

## What To Do If *mysql* Cannot Be Found

If your shell or command interpreter can't find *mysql* when you invoke it, you'll see some sort of error message. It may look like this under Unix:

```
% mysql
mysql: Command not found.
```

Or like this under Windows:

```
C:\> mysql
Bad command or invalid filename
```

One way to tell your shell where to find *mysql* is to type its full pathname each time you run it. The command might look like this under Unix:

```
% /usr/local/mysql/bin/mysql
```

Or like this under Windows:

```
C:\> C:\mysql\bin\mysql
```

Typing long pathnames gets tiresome pretty quickly, though. You can avoid doing so by changing into the directory where *mysql* is installed before you run it. However, I recommend that you *not* do that. If you do, the inevitable result is that you'll end up putting all your data files and query batch files in the same directory as *mysql*, thus unnecessarily cluttering up what should be a location intended only for programs.

A better solution is to make sure that the directory where *mysql* is installed is included in the `PATH` variable that contains the command search path used by your shell. (See the "Setting Environment Variables" sidebar.) Then you can invoke *mysql* from any directory by entering just its name, and your shell will be able to find it. This eliminates a lot of unnecessary pathname typing. An additional benefit is that because you can easily run *mysql* from anywhere, you will have no need to put your data files in the same directory where *mysql* is located. When you're not operating under the burden of running *mysql* from a particular location, you'll be free to organize your files in a way that makes sense to you, not in a way imposed by some artificial necessity. For example, you can create a directory under your home directory for each database you have, and put the files associated with each database in the appropriate directory.

I've pointed out the importance of the search path because I receive many questions from people who aren't aware of the existence of such a thing, and who consequently try to do all their MySQL-related work in the *bin* directory where *mysql* is installed. This seems particularly common among Windows users. Perhaps the reason is that, except for Windows NT, the Windows Help application seems to be silent on the subject of the command interpreter search path or how to set it. (Apparently, Windows Help considers it dangerous for people to know how to do something useful for themselves.)

Another way for Windows users to avoid typing the pathname or changing into the *mysql* directory is to create a shortcut and place it in a more convenient location. That has the advantage of making it easy to start up *mysql* just by opening the shortcut. The disadvantage is that to specify command line options or the startup directory, you have to edit the shortcut's properties, which is pretty cumbersome unless you always invoke *mysql* with the same options. A workaround is to create a shortcut for each set of options you need—for example, one shortcut to connect as an ordinary user for general work and another to connect as the MySQL `root` user for administrative purposes.

---

### <sidebar> Setting Environment Variables

The shell or command interpreter you use to run programs from the command-line prompt includes an environment in which you can store variable values. Some of these variables are used by the shell itself (such as `PATH`, which lists pathnames of directories where the shell looks for commands). Some variables are used by other programs (such as `PERL5LIB`, which tells Perl where to look for library files referenced

by Perl scripts).

The shell you use determines the syntax used to set environment variables, as well as the startup file in which to place the settings. Typical startup files for various shells are shown in the following table. If you've never looked through your shell's startup files, it's a good idea to do so to familiarize yourself with their contents.

Shell	Startup Files
<i>csh, tcsh</i>	<i>.login, .cshrc, .tