settings you want, start the server without `--verbose` and `--help`.

If `mysqld` is currently running, you can find out what path settings it is using by executing this command:

```
shell> mysqladmin variables
```

Or:

```
shell> mysqladmin -h host_name variables
```

`host_name` is the name of the MySQL server host.

If you get `Errcode 13` (which means `Permission denied`) when starting `mysqld`, this means that the privileges of the data directory or its contents do not permit server access. In this case, you change the permissions for the involved files and directories so that the server has the right to use them. You can also start the server as `root`, but this raises security issues and should be avoided.

On Unix, change location into the data directory and check the ownership of the data directory and its contents to make sure the server has access. For example, if the data directory is `/usr/local/mysql/var`, use this command:

```
shell> ls -la /usr/local/mysql/var
```

If the data directory or its files or subdirectories are not owned by the login account that you use for running the server, change their ownership to that account. If the account is named `mysql`, use these commands:

```
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql/var
```

If it possible that even with correct ownership, MySQL may fail to start up if there is other security software running on your system that manages application access to various parts of the file system. In this case, you may need to reconfigure that software to enable `mysqld` to access the directories it uses during normal operation.

If the server fails to start up correctly, check the error log. Log files are located in the data directory (typically `C:\Program Files\MySQL\MySQL Server 5.5\data` on Windows, `/usr/local/mysql/data` for a Unix binary distribution, and `/usr/local/var` for a Unix source distribution). Look in the data directory for files with names of the form `host_name.err` and `host_name.log`, where `host_name` is the name of your server host. Then examine the last few lines of these files. On Unix, you can use `tail` to display them:

```
shell> tail host_name.err
shell> tail host_name.log
```

The error log should contain information that indicates why the server could not start.

If either of the following errors occur, it means that some other program (perhaps another `mysqld` server) is using the TCP/IP port or Unix socket file that `mysqld` is trying to use:

```
Can't start server: Bind on TCP/IP port: Address already in use
Can't start server: Bind on unix socket...
```

Use `ps` to determine whether you have another `mysqld` server running. If so, shut down the server before starting `mysqld` again. (If another server is running, and you really want to run multiple servers, you can find information about how to do so in [Running Multiple MySQL Instances on One Machine](#).)

If no other server is running, try to execute the command `telnet your_host_name tcp_ip_port_number`. (The default MySQL port number is 3306.) Then press Enter a couple of times. If you do not get an error message like `telnet: Unable to connect to remote host: Connection refused`, some other program is using the TCP/IP port that `mysqld` is trying to use. You will need to track down what program this is and disable it, or else tell `mysqld` to listen to a different port with the `--port` option. In this case, you will also need to specify the port number for client programs when connecting to the server using TCP/IP.

Another reason the port might be inaccessible is that you have a firewall running that blocks connections to it. If so, modify the firewall settings to permit access to the port.

If the server starts but you cannot connect to it, you should make sure that you have an entry in `/etc/hosts` that looks like this:

```
127.0.0.1      localhost
```

If you cannot get `mysqld` to start, you can try to make a trace file to find the problem by using the `--debug` option. See [MySQL Internals: Porting to Other Systems](#).