NAME

mysql - the MySQL command-line tool

SYNOPSIS

mysql [*options*] *db_name*

DESCRIPTION

mysql is a simple SQL shell with input line editing capabilities. It supports interactive and noninteractive use. When used interactively, query results are presented in an ASCII-table format. When used noninteractively (for example, as a filter), the result is presented in tab-separated format. The output format can be changed using command options.

If you have problems due to insufficient memory for large result sets, use the **--quick** option. This forces **mysql** to retrieve results from the server a row at a time rather than retrieving the entire result set and buffering it in memory before displaying it. This is done by returning the result set using the mysql_use_result() C API function in the client/server library rather than mysql_store_result().

Using **mysql** is very easy. Invoke it from the prompt of your command interpreter as follows:

shell> **mysql** *db_name*

Or:

shell> **mysql --user=***user_name* **--password** *db_name*
Enter password: *your_password*

Then type an SQL statement, end it with ;, g, or G and press Enter.

Typing Control+C causes **mysql** to attempt to kill the current statement. If this cannot be done, or Control+C is typed again before the statement is killed, **mysql** exits.

You can execute SQL statements in a script file (batch file) like this:

shell> **mysql** *db_name* < *script.sql* > *output.tab*

On Unix, the **mysql** client logs statements executed interactively to a history file. See the section called "MYSQL LOGGING".

MYSQL OPTIONS

mysql supports the following options, which can be specified on the command line or in the [mysql] and [client] groups of an option file. For information about option files used by MySQL programs, see Section 4.2.6, "Using Option Files".

- **--help**, **-?**

  Display a help message and exit.

- **--auto-rehash**

  Enable automatic rehashing. This option is on by default, which enables database, table, and column name completion. Use **--disable-auto-rehash** to disable rehashing. That causes **mysql** to start faster, but you must issue the rehash command or its # shortcut if you want to use name completion.

  To complete a name, enter the first part and press Tab. If the name is unambiguous, **mysql** completes it. Otherwise, you can press Tab again to see the possible names that begin with what you have typed so far. Completion does not occur if there is no default database.

  Note

  This feature requires a MySQL client that is compiled with the **readline** library. Typically, the **readline** library is not available on Windows.

- **--auto-vertical-output**

  Cause result sets to be displayed vertically if they are too wide for the current window, and using normal tabular format otherwise. (This applies to statements terminated by ; or G.)

This option was added in MySQL 5.5.3.

- **--batch**, **-B**

  Print results using tab as the column separator, with each row on a new line. With this option, **mysql** does not use the history file.

  Batch mode results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the **--raw** option.
- **--binary-as-hex**

  When this option is given, **mysql** displays binary data using hexadecimal notation (0x*value*). This occurs whether the overall output dislay format is tabular, vertical, HTML, or XML.

  This option was added in MySQL 5.5.57.
- **--bind-address**=*ip_address*

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

  This option is supported only in the version of the **mysql** client that is supplied with NDB Cluster. It is not available in standard MySQL Server 5.5 releases.
- **--character-sets-dir**=*dir_name*

  The directory where character sets are installed. See Section 10.14, “Character Set Configuration”.
- **--column-names**

  Write column names in results.
- **--column-type-info**

  Display result set metadata.
- **--comments**, **-c**

  Whether to strip or preserve comments in statements sent to the server. The default is **--skip-comments** (strip comments), enable with **--comments** (preserve comments).
- **--compress**, **-C**

  Compress all information sent between the client and the server if both support compression.
- **--database**=*db_name*, **-D** *db_name*

  The database to use. This is useful primarily in an option file.
- **--debug**[=*debug_options*], **-#** [*debug_options*]

  Write a debugging log. A typical *debug_options* string is d:t:o,*file_name*. The default is d:t:o,/tmp/mysql.trace.

  This option is available only if MySQL was built using **WITH_DEBUG**. MySQL release binaries provided by Oracle are *not* built using this option.
- **--debug-check**

  Print some debugging information when the program exits.
- **--debug-info**, **-T**

  Print debugging information and memory and CPU usage statistics when the program exits.
- **--default-auth**=*plugin*

  A hint about the client-side authentication plugin to use. See Section 6.3.6, “Pluggable Authentication”.

  This option was added in MySQL 5.5.7.
- **--default-character-set**=*charset_name*

Use *charset_name* as the default character set for the client and connection.

This option can be useful if the operating system uses one character set and the **mysql** client by default uses another. In this case, output may be formatted incorrectly. You can usually fix such issues by using this option to force the client to use the system character set instead.

For more information, see Section 10.4, "Connection Character Sets and Collations", and Section 10.14, "Character Set Configuration".

