Starting and Stopping MySQL
Abstract

This is the Starting and Stopping MySQL extract from the MySQL 8.0 Reference Manual.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2020-04-02 (revision: 65595)
# Table of Contents

Preface and Legal Notices ........................................................................................................... v  
1 Installing MySQL on Unix/Linux Using Generic Binaries ...................................................... 1  
2 Starting the Server for the First Time on Windows ................................................................. 5  
3 The Server Shutdown Process ................................................................................................. 7  
4 Server and Server-Startup Programs ....................................................................................... 9  
  4.1 mysql — The MySQL Server ............................................................................................... 9  
  4.2 mysql_safe — MySQL Server Startup Script ................................................................. 9  
  4.3 mysql.server — MySQL Server Startup Script ............................................................ 15  
  4.4 mysql_multi — Manage Multiple MySQL Servers ......................................................... 17
Preface and Legal Notices

This is the Starting and Stopping MySQL extract from the MySQL 8.0 Reference Manual.

Licensing information—MySQL 8.0. This product may include third-party software, used under license. If you are using a Commercial release of MySQL 8.0, see the MySQL 8.0 Commercial Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a Community release of MySQL 8.0, see the MySQL 8.0 Community Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 1997, 2020, 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 is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.
Oracle provides a set of binary distributions of MySQL. These include generic binary distributions in the form of compressed tar files (files with a .tar.xz extension) for a number of platforms, and binaries in platform-specific package formats for selected platforms.

This section covers the installation of MySQL from a compressed tar file binary distribution on Unix/Linux platforms. For other platform-specific binary package formats, see the other platform-specific sections in this manual. For example, for Windows distributions, see Installing MySQL on Microsoft Windows. See How to Get MySQL on how to obtain MySQL in different distribution formats.

MySQL compressed tar file binary distributions have names of the form mysql-VERSION-OS.tar.xz, where VERSION is a number (for example, 8.0.22), and OS indicates the type of operating system for which the distribution is intended (for example, pc-linux-i686 or winx64).

There is also a “minimal install” version of the MySQL compressed tar file for the Linux generic binary distribution, which has a name of the form mysql-VERSION-OS-minimal.tar.xz. The minimal install distribution excludes debug binaries and is stripped of debug symbols, making it significantly smaller than the regular binary distribution. If you choose to install the minimal install distribution, remember to adjust for the difference in file name format in the instructions that follow.

Warnings

- If you have previously installed MySQL using your operating system native package management system, such as Yum or APT, you may experience problems installing using a native binary. Make sure your previous MySQL 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 for configuration files such as /etc/my.cnf or the /etc/mysql directory and delete them.

For information about replacing third-party packages with official MySQL packages, see the related APT guide or Yum guide.

- MySQL has a dependency on the libaio library. Data directory initialization and subsequent server startup steps will fail if this library is not installed locally. If necessary, install it using the appropriate package manager. For example, on Yum-based systems:

```shell
shell> yum search libaio # search for info
shell> yum install libaio # install library
```

Or, on APT-based systems:

```shell
shell> apt-cache search libaio # search for info
shell> apt-get install libaio1 # install library
```

- Oracle Linux 8 / Red Hat 8 (EL8): These platforms by default do not install the file /lib64/libtinfo.so.5, which is required by the MySQL client bin/mysql for packages mysql-VERSION-el7-x86_64.tar.gz and mysql-VERSION-linux-glibc2.12-x86_64.tar.xz. To work around this issue, install the ncurses-compat-libs package:

```shell
shell> yum install ncurses-compat-libs
```

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.
Create a mysql User and Group

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

<table>
<thead>
<tr>
<th>Directory</th>
<th>Contents of Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td><code>mysqlld</code> server, client and utility programs</td>
</tr>
<tr>
<td>docs</td>
<td>MySQL manual in Info format</td>
</tr>
<tr>
<td>man</td>
<td>Unix manual pages</td>
</tr>
<tr>
<td>include</td>
<td>Include (header) files</td>
</tr>
<tr>
<td>lib</td>
<td>Libraries</td>
</tr>
<tr>
<td>share</td>
<td>Error messages, dictionary, and SQL for database installation</td>
</tr>
<tr>
<td>support-files</td>
<td>Miscellaneous support files</td>
</tr>
</tbody>
</table>

Debug versions of the `mysqlld` binary are available as `mysqlld-debug`. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. See *Installing MySQL from Source*.

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

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
shell> cd /usr/local
shell> tar xvf /path/to/mysql-VERSION-OS.tar.xz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> mkdir mysql-files
shell> chown mysql:mysql mysql-files
shell> chmod 750 mysql-files
shell> bin/mysqlld --initialize --user=mysql
shell> bin/mysql_ssl_rsa_setup
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

**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` (Solaris) command.

The `mysql-files` directory provides a convenient location to use as the value for the `secure_file_priv` system variable, which limits import and export operations to a specific directory. See *Server System Variables*.