- **--defaults-extra-file=***file_name*

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. Before MySQL 5.5.8, *file_name* must be the full path name to the file. As of MySQL 5.5.8, the name is interpreted relative to the current directory if given as a relative path name.
- **--defaults-file=***file_name*

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. Before MySQL 5.5.8, *file_name* must be the full path name to the file. As of MySQL 5.5.8, the name is interpreted relative to the current directory if given as a relative path name.
- **--defaults-group-suffix=***str*

  Read not only the usual option groups, but also groups with the usual names and a suffix of *str*. For example, **mysql** normally reads the [client] and [mysql] groups. If the **--defaults-group-suffix=_other** option is given, **mysql** also reads the [client_other] and [mysql_other] groups.
- **--delimiter=***str*

  Set the statement delimiter. The default is the semicolon character (;).
- **--disable-named-commands**

  Disable named commands. Use the * form only, or use named commands only at the beginning of a line ending with a semicolon (;). **mysql** starts with this option *enabled* by default. However, even with this option, long-format commands still work from the first line. See the section called "MYSQL COMMANDS".
- **--enable-cleartext-plugin**

  Enable the mysql_clear_password cleartext authentication plugin. (See Section 6.5.1.3, "Client-Side Cleartext Pluggable Authentication".) This option was added in MySQL 5.5.27.
- **--execute=***statement*, **-e** *statement*

  Execute the statement and quit. The default output format is like that produced with **--batch**. See Section 4.2.4, "Using Options on the Command Line", for some examples. With this option, **mysql** does not use the history file.
- **--force**, **-f**

  Continue even if an SQL error occurs.
- **--host=***host_name*, **-h** *host_name*

  Connect to the MySQL server on the given host.
- **--html**, **-H**

  Produce HTML output.
- **--ignore-spaces**, **-i**

  Ignore spaces after function names. The effect of this is described in the discussion for the IGNORE_SPACE SQL mode (see Section 5.1.10, "Server SQL Modes").
- **--init-command=str**

  SQL statement to execute after connecting to the server. If auto-reconnect is enabled, the statement is executed again after reconnection occurs.

- **--line-numbers**

  Write line numbers for errors. Disable this with **--skip-line-numbers**.
- **--local-infile**[={**0**|**1**}]

  Enable or disable LOCAL capability for LOAD DATA INFILE. For **mysql**, this capability is disabled by default. With no value, the option enables LOCAL. The option may be given as **--local-infile=0** or **--local-infile=1** to explicitly disable or enable LOCAL. Enabling local data loading also requires that the server permits it; see Section 6.1.6, "Security Issues with LOAD DATA LOCAL"
- **--named-commands**, **-G**

  Enable named **mysql** commands. Long-format commands are permitted, not just short-format commands. For example, quit and q both are recognized. Use **--skip-named-commands** to disable named commands. See the section called "MYSQL COMMANDS".
- **--no-auto-rehash**, **-A**

  This has the same effect as **--skip-auto-rehash**. See the description for **--auto-rehash**.
- **--no-beep**, **-b**

  Do not beep when errors occur.
- **--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.
- **--no-named-commands**, **-g**

  Deprecated, use **--disable-named-commands** instead. **--no-named-commands** was removed in MySQL 5.5.3.
- **--no-pager**

  Deprecated form of **--skip-pager**. See the **--pager** option. **--no-pager** was removed in MySQL 5.5.3.
- **--no-tee**

  Deprecated form of **--skip-tee**. See the **--tee** option. **--no-tee** is removed in MySQL 5.5.3.
- **--one-database**, **-o**

  Ignore statements except those that occur while the default database is the one named on the command line. This option is rudimentary and should be used with care. Statement filtering is based only on USE statements.

  Initially, **mysql** executes statements in the input because specifying a database *db_name* on the command line is equivalent to inserting USE *db_name* at the beginning of the input. Then, for each USE statement encountered, **mysql** accepts or rejects following statements depending on whether the database named is the one on the command line. The content of the statements is immaterial.

  Suppose that **mysql** is invoked to process this set of statements:

  ```
  DELETE FROM db2.t2;
  USE db2;
  DROP TABLE db1.t1;
  CREATE TABLE db1.t1 (i INT);
  USE db1;
  INSERT INTO t1 (i) VALUES(1)
  CREATE TABLE db2.t1 (j INT);
  ```

  If the command line is **mysql --force --one-database db1**, **mysql** handles the input as follows:

  - The DELETE statement is executed because the default database is db1, even though the statement names a table in a different database.
  - The DROP TABLE and CREATE TABLE statements are not executed because the default database is not db1, even though the statements name a table in db1.
  - The INSERT and CREATE TABLE statements are executed because the default database is db1, even though the CREATE TABLE statement names a table in a different database.
- **--pager**[=*command*]

  Use the given command for paging query output. If the command is omitted, the default pager is the value of your PAGER environment variable. Valid pagers are **less**, **more**, **cat** [> **filename**], and so forth. This option works only on Unix and only in interactive mode. To disable paging, use **--skip-pager**. the section called "MYSQL COMMANDS", discusses output paging further.
- **--password**[=*password*], **-p**[*password*]

  The password to use when connecting to the server. If you use the short option form (**-p**), you *cannot* have a space between the option and the password. If you omit the *password* value following the **--password** or **-p** option on the command line, **mysql** prompts for one.

  Specifying a password on the command line should be considered insecure. See Section 6.1.2.1, "End-User Guidelines for Password Security". You can use an option file to avoid giving the password on the command line.
- **--pipe**, **-W**

  On Windows, connect to the server using a named pipe. This option applies only if the server supports named-pipe connections.
- **--plugin-dir**=*dir_name*

  The directory in which to look for plugins. Specify this option if the **--default-auth** option is used to specify an authentication plugin but **mysql** does not find it. See Section 6.3.6, "Pluggable Authentication".

  This option was added in MySQL 5.5.7.
- **--port**=*port_num*, **-P** *port_num*

  The TCP/IP port number to use for the connection.
- **--print-defaults**

  Print the program name and all options that it gets from option files.
- **--prompt**=*format_str*

  Set the prompt to the specified format. The default is mysql>. The special sequences that the prompt can contain are described in the section called "MYSQL COMMANDS".
- **--protocol**={**TCP**|**SOCKET**|**PIPE**|**MEMORY**}

  The connection protocol to use for connecting to the server. It is useful when the other connection parameters normally would cause a protocol to be used other than the one you want. For details on the permissible values, see Section 4.2.2, "Connecting to the MySQL Server".
- **--quick**, **-q**

  Do not cache each query result, print each row as it is received. This may slow down the server if the output is suspended. With this option, **mysql** does not use the history file.
- **--raw**, **-r**

  For tabular output, the "boxing" around columns enables one column value to be distinguished from another. For nontabular output (such as is produced in batch mode or when the **--batch** or **--silent** option is given), special characters are escaped in the output so they can be identified easily. Newline, tab, NUL, and backslash are written as n, t, 0, and . The **--raw** option disables this character escaping.

The following example demonstrates tabular versus nontabular output and the use of raw mode to disable escaping:

```
% mysql
mysql> SELECT CHAR(92)
+----------+
| CHAR(92) |
+----------+
| |
+----------+
% mysql -s
mysql> SELECT CHAR(92)
CHAR(92)

% mysql -s -r
mysql> SELECT CHAR(92)
CHAR(92)
```

- **--reconnect**

  If the connection to the server is lost, automatically try to reconnect. A single reconnect attempt is made each time the connection is lost. To suppress reconnection behavior, use **--skip-reconnect**.
- **--safe-updates**, **--i-am-a-dummy**, **-U**

  Permit only those UPDATE and DELETE statements that specify which rows to modify by using key values. If you have set this option in an option file, you can override it by using **--safe-updates** on the command line. See the section called "MYSQL TIPS", for more information about this option.
- **--secure-auth**

  Do not send passwords to the server in old (pre-4.1) format. This prevents connections except for servers that use the newer password format.

  Note

  Passwords that use the pre-4.1 hashing method are less secure than passwords that use the native password hashing method and should be avoided.
- **--shared-memory-base-name=***name*

  On Windows, the shared-memory name to use, for connections made using shared memory to a local server. The default value is MYSQL. The shared-memory name is case-sensitive.

  The server must be started with the **--shared-memory** option to enable shared-memory connections.
- **--show-warnings**

  Cause warnings to be shown after each statement if there are any. This option applies to interactive and batch mode.
- **--sigint-ignore**

  Ignore SIGINT signals (typically the result of typing Control+C).
- **--silent**, **-s**

  Silent mode. Produce less output. This option can be given multiple times to produce less and less output.

  This option results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the **--raw** option.
- **--skip-column-names**, **-N**

  Do not write column names in results.

- **--skip-line-numbers**, **-L**

  Do not write line numbers for errors. Useful when you want to compare result files that include error messages.
- **--socket**=*path*, **-S** *path*

  For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
- **--ssl***

  Options that begin with **--ssl** specify whether to connect to the server using SSL and indicate where to find SSL keys and certificates. See Section 6.4.2, “Command Options for Encrypted Connections”.
- **--table**, **-t**

  Display output in table format. This is the default for interactive use, but can be used to produce table output in batch mode.
- **--tee**=*file_name*

  Append a copy of output to the given file. This option works only in interactive mode. the section called “MYSQL COMMANDS”, discusses tee files further.
- **--unbuffered**, **-n**

  Flush the buffer after each query.
- **--user**=*user_name*, **-u** *user_name*

  The MySQL user name to use when connecting to the server.
- **--verbose**, **-v**

  Verbose mode. Produce more output about what the program does. This option can be given multiple times to produce more and more output. (For example, **-v -v -v** produces table output format even in batch mode.)
- **--version**, **-V**

  Display version information and exit.
- **--vertical**, **-E**

  Print query output rows vertically (one line per column value). Without this option, you can specify vertical output for individual statements by terminating them with G.
- **--wait**, **-w**

  If the connection cannot be established, wait and retry instead of aborting.
- **--xml**, **-X**

  Produce XML output.

  <field name=*column_name*>NULL</field>

  The output when **--xml** is used with **mysql** matches that of **mysqldump --xml**. See **mysqldump(1)**, for details.

  The XML output also uses an XML namespace, as shown here:

  shell> **mysql --xml -uroot -e SHOW VARIABLES LIKE version%**
  <?xml version=1.0?>
  <resultset statement=SHOW VARIABLES LIKE version% > -P xmlns:xsi= -- http://www.w3.org/2001
  <row>
  <field name=Variable_name>version</field>
  <field name=Value>5.0.40-debug</field>
  </row>
  <row>
  <field name=Variable_name>version_comment</field>
  <field name=Value>Source distribution</field>
  </row>

<row>
<field name=Variable_name>version_compile_machine</field>
<field name=Value>i686</field>
</row>
<row>
<field name=Variable_name>version_compile_os</field>
<field name=Value>suse-linux-gnu</field>
</row>
</resultset>

(See Bug #25946.)

You can also set the following variables by using **--***var_name*=*value*. The **--set-variable** format is deprecated and was removed in MySQL 5.5.3.

- connect_timeout

  The number of seconds before connection timeout. (Default value is 0.)
- max_allowed_packet

  The maximum size of the buffer for client/server communication. The default is 16MB, the maximum is 1GB.
- max_join_size

  The automatic limit for rows in a join when using **--safe-updates**. (Default value is 1,000,000.)
- net_buffer_length

  The buffer size for TCP/IP and socket communication. (Default value is 16KB.)
- select_limit

  The automatic limit for SELECT statements when using **--safe-updates**. (Default value is 1,000.)

MYSQL COMMANDS

mysql sends each SQL statement that you issue to the server to be executed. There is also a set of commands that **mysql** itself interprets. For a list of these commands, type help or h at the mysql> prompt:

mysql> **help**
List of all MySQL commands:
Note that all text commands must be first on line and end with ;
? (?) Synonym for ‘help.
clear (c) Clear command.
connect (r) Reconnect to the server. Optional arguments are db and host.
delimiter (d) Set statement delimiter.
edit (e) Edit command with $EDITOR.
ego (G) Send command to mysql server, display result vertically.
exit (q) Exit mysql. Same as quit.
go (g) Send command to mysql server.
help (h) Display this help.
nopager (n) Disable pager, print to stdout.
notee (t) Dont write into outfile.
pager (P) Set PAGER [to_pager]. Print the query results via PAGER.
print (p) Print current command.
prompt (R) Change your mysql prompt.
quit (q) Quit mysql.
rehash (#) Rebuild completion hash.
source (.) Execute an SQL script file. Takes a file name as an argument.
status (s) Get status information from the server.
system (!) Execute a system shell command.

tee (T) Set outfile [to_outfile]. Append everything into given
outfile.
use (u) Use another database. Takes database name as argument.
charset (C) Switch to another charset. Might be needed for processing
binlog with multi-byte charsets.
warnings (W) Show warnings after every statement.
nowarning (w) Dont show warnings after every statement.
For server side help, type help contents

Each command has both a long and short form. The long form is not case-sensitive; the short form is. The long form can be followed by an optional semicolon terminator, but the short form should not.

The use of short-form commands within multiple-line /* ... */ comments is not supported.

- help [*arg*], h [*arg*], ? [*arg*], ? [*arg*]

  Display a help message listing the available **mysql** commands.

  If you provide an argument to the help command, **mysql** uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. For more information, see the section called "MYSQL SERVER-SIDE HELP".
- charset *charset_name*, C *charset_name*

  Change the default character set and issue a SET NAMES statement. This enables the character set to remain synchronized on the client and server if **mysql** is run with auto-reconnect enabled (which is not recommended), because the specified character set is used for reconnects.
- clear, c

  Clear the current input. Use this if you change your mind about executing the statement that you are entering.
- connect [*db_name host_name*]], r [*db_name host_name*]]

  Reconnect to the server. The optional database name and host name arguments may be given to specify the default database or the host where the server is running. If omitted, the current values are used.
- delimiter *str*, d *str*

  Change the string that **mysql** interprets as the separator between SQL statements. The default is the semicolon character (;).

  The delimiter string can be specified as an unquoted or quoted argument on the delimiter command line. Quoting can be done with either single quote (), double quote (), or backtick (‘) characters. To include a quote within a quoted string, either quote the string with a different quote character or escape the quote with a backslash () character. Backslash should be avoided outside of quoted strings because it is the escape character for MySQL. For an unquoted argument, the delimiter is read up to the first space or end of line. For a quoted argument, the delimiter is read up to the matching quote on the line.

  mysql interprets instances of the delimiter string as a statement delimiter anywhere it occurs, except within quoted strings. Be careful about defining a delimiter that might occur within other words. For example, if you define the delimiter as X, you will be unable to use the word INDEX in statements. **mysql** interprets this as INDE followed by the delimiter X.

  When the delimiter recognized by **mysql** is set to something other than the default of ;, instances of that character are sent to the server without interpretation. However, the server itself still interprets ; as a statement delimiter and processes statements accordingly. This behavior on the server side comes into play for multiple-statement execution (see Section 23.8.16, "C API Multiple Statement Execution Support"), and for parsing the body of stored procedures and functions, triggers, and events (see Section 20.1, "Defining

Stored Programs").

- edit, e

  Edit the current input statement. **mysql** checks the values of the EDITOR and VISUAL environment variables to determine which editor to use. The default editor is **vi** if neither variable is set.

  The edit command works only in Unix.
- ego, G

  Send the current statement to the server to be executed and display the result using vertical format.
- exit, q

  Exit **mysql**.
- go, g

  Send the current statement to the server to be executed.
- nopager, n

  Disable output paging. See the description for pager.

  The nopager command works only in Unix.
- notee, t

  Disable output copying to the tee file. See the description for tee.
- nowarning, w

  Disable display of warnings after each statement.
- pager [*command*], P [*command*]

  Enable output paging. By using the **--pager** option when you invoke **mysql**, it is possible to browse or search query results in interactive mode with Unix programs such as **less**, **more**, or any other similar program. If you specify no value for the option, **mysql** checks the value of the PAGER environment variable and sets the pager to that. Pager functionality works only in interactive mode.

  Output paging can be enabled interactively with the pager command and disabled with nopager. The command takes an optional argument; if given, the paging program is set to that. With no argument, the pager is set to the pager that was set on the command line, or stdout if no pager was specified.

  Output paging works only in Unix because it uses the popen() function, which does not exist on Windows. For Windows, the tee option can be used instead to save query output, although it is not as convenient as pager for browsing output in some situations.
- print, p

  Print the current input statement without executing it.
- prompt [*str*], R [*str*]

  Reconfigure the **mysql** prompt to the given string. The special character sequences that can be used in the prompt are described later in this section.

  If you specify the prompt command with no argument, **mysql** resets the prompt to the default of mysql>.
- quit, q

  Exit **mysql**.
- rehash, #

  Rebuild the completion hash that enables database, table, and column name completion while you are entering statements. (See the description for the **--auto-rehash** option.)
- source *file_name*, . *file_name*

  Read the named file and executes the statements contained therein. On Windows, you can

specify path name separators as / or .

Quote characters are taken as part of the file name itself. For best results, the name should not include space characters.

- status, s

  Provide status information about the connection and the server you are using. If you are running in **--safe-updates** mode, status also prints the values for the **mysql** variables that affect your queries.
- system *command*, ! *command*

  Execute the given command using your default command interpreter.

  The system command works only in Unix.
- tee [*file_name*], T [*file_name*]

  By using the **--tee** option when you invoke **mysql**, you can log statements and their output. All the data displayed on the screen is appended into a given file. This can be very useful for debugging purposes also. **mysql** flushes results to the file after each statement, just before it prints its next prompt. Tee functionality works only in interactive mode.

  You can enable this feature interactively with the tee command. Without a parameter, the previous file is used. The tee file can be disabled with the notee command. Executing tee again re-enables logging.
- use *db_name*, u *db_name*

  Use *db_name* as the default database.
- warnings, W

  Enable display of warnings after each statement (if there are any).

Here are a few tips about the pager command:

- You can use it to write to a file and the results go only to the file:

  mysql> **pager cat > /tmp/log.txt**

  You can also pass any options for the program that you want to use as your pager:

  mysql> **pager less -n -i -S**
- In the preceding example, note the **-S** option. You may find it very useful for browsing wide query results. Sometimes a very wide result set is difficult to read on the screen. The **-S** option to **less** can make the result set much more readable because you can scroll it horizontally using the left-arrow and right-arrow keys. You can also use **-S** interactively within **less** to switch the horizontal-browse mode on and off. For more information, read the **less** manual page:

  shell> **man less**
- The **-F** and **-X** options may be used with **less** to cause it to exit if output fits on one screen, which is convenient when no scrolling is necessary:

  mysql> **pager less -n -i -S -F -X**
- You can specify very complex pager commands for handling query output:

  mysql> **pager cat | tee /dr1/tmp/res.txt**
  | tee /dr2/tmp/res2.txt | less -n -i -S

  In this example, the command would send query results to two files in two different directories on two different file systems mounted on /dr1 and /dr2, yet still display the results onscreen using **less**.

You can also combine the tee and pager functions. Have a tee file enabled and pager set to **less**, and you are able to browse the results using the **less** program and still have everything appended into a file the same time. The difference between the Unix tee used with the pager command and the **mysql** built-in tee command is that the built-in tee works even if you do not have the Unix

tee available. The built-in tee also logs everything that is printed on the screen, whereas the Unix **tee** used with pager does not log quite that much. Additionally, tee file logging can be turned on and off interactively from within **mysql**. This is useful when you want to log some queries to a file, but not others.

The prompt command reconfigures the default mysql> prompt. The string for defining the prompt can contain the following special sequences.

b
.br
.br
12230

Option	Description
	A counter that increments for each statement you issue
	The full current date
	The default database
	The server host
	The current delimiter
	Minutes of the current time
	A newline character
	The current month in three-letter format (Jan, Feb, ...)
	The current month in numeric format
P	am/pm
	The current TCP/IP port or socket file
	The current time, in 24-hour military time (0–23)
	The current time, standard 12-hour time (1–12)
	Semicolon
	Seconds of the current time
	A tab character
U	Your full *user_name*@*host_name* account name
	Your user name
	The server version
	The current day of the week in three-letter format (Mon, Tue, ...)
	The current year, four digits
y	The current year, two digits
_	A space
	A space (a space follows the backslash)
´	Single quote
	Double quote
T}:T{ A literal backslash character	
fIx	*x*, for any "*x*" not listed above

You can set the prompt in several ways:

- *Use an environment variable.* Y ou can set the MYSQL_PS1 environment variable to a prompt string. For example:

  shell> **export MYSQL_PS1=(u@h) [d]>**
- *Use a command-line option.* Y ou can set the **--prompt** option on the command line to **mysql**. For example:

shell> **mysql --prompt=(u@h) [d]>**
(user@host) [database]>

- *Use an option file.* Y ou can set the prompt option in the [mysql] group of any MySQL option file, such as /etc/my.cnf or the .my.cnf file in your home directory. For example:

  [mysql]
  prompt=(u@h) [d]>_

  In this example, note that the backslashes are doubled. If you set the prompt using the prompt option in an option file, it is advisable to double the backslashes when using the special prompt options. There is some overlap in the set of permissible prompt options and the set of special escape sequences that are recognized in option files. (The rules for escape sequences in option files are listed in Section 4.2.6, "Using Option Files".) The overlap may cause you problems if you use single backslashes. For example, s is interpreted as a space rather than as the current seconds value. The following example shows how to define a prompt within an option file to include the current time in HH:MM:SS> format:

  [mysql]
  prompt=r:m:s>
- *Set the prompt interactively.* You can change your prompt interactively by using the prompt (or R) command. For example:

  mysql> **prompt (u@h) [d]>_**
  PROMPT set to (u@h) [d]>_
  (*user*@*host*) [*database*]>
  (*user*@*host*) [*database*]> prompt
  Returning to default PROMPT of mysql>
  mysql>

MYSQL LOGGING

On Unix, the **mysql** client logs statements executed interactively to a history file. By default, this file is named .mysql_history in your home directory. To specify a different file, set the value of the MYSQL_HISTFILE environment variable. How Logging Occurs.PP Statement logging occurs as follows:

- Statements are logged only when executed interactively. Statements are noninteractive, for example, when read from a file or a pipe. It is also possible to suppress statement logging by using the **--batch** or **--execute** option.
- **mysql** logs each nonempty statement line individually.
- If a statement spans multiple lines (not including the terminating delimiter), **mysql** concatenates the lines to form the complete statement, maps newlines to spaces, and logs the result, plus a delimiter.

Consequently, an input statement that spans multiple lines can be logged twice. Consider this input:

mysql> **SELECT**
-> **Today is**
-> **,**
-> **CURDATE()**
-> **;**

In this case, **mysql** logs the "SELECT", "Today is", ",", "CURDATE()", and ";" lines as it reads them. It also logs the complete statement, after mapping SELECTnToday isn,nCURDATE() to SELECT Today is , CURDATE(), plus a delimiter. Thus, these lines appear in logged output:

SELECT
Today is
,
CURDATE()
;

SELECT Today is , CURDATE();

Controlling the History File.PP The .mysql_history file should be protected with a restrictive access mode because sensitive information might be written to it, such as the text of SQL statements that contain passwords. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

If you do not want to maintain a history file, first remove .mysql_history if it exists. Then use either of the following techniques to prevent it from being created again:

- Set the MYSQL_HISTFILE environment variable to /dev/null. To cause this setting to take effect each time you log in, put it in one of your shells startup files.
- Create .mysql_history as a symbolic link to /dev/null; this need be done only once:

  shell> **ln -s /dev/null $HOME/.mysql_history**

MYSQL SERVER–SIDE HELP

mysql> **help** *search_string*

If you provide an argument to the help command, **mysql** uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. The proper operation of this command requires that the help tables in the mysql database be initialized with help topic information (see Section 5.1.13, “Server-Side Help”).

If there is no match for the search string, the search fails:

mysql> **help me**
Nothing found
Please try to run help contents for a list of all accessible topics

Use **help contents** to see a list of the help categories:

mysql> **help contents**
You asked for help about help category: Contents
For more information, type help <item>, where <item> is one of the
following categories:
Account Management
Administration
Data Definition
Data Manipulation
Data Types
Functions
Functions and Modifiers for Use with GROUP BY
Geographic Features
Language Structure
Plugins
Storage Engines
Stored Routines
Table Maintenance
Transactions
Triggers

If the search string matches multiple items, **mysql** shows a list of matching topics:

mysql> **help logs**
Many help items for your request exist.
To make a more specific request, please type help <item>,
where <item> is one of the following topics:
SHOW
SHOW BINARY LOGS
SHOW ENGINE
SHOW LOGS

Use a topic as the search string to see the help entry for that topic:

```
mysql> help show binary logs
Name: SHOW BINARY LOGS
Description:
Syntax:
SHOW BINARY LOGS
SHOW MASTER LOGS
Lists the binary log files on the server. This statement is used as
part of the procedure described in [purge-binary-logs], that shows how
to determine which logs can be purged.
mysql> SHOW BINARY LOGS;
+---------------+-----------+
| Log_name | File_size |
+---------------+-----------+
| binlog.000015 | 724935 |
| binlog.000016 | 733481 |
+---------------+-----------+
```

The search string can contain the wildcard characters % and _. These have the same meaning as for pattern-matching operations performed with the LIKE operator. For example, HELP rep% returns a list of topics that begin with rep:

```
mysql> HELP rep%
Many help items for your request exist.
To make a more specific request, please type help <item>,
where <item> is one of the following
topics:
REPAIR TABLE
REPEAT FUNCTION
REPEAT LOOP
REPLACE
REPLACE FUNCTION
```

EXECUTING SQL STATEMENTS FROM A TEXT FILE

The **mysql** client typically is used interactively, like this:

shell> **mysql** *db_name*

However, it is also possible to put your SQL statements in a file and then tell **mysql** to read its input from that file. To do so, create a text file *text_file* that contains the statements you wish to execute. Then invoke **mysql** as shown here:

shell> **mysql** *db_name* < *text_file*

If you place a USE *db_name* statement as the first statement in the file, it is unnecessary to specify the database name on the command line:

shell> **mysql < text_file**

If you are already running **mysql**, you can execute an SQL script file using the source command or . command:

mysql> **source** *file_name*
mysql> **.** *file_name*

Sometimes you may want your script to display progress information to the user. For this you can insert statements like this:

SELECT <info_to_display> AS ;

The statement shown outputs <info_to_display>.

You can also invoke **mysql** with the **--verbose** option, which causes each statement to be displayed before the result that it produces.

mysql ignores Unicode byte order mark (BOM) characters at the beginning of input files. Previously, it read them and sent them to the server, resulting in a syntax error. Presence of a BOM does not cause **mysql** to change its default character set. To do that, invoke **mysql** with an option such as **--default-character-set=utf8**.

For more information about batch mode, see Section 3.5, "Using mysql in Batch Mode".

MYSQL TIPS

This section describes some techniques that can help you use **mysql** more effectively.

Input–Line Editing

mysql supports input-line editing, which enables you to modify the current input line in place or recall previous input lines. For example, the left-arrow and right-arrow keys move horizontally within the current input line, and the up-arrow and down-arrow keys move up and down through the set of previously entered lines. Backspace deletes the character before the cursor and typing new characters enters them at the cursor position. To enter the line, press Enter.