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

---

## Create a mysql User and Group

If your system does not already have a user and group to use for running `mysqlld`, you may need to create them. 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/Linux, or they may have different names such as `adduser` and `addgroup`.

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
```

**Note**

Because the user is required only for ownership purposes, not login purposes, the `useradd` command uses the `-r` and `-s /bin/false` options to create
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.

```shell> tar xvf /path/to/mysql-VERSION-OS.tar.xz```

The `tar` command creates a directory named `mysql-VERSION-OS`.

To install MySQL from a compressed `tar` file binary distribution, your system must have GNU XZ Utils to uncompress the distribution and a reasonable `tar` to unpack it.

```
shell> export PATH=$PATH:/usr/local/mysql/bin
```

Perform Postinstallation Setup

The remainder of the installation process involves setting distribution ownership and access permissions, initializing the data directory, starting the MySQL server, and setting up the configuration file. For instructions, see Postinstallation Setup and Testing.
Chapter 2 Starting the Server for the First Time on Windows

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the noinstall version, or if you wish to configure and test MySQL manually rather than with the MySQL Installer.

The examples in these sections assume that MySQL is installed under the default location of C:\Program Files\MySQL\MySQL Server 8.0. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the shared_memory system variable enabled. Clients can connect through shared memory by using the --protocol=MEMORY option.

For information about which server binary to run, see Selecting a MySQL Server Type.

Testing is best done from a command prompt in a console window (or “DOS window”). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

Note
The database must be initialized before MySQL can be started. For additional information about the initialization process, see Initializing the Data Directory.

To start the server, enter this command:

C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console

For a server that includes InnoDB support, you should see the messages similar to those following as it starts (the path names and sizes may differ):

InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 209715200
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started

When the server finishes its startup sequence, you should see something like this, which indicates that the server is ready to service client connections:

mysqld: ready for connections
Version: '8.0.22' socket: '' port: 3306

The server continues to write to the console any further diagnostic output it produces. You can open a new console window in which to run client programs.
If you omit the `--console` option, the server writes diagnostic output to the error log in the data directory (`C:\Program Files\MySQL\MySQL Server 8.0\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

**Note**

The initial `root` account in the MySQL grant tables has no password. After starting the server, you should set up a password for it using the instructions in *Securing the Initial MySQL Account*. 
Chapter 3 The Server Shutdown Process

The server shutdown process takes place as follows:

1. The shutdown process is initiated.

   This can occur initiated several ways. For example, a user with the `SHUTDOWN` privilege can execute a `mysqladmin shutdown` command. `mysqladmin` can be used on any platform supported by MySQL. Other operating system-specific shutdown initiation methods are possible as well: The server shuts down on Unix when it receives a `SIGTERM` signal. A server running as a service on Windows shuts down when the services manager tells it to.

2. The server creates a shutdown thread if necessary.

   Depending on how shutdown was initiated, the server might create a thread to handle the shutdown process. If shutdown was requested by a client, a shutdown thread is created. If shutdown is the result of receiving a `SIGTERM` signal, the signal thread might handle shutdown itself, or it might create a separate thread to do so. If the server tries to create a shutdown thread and cannot (for example, if memory is exhausted), it issues a diagnostic message that appears in the error log:

   Error: Can't create thread to kill server

3. The server stops accepting new connections.

   To prevent new activity from being initiated during shutdown, the server stops accepting new client connections by closing the handlers for the network interfaces to which it normally listens for connections: the TCP/IP port, the Unix socket file, the Windows named pipe, and shared memory on Windows.

4. The server terminates current activity.

   For each thread associated with a client connection, the server breaks the connection to the client and marks the thread as killed. Threads die when they notice that they are so marked. Threads for idle connections die quickly. Threads that currently are processing statements check their state periodically and take longer to die. For additional information about thread termination, see KILL Statement, in particular for the instructions about killed `REPAIR TABLE` or `OPTIMIZE TABLE` operations on `MyISAM` tables.

   For threads that have an open transaction, the transaction is rolled back. If a thread is updating a nontransactional table, an operation such as a multiple-row `UPDATE` or `INSERT` may leave the table partially updated because the operation can terminate before completion.

   If the server is a master replication server, it treats threads associated with currently connected slaves like other client threads. That is, each one is marked as killed and exits when it next checks its state.

   If the server is a slave replication server, it stops the I/O and SQL threads, if they are active, before marking client threads as killed. The SQL thread is permitted to finish its current statement (to avoid causing replication problems), and then stops. If the SQL thread is in the middle of a transaction at this point, the server waits until the current replication event group (if any) has finished executing, or until the user issues a `KILL QUERY` or `KILL CONNECTION` statement. See also STOP SLAVE Statement. Since nontransactional statements cannot be rolled back, in order to guarantee crash-safe replication, only transactional tables should be used.

   **Note**

   To guarantee crash safety on the slave, you must run the slave with `--relay-log-recovery` enabled.

   See also Replication Relay and Status Logs).
5. The server shuts down or closes storage engines.

At this stage, the server flushes the table cache and closes all open tables.

Each storage engine performs any actions necessary for tables that it manages. **InnoDB** flushes its buffer pool to disk (unless `innodb_fast_shutdown` is 2), writes the current LSN to the tablespace, and terminates its own internal threads. **MyISAM** flushes any pending index writes for a table.

6. The server exits.

To provide information to management processes, the server returns one of the exit codes described in the following list. The phrase in parentheses indicates the action taken by systemd in response to the code, for platforms on which systemd is used to manage the server.

- 0 = successful termination (no restart done)
- 1 = unsuccessful termination (no restart done)
- 2 = unsuccessful termination (restart done)
Chapter 4 Server and Server-Startup Programs

Table of Contents

4.1 mysqld — The MySQL Server ................................................................. 9
4.2 mysqld_safe — MySQL Server Startup Script ........................................... 9
4.3 mysql.server — MySQL Server Startup Script ......................................... 15
4.4 mysqld_multi — Manage Multiple MySQL Servers ................................. 17

This section describes mysqld, the MySQL server, and several programs that are used to start the server.

4.1 mysqld — The MySQL Server

mysqld, also known as MySQL Server, is the main program that does most of the work in a MySQL installation. MySQL Server manages access to the MySQL data directory that contains databases and tables. The data directory is also the default location for other information such as log files and status files.

Note
Some installation packages contain a debugging version of the server named mysqld-debug. Invoke this version instead of mysqld for debugging support, memory allocation checking, and trace file support (see Creating Trace Files).

When MySQL server starts, it listens for network connections from client programs and manages access to databases on behalf of those clients.

The mysqld program has many options that can be specified at startup. For a complete list of options, run this command:

shell> mysqld --verbose --help

MySQL Server also has a set of system variables that affect its operation as it runs. System variables can be set at server startup, and many of them can be changed at runtime to effect dynamic server reconfiguration. MySQL Server also has a set of status variables that provide information about its operation. You can monitor these status variables to access runtime performance characteristics.

For a full description of MySQL Server command options, system variables, and status variables, see The MySQL Server. For information about installing MySQL and setting up the initial configuration, see Installing and Upgrading MySQL.

4.2 mysqld_safe — MySQL Server Startup Script

mysqld_safe is the recommended way to start a mysql server on Unix. mysqld_safe adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log. A description of error logging is given later in this section.

Note
For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, mysqld_safe is not installed because it is unnecessary. For more information, see Managing MySQL Server with systemd.

mysqld_safe tries to start an executable named mysqld. To override the default behavior and specify explicitly the name of the server you want to run, specify a --mysqld or --mysqld-version option to mysqld_safe. You can also use --ledir to indicate the directory where mysqld_safe should look for the server.
Many of the options to `mysqld_safe` are the same as the options to `mysqld`. See Server Command Options.

Options unknown to `mysqld_safe` are passed to `mysqld` if they are specified on the command line, but ignored if they are specified in the `[mysqld_safe]` group of an option file. See Using Option Files.

`mysqld_safe` reads all options from the `[mysqld]`, `[server]`, and `[mysqld_safe]` sections in option files. For example, if you specify a `[mysqld]` section like this, `mysqld_safe` will find and use the `--log-error` option:

```
[mysqld]
log-error=error.log
```

For backward compatibility, `mysqld_safe` also reads `[safe_mysqld]` sections, but to be current you should rename such sections to `[mysqld_safe]`.

`mysqld_safe` accepts options on the command line and in option files, as described in the following table. For information about option files used by MySQL programs, see Using Option Files.