On Windows, the editing key sequences are the same as supported for command editing in console windows. On Unix, the key sequences depend on the input library used to build **mysql** (for example, the libedit or readline library).

Documentation for the libedit and readline libraries is available online. To change the set of key sequences permitted by a given input library, define key bindings in the library startup file. This is a file in your home directory: .editrc for libedit and .inputrc for readline.

For example, in libedit, Control+W deletes everything before the current cursor position and Control+U deletes the entire line. In readline, Control+W deletes the word before the cursor and Control+U deletes everything before the current cursor position. If **mysql** was built using libedit, a user who prefers the readline behavior for these two keys can put the following lines in the .editrc file (creating the file if necessary):

```
bind ^W ed-delete-prev-word
bind ^U vi-kill-line-prev
```

To see the current set of key bindings, temporarily put a line that says only bind at the end of .editrc. **mysql** will show the bindings when it starts.

Displaying Query Results Vertically

Some query results are much more readable when displayed vertically, instead of in the usual horizontal table format. Queries can be displayed vertically by terminating the query with G instead of a semicolon. For example, longer text values that include newlines often are much easier to read with vertical output:

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1G
*************************** 1. row ***************************
msg_nro: 3068
date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
reply: monty@no.spam.com
mail_to: Thimble Smith <tim@no.spam.com>
sbj: UTF-8
txt: >>>>> Thimble == Thimble Smith writes:
Thimble> Hi. I think this is a good idea. Is anyone familiar
Thimble> with UTF-8 or Unicode? Otherwise, Ill put this on my
Thimble> TODO list and see what happens.
Yes, please do that.
Regards,
```

Monty
file: inbox-jani-1
hash: 190402944
1 row in set (0.09 sec)

Using the –-safe–updates Option

For beginners, a useful startup option is **--safe-updates** (or **--i-am-a-dummy**, which has the same effect). It is helpful for cases when you might have issued a DELETE FROM *tbl_name* statement but forgotten the WHERE clause. Normally, such a statement deletes all rows from the table. With **--safe-updates**, you can delete rows only by specifying the key values that identify them. This helps prevent accidents.

When you use the **--safe-updates** option, **mysql** issues the following statement when it connects to the MySQL server:

SET sql_safe_updates=1, sql_select_limit=1000, sql_max_join_size=1000000;

See Section 5.1.7, "Server System Variables".

The SET statement has the following effects:

- You are not permitted to execute an UPDATE or DELETE statement unless you specify a key constraint in the WHERE clause or provide a LIMIT clause (or both). For example:

  UPDATE *tbl_name* SET *not_key_column=val* WHERE *key_column=val*;
  UPDATE *tbl_name* SET *not_key_column=val* LIMIT 1;
- The server limits all large SELECT results to 1,000 rows unless the statement includes a LIMIT clause.
- The server aborts multiple-table SELECT statements that probably need to examine more than 1,000,000 row combinations.

To specify limits different from 1,000 and 1,000,000, you can override the defaults by using the **--select_limit** and **--max_join_size** options:

shell> **mysql --safe-updates --select_limit=500 --max_join_size=10000**

Disabling mysql Auto–Reconnect

If the **mysql** client loses its connection to the server while sending a statement, it immediately and automatically tries to reconnect once to the server and send the statement again. However, even if **mysql** succeeds in reconnecting, your first connection has ended and all your previous session objects and settings are lost: temporary tables, the autocommit mode, and user-defined and session variables. Also, any current transaction rolls back. This behavior may be dangerous for you, as in the following example where the server was shut down and restarted between the first and second statements without you knowing it:

mysql> **SET @a=1;**
Query OK, 0 rows affected (0.05 sec)
mysql> **INSERT INTO t VALUES(@a);**
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 1
Current database: test
Query OK, 1 row affected (1.30 sec)
mysql> **SELECT * FROM t;**
+------+
| a |
+------+
| NULL |
+------+
1 row in set (0.05 sec)

The @a user variable has been lost with the connection, and after the reconnection it is

undefined. If it is important to have **mysql** terminate with an error if the connection has been lost, you can start the **mysql** client with the **--skip-reconnect** option.

For more information about auto-reconnect and its effect on state information when a reconnection occurs, see Section 23.8.20, “C API Automatic Reconnection Control”.

COPYRIGHT

SEE ALSO

For more information, please refer to the MySQL Reference Manual, which may already be installed locally and which is also available online at http://dev.mysql.com/doc/.

AUTHOR

Oracle Corporation (http://dev.mysql.com/).