### Table 4.1 mysqld_safe Options

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--basedir</td>
<td>Path to MySQL installation directory</td>
</tr>
<tr>
<td>--core-file-size</td>
<td>Size of core file that mysqld should be able to create</td>
</tr>
<tr>
<td>--datadir</td>
<td>Path to data directory</td>
</tr>
<tr>
<td>--defaults-extra-file</td>
<td>Read named option file in addition to usual option files</td>
</tr>
<tr>
<td>--defaults-file</td>
<td>Read only named option file</td>
</tr>
<tr>
<td>--help</td>
<td>Display help message and exit</td>
</tr>
<tr>
<td>--ledir</td>
<td>Path to directory where server is located</td>
</tr>
<tr>
<td>--log-error</td>
<td>Write error log to named file</td>
</tr>
<tr>
<td>--malloc-lib</td>
<td>Alternative malloc library to use for mysqld</td>
</tr>
<tr>
<td>--mysqld</td>
<td>Name of server program to start (in ledir directory)</td>
</tr>
<tr>
<td>--mysqld-safe-log-timestamps</td>
<td>Timestamp format for logging</td>
</tr>
<tr>
<td>--mysqld-version</td>
<td>Suffix for server program name</td>
</tr>
<tr>
<td>--nice</td>
<td>Use nice program to set server scheduling priority</td>
</tr>
<tr>
<td>--no-defaults</td>
<td>Read no option files</td>
</tr>
<tr>
<td>--open-files-limit</td>
<td>Number of files that mysqld should be able to open</td>
</tr>
<tr>
<td>--pid-file</td>
<td>Path name of server process ID file</td>
</tr>
<tr>
<td>--plugin-dir</td>
<td>Directory where plugins are installed</td>
</tr>
<tr>
<td>--port</td>
<td>Port number on which to listen for TCP/IP connections</td>
</tr>
<tr>
<td>--skip-kill-mysqld</td>
<td>Do not try to kill stray mysqld processes</td>
</tr>
<tr>
<td>--skip-syslog</td>
<td>Do not write error messages to syslog; use error log file</td>
</tr>
<tr>
<td>--socket</td>
<td>Socket file on which to listen for Unix socket connections</td>
</tr>
<tr>
<td>--syslog</td>
<td>Write error messages to syslog</td>
</tr>
<tr>
<td>--syslog-tag</td>
<td>Tag suffix for messages written to syslog</td>
</tr>
<tr>
<td>--timezone</td>
<td>Set TZ time zone environment variable to named value</td>
</tr>
<tr>
<td>--user</td>
<td>Run mysqld as user having name user_name or numeric user ID user_id</td>
</tr>
</tbody>
</table>

- **--help**

  Display a help message and exit.
• --basedir=dir_name

The path to the MySQL installation directory.

• --core-file-size=size

The size of the core file that mysqld should be able to create. The option value is passed to ulimit -c.

Note

The innodb_buffer_pool_in_core_file variable can be used to reduce the size of core files on operating systems that support it. For more information, see Excluding Buffer Pool Pages from Core Files.

• --datadir=dir_name

The path to the data directory.

• --defaults-extra-file=file_name

Read this option file in addition to the usual option files. If the file does not exist or is otherwise inaccessible, the server will exit with an error. file_name is interpreted relative to the current directory if given as a relative path name rather than a full path name. This must be the first option on the command line if it is used.

For additional information about this and other option-file options, see Command-Line Options that Affect Option-File Handling.

• --defaults-file=file_name

Use only the given option file. If the file does not exist or is otherwise inaccessible, the server will exit with an error. file_name is interpreted relative to the current directory if given as a relative path name rather than a full path name. This must be the first option on the command line if it is used.

For additional information about this and other option-file options, see Command-Line Options that Affect Option-File Handling.

• --ledir=dir_name

If mysqld_safe cannot find the server, use this option to indicate the path name to the directory where the server is located.

This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of MYSQLD_OPTS. See Managing MySQL Server with systemd.

• --log-error=file_name

Write the error log to the given file. See The Error Log.

• --mysqld-safe-log-timestamps

This option controls the format for timestamps in log output produced by mysqld_safe. The following list describes the permitted values. For any other value, mysqld_safe logs a warning and uses UTC format.

• UTC, utc

ISO 8601 UTC format (same as --log_timestamps=UTC for the server). This is the default.

• SYSTEM, system
ISO 8601 local time format (same as --log_timestamps=SYSTEM for the server).

- **HYPHEN**, hyphen
  
  YY-MM-DD h:mm:ss format, as in mysqld_safe for MySQL 5.6.

- **LEGACY**, legacy
  
  YYMMDD hh:mm:ss format, as in mysqld_safe prior to MySQL 5.6.

- **--malloc-lib=[lib_name]**

  The name of the library to use for memory allocation instead of the system malloc() library. The option value must be one of the directories /usr/lib, /usr/lib64, /usr/lib/i386-linux-gnu, or /usr/lib/x86_64-linux-gnu.

  The --malloc-lib option works by modifying the LD_PRELOAD environment value to affect dynamic linking to enable the loader to find the memory-allocation library when mysqld runs:

  - If the option is not given, or is given without a value (--malloc-lib=), LD_PRELOAD is not modified and no attempt is made to use tcmalloc.

  - If the option is given as --malloc-lib=tcmalloc, mysqld_safe looks for a tcmalloc library in /usr/lib. If tcmalloc is found, its path name is added to the beginning of the LD_PRELOAD value for mysqld. If tcmalloc is not found, mysqld_safe aborts with an error.

  - If the option is given as --malloc-lib=/path/to/some/library, that full path is added to the beginning of the LD_PRELOAD value. If the full path points to a nonexistent or unreadable file, mysqld_safe aborts with an error.

  - For cases where mysqld_safe adds a path name to LD_PRELOAD, it adds the path to the beginning of any existing value the variable already has.

  **Note**

  On systems that manage the server using systemd, mysqld_safe is not available. Instead, specify the allocation library by setting LD_PRELOAD in /etc/sysconfig/mysql.

  Linux users can use the libtcmalloc_minimal.so library on any platform for which a tcmalloc package is installed in /usr/lib by adding these lines to the my.cnf file:

  ```
  [mysqld_safe]
  malloc-lib=tcmalloc
  
  To use a specific tcmalloc library, specify its full path name. Example:

  ```

  ```
  [mysqld_safe]
  malloc-lib=/opt/lib/libtcmalloc_minimal.so
  ```

  - **--mysqld=prog_name**

    The name of the server program (in the ledir directory) that you want to start. This option is needed if you use the MySQL binary distribution but have the data directory outside of the binary distribution. If mysqld_safe cannot find the server, use the --ledir option to indicate the path name to the directory where the server is located.

    This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of MYSQLD_OPTS. See Managing MySQL Server with systemd.

  - **--mysqld-version=suffix**
This option is similar to the `--mysqld` option, but you specify only the suffix for the server program name. The base name is assumed to be `mysqld`. For example, if you use `--mysqld-version=debug`, `mysqld_safe` starts the `mysqld-debug` program in the `ledir` directory. If the argument to `--mysqld-version` is empty, `mysqld_safe` uses `mysqld` in the `ledir` directory.

This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of `MYSQLD_OPTS`. See Managing MySQL Server with systemd.

- `--nice=priority`

  Use the `nice` program to set the server's scheduling priority to the given value.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read. This must be the first option on the command line if it is used.

  For additional information about this and other option-file options, see Command-Line Options that Affect Option-File Handling.

- `--open-files-limit=count`

  The number of files that `mysqld` should be able to open. The option value is passed to `ulimit -n`.

  **Note**

  You must start `mysqld_safe` as `root` for this to function properly.

- `--pid-file=file_name`

  The path name that `mysqld` should use for its process ID file.

- `--plugin-dir=dir_name`

  The path name of the plugin directory.

- `--port=port_num`

  The port number that the server should use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the `root` operating system user.

- `--skip-kill-mysqld`

  Do not try to kill stray `mysqld` processes at startup. This option works only on Linux.

- `--socket=path`

  The Unix socket file that the server should use when listening for local connections.

- `--syslog,--skip-syslog`

  `--syslog` causes error messages to be sent to `syslog` on systems that support the `logger` program. `--skip-syslog` suppresses the use of `syslog`; messages are written to an error log file.

  When `syslog` is used for error logging, the `daemon.err` facility/severity is used for all log messages.

  Using these options to control `mysqld` logging is deprecated. To write error log output to the system log, use the instructions at Error Logging to the System Log. To control the facility, use the server log_syslog_facility system variable.
mysqld_safe — MySQL Server Startup Script

- **--syslog-tag=tag**

  For logging to syslog, messages from mysqld_safe and mysqld are written with identifiers of mysqld_safe and mysqld, respectively. To specify a suffix for the identifiers, use **--syslog-tag=tag**, which modifies the identifiers to be mysqld_safe-tag and mysqld-tag.

  Using this option to control mysqld logging is deprecated. Use the server **log_syslog_tag** system variable instead. See Error Logging to the System Log.

- **--timezone=timezone**

  Set the **TZ** time zone environment variable to the given option value. Consult your operating system documentation for legal time zone specification formats.

- **--user={user_name|user_id}**

  Run the mysqld server as the user having the name user_name or the numeric user ID user_id. (“User” in this context refers to a system login account, not a MySQL user listed in the grant tables.)

  If you execute mysqld_safe with the **--defaults-file** or **--defaults-extra-file** option to name an option file, the option must be the first one given on the command line or the option file will not be used. For example, this command will not use the named option file:

  ```
  mysql> mysqld_safe --port=port_num --defaults-file=file_name
  ```

  Instead, use the following command:

  ```
  mysql> mysqld_safe --defaults-file=file_name --port=port_num
  ```

  The mysqld_safe script is written so that it normally can start a server that was installed from either a source or a binary distribution of MySQL, even though these types of distributions typically install the server in slightly different locations. (See Installation Layouts.) mysqld_safe expects one of the following conditions to be true:

  - The server and databases can be found relative to the working directory (the directory from which mysqld_safe is invoked). For binary distributions, mysqld_safe looks under its working directory for bin and data directories. For source distributions, it looks for libexec and var directories. This condition should be met if you execute mysqld_safe from your MySQL installation directory (for example, /usr/local/mysql for a binary distribution).

  - If the server and databases cannot be found relative to the working directory, mysqld_safe attempts to locate them by absolute path names. Typical locations are /usr/local/libexec and /usr/local/var. The actual locations are determined from the values configured into the distribution at the time it was built. They should be correct if MySQL is installed in the location specified at configuration time.

  Because mysqld_safe tries to find the server and databases relative to its own working directory, you can install a binary distribution of MySQL anywhere, as long as you run mysqld_safe from the MySQL installation directory:

  ```
  shell> cd mysql_installation_directory
  shell> bin/mysqld_safe &
  ```

  If mysqld_safe fails, even when invoked from the MySQL installation directory, specify the **--ledir** and **--datadir** options to indicate the directories in which the server and databases are located on your system.

  mysqld_safe tries to use the **sleep** and **date** system utilities to determine how many times per second it has attempted to start. If these utilities are present and the attempted starts per second is greater than 5, mysqld_safe waits 1 full second before starting again. This is intended to prevent excessive CPU usage in the event of repeated failures. (Bug #11761530, Bug #54035)
When you use `mysqld_safe` to start `mysqld`, `mysqld_safe` arranges for error (and notice) messages from itself and from `mysqld` to go to the same destination.

There are several `mysqld_safe` options for controlling the destination of these messages:

- `--log-error=file_name`: Write error messages to the named error file.
- `--syslog`: Write error messages to `syslog` on systems that support the `logger` program.
- `--skip-syslog`: Do not write error messages to `syslog`. Messages are written to the default error log file (`host_name.err` in the data directory), or to a named file if the `--log-error` option is given.

If none of these options is given, the default is `--skip-syslog`.

When `mysqld_safe` writes a message, notices go to the logging destination (`syslog` or the error log file) and `stdout`. Errors go to the logging destination and `stderr`.

---

4.3 `mysql.server` — MySQL Server Startup Script

MySQL distributions on Unix and Unix-like system include a script named `mysql.server`, which starts the MySQL server using `mysqld_safe`. It can be used on systems such as Linux and Solaris that use System V-style run directories to start and stop system services. It is also used by the macOS Startup Item for MySQL.

`mysql.server` is the script name as used within the MySQL source tree. The installed name might be different (for example, `mysqld` or `mysql`). In the following discussion, adjust the name `mysql.server` as appropriate for your system.

---

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

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

`mysql.server` changes location to the MySQL installation directory, then invokes `mysqld_safe`. To run the server as some specific user, add an appropriate `user` option to the `[mysqld]` group of the global `/etc/my.cnf` option file, as shown later in this section. (It is possible that you must 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; 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 `mysqladmin shutdown`.

To start and stop MySQL automatically on your server, you must add start and stop commands to the appropriate places in your `/etc/rc*` files:
mysql.server — MySQL Server Startup Script

- 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 mysqld or mysql. See Installing MySQL on Linux Using RPM Packages from Oracle, for more information on the Linux RPM packages.

- If you install MySQL from a source distribution or using a binary distribution format that does not install mysql.server automatically, you can install the script manually. It can be found in the support-files directory under the MySQL installation directory or in a MySQL source tree. Copy the script to the /etc/init.d directory with the name mysql and make it executable:

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

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/. Install the mysql.server script as /usr/local/etc/rc.d/mysql.server.sh to enable automatic startup. The rc(8) manual page states that scripts in this directory are executed only if their base name matches the *.sh shell file name pattern. Any other files or directories present within the directory are silently ignored.

- 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, 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.

mysql.server reads options from the [mysql.server] and [mysqld] sections of option files. For backward compatibility, it also reads [mysql_server] sections, but to be current you should rename such sections to [mysql.server].

You can add options for mysql.server in a global /etc/my.cnf file. A typical 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 options shown in the following table. 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.

### Table 4.2 mysql.server Option-File Options

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basedir</td>
<td>Path to MySQL installation directory</td>
<td>Directory name</td>
</tr>
<tr>
<td>datadir</td>
<td>Path to MySQL data directory</td>
<td>Directory name</td>
</tr>
</tbody>
</table>
### Option Table

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid-file</td>
<td>File in which server should write its process ID</td>
<td>File name</td>
</tr>
<tr>
<td>service-startup-timeout</td>
<td>How long to wait for server startup</td>
<td>Integer</td>
</tr>
</tbody>
</table>

- **basedir=dir_name**
  
  The path to the MySQL installation directory.

- **datadir=dir_name**
  
  The path to the MySQL data directory.

- **pid-file=file_name**
  
  The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.

If this option is not given, `mysql.server` uses a default value of `host_name.pid`. The PID file value passed to `mysqld_safe` overrides any value specified in the `[mysqld_safe]` option file group. Because `mysql.server` reads the `[mysqld]` option file group but not the `[mysqld_safe]` group, you can ensure that `mysqld_safe` gets the same value when invoked from `mysql.server` as when invoked manually by putting the same `pid-file` setting in both the `[mysqld_safe]` and `[mysqld]` groups.

- **service-startup-timeout=seconds**
  
  How long in seconds to wait for confirmation of server startup. If the server does not start within this time, `mysql.server` exits with an error. The default value is 900. A value of 0 means not to wait at all for startup. Negative values mean to wait forever (no timeout).

## 4.4 mysqld_multi — Manage Multiple MySQL Servers

`mysqld_multi` is designed to manage several `mysqld` processes that listen for connections on different Unix socket files and TCP/IP ports. It can start or stop servers, or report their current status.

**Note**

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, `mysqld_multi` is not installed because it is unnecessary. For information about using systemd to handle multiple MySQL instances, see Managing MySQL Server with systemd.

`mysqld_multi` searches for groups named `[mysqldN]` in `my.cnf` (or in the file named by the `--defaults-file` option). `N` can be any positive integer. This number is referred to in the following discussion as the option group number, or `GNR`. Group numbers distinguish option groups from one another and are used as arguments to `mysqld_multi` to specify which servers you want to start, stop, or obtain a status report for. Options listed in these groups are the same that you would use in the `[mysqld]` group used for starting `mysqld`. (See, for example, Starting and Stopping MySQL Automatically.) However, when using multiple servers, it is necessary that each one use its own value for options such as the Unix socket file and TCP/IP port number. For more information on which options must be unique per server in a multiple-server environment, see Running Multiple MySQL Instances on One Machine.

To invoke `mysqld_multi`, use the following syntax:

```shell
mysqld_multi [options] {start|stop|reload|report} [GNR[,GNR] ...]
```
start, stop, reload (stop and restart), and report indicate which operation to perform. You can perform the designated operation for a single server or multiple servers, depending on the GNR list that follows the option name. If there is no list, mysqld_multi performs the operation for all servers in the option file.

Each GNR value represents an option group number or range of group numbers. The value should be the number at the end of the group name in the option file. For example, the GNR for a group named [mysqld17] is 17. To specify a range of numbers, separate the first and last numbers by a dash. The GNR value 10-13 represents groups [mysqld10] through [mysqld13]. Multiple groups or group ranges can be specified on the command line, separated by commas. There must be no whitespace characters (spaces or tabs) in the GNR list; anything after a whitespace character is ignored.

This command starts a single server using option group [mysqld17]:

shell> mysqld_multi start 17

This command stops several servers, using option groups [mysqld8] and [mysqld10] through [mysqld13]:

shell> mysqld_multi stop 8,10-13

For an example of how you might set up an option file, use this command:

shell> mysqld_multi --example

mysqld_multi searches for option files as follows:

- With --no-defaults, no option files are read.
- With --defaults-file=file_name, only the named file is read.
- Otherwise, option files in the standard list of locations are read, including any file named by the --defaults-extra-file=file_name option, if one is given. (If the option is given multiple times, the last value is used.)

For additional information about these and other option-file options, see Command-Line Options that Affect Option-File Handling.

Option files read are searched for [mysqld_multi] and [mysqldN] option groups. The [mysqld_multi] group can be used for options to mysqld_multi itself. [mysqldN] groups can be used for options passed to specific mysqld instances.

The [mysqld] or [mysqld_safe] groups can be used for common options read by all instances of mysqld or mysqld_safe. You can specify a --defaults-file=file_name option to use a different configuration file for that instance, in which case the [mysqld] or [mysqld_safe] groups from that file will be used for that instance.

mysqld_multi supports the following options:

- --help
  Display a help message and exit.
- --example
  Display a sample option file.
- --log=file_name
  Specify the name of the log file. If the file exists, log output is appended to it.
- --mysqladmin=prog_name
The `mysqladmin` binary to be used to stop servers.

- **--mysqld=prog_name**
  The `mysqld` binary to be used. Note that you can specify `mysqld_safe` as the value for this option also. If you use `mysqld_safe` to start the server, you can include the `mysqld` or `ledir` options in the corresponding `[mysqldN]` option group. These options indicate the name of the server that `mysqld_safe` should start and the path name of the directory where the server is located. (See the descriptions for these options in Section 4.2, "mysqld_safe — MySQL Server Startup Script"). Example:
  ```
  [mysqld38]
  mysqld = mysqld-debug
  ledir = /opt/local/mysql/libexec
  ```

- **--no-log**
  Print log information to `stdout` rather than to the log file. By default, output goes to the log file.

- **--password=password**
  The password of the MySQL account to use when invoking `mysqladmin`. Note that the password value is not optional for this option, unlike for other MySQL programs.

- **--silent**
  Silent mode; disable warnings.

- **--tcp-ip**
  Connect to each MySQL server through the TCP/IP port instead of the Unix socket file. (If a socket file is missing, the server might still be running, but accessible only through the TCP/IP port.) By default, connections are made using the Unix socket file. This option affects `stop` and `report` operations.

- **--user=user_name**
  The user name of the MySQL account to use when invoking `mysqladmin`.

- **--verbose**
  Be more verbose.

- **--version**
  Display version information and exit.

Some notes about `mysqld_multi`:

- **Most important**: Before using `mysqld_multi` be sure that you understand the meanings of the options that are passed to the `mysqld` servers and why you would want to have separate `mysqld` processes. Beware of the dangers of using multiple `mysqld` servers with the same data directory. Use separate data directories, unless you know what you are doing. Starting multiple servers with the same data directory does not give you extra performance in a threaded system. See Running Multiple MySQL Instances on One Machine.

  **Important**
  Make sure that the data directory for each server is fully accessible to the Unix account that the specific `mysqld` process is started as. Do not use the Unix `root` account for this, unless you know what you are doing. See How to Run MySQL as a Normal User.
• Make sure that the MySQL account used for stopping the mysqld servers (with the mysqladmin program) has the same user name and password for each server. Also, make sure that the account has the SHUTDOWN privilege. If the servers that you want to manage have different user names or passwords for the administrative accounts, you might want to create an account on each server that has the same user name and password. For example, you might set up a common multi_admin account by executing the following commands for each server:

```shell
csh> mysql -u root -S /tmp/mysql.sock -p
Enter password:
mysql> CREATE USER 'multi_admin'@'localhost' IDENTIFIED BY 'multipass';
mysql> GRANT SHUTDOWN ON *.* TO 'multi_admin'@'localhost';
```

See Access Control and Account Management. You have to do this for each mysqld server. Change the connection parameters appropriately when connecting to each one. Note that the host name part of the account name must permit you to connect as multi_admin from the host where you want to run mysqld_multi.

• The Unix socket file and the TCP/IP port number must be different for every mysqld. (Alternatively, if the host has multiple network addresses, you can set the bind_address system variable to cause different servers to listen to different interfaces.)

• The --pid-file option is very important if you are using mysqld_safe to start mysqld (for example, --mysqld=mysqld_safe) Every mysqld should have its own process ID file. The advantage of using mysqld_safe instead of mysqld is that mysqld_safe monitors its mysqld process and restarts it if the process terminates due to a signal sent using kill -9 or for other reasons, such as a segmentation fault.

• You might want to use the --user option for mysqld, but to do this you need to run the mysqld_multi script as the Unix superuser (root). Having the option in the option file doesn't matter; you just get a warning if you are not the superuser and the mysqld processes are started under your own Unix account.

The following example shows how you might set up an option file for use with mysqld_multi. The order in which the mysqld programs are started or stopped depends on the order in which they appear in the option file. Group numbers need not form an unbroken sequence. The first and fifth [mysqld] groups were intentionally omitted from the example to illustrate that you can have “gaps” in the option file. This gives you more flexibility.

```bash
# This is an example of a my.cnf file for mysqld_multi.
# Usually this file is located in home dir ~/.my.cnf or /etc/my.cnf
[mysqld_multi]
mysqld   = /usr/local/mysql/bin/mysqld_safe
mysqladmin = /usr/local/mysql/bin/mysqladmin
user      = multi_admin
password  = my_password

[mysqld2]
socket    = /tmp/mysql.sock2
port      = 3307
pid-file  = /usr/local/mysql/data2/hostname.pid2
datadir   = /usr/local/mysql/data2
language  = /usr/local/mysql/share/mysql/english
user      = unix_user1

[mysqld3]
mysqld   = /path/to/mysqld_safe
basedir  = /path/to/mysqld-binary/
mysqladmin = /path/to/mysqladmin
socket    = /tmp/mysql.sock3
port      = 3308
pid-file  = /usr/local/mysql/data3/hostname.pid3
datadir   = /usr/local/mysql/data3
language  = /usr/local/mysql/share/mysql/swedish
user      = unix_user2

[mysqld4]
socket    = /tmp/mysql.sock4
port      = 3309
pid-file  = /usr/local/mysql/data4/hostname.pid4
```
<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datadir</td>
<td>/usr/local/mysql/data4</td>
</tr>
<tr>
<td>language</td>
<td>/usr/local/mysql/share/mysql/estonia</td>
</tr>
<tr>
<td>user</td>
<td>unix_user3</td>
</tr>
<tr>
<td>[mysql8]</td>
<td></td>
</tr>
<tr>
<td>socket</td>
<td>/tmp/mysql.sock8</td>
</tr>
<tr>
<td>port</td>
<td>3311</td>
</tr>
<tr>
<td>pid-file</td>
<td>/usr/local/mysql/data8/hostname.pid8</td>
</tr>
<tr>
<td>datadir</td>
<td>/usr/local/mysql/data8</td>
</tr>
<tr>
<td>language</td>
<td>/usr/local/mysql/share/mysql/japanese</td>
</tr>
<tr>
<td>user</td>
<td>unix_user4</td>
</tr>
</tbody>
</table>

See Using Option Files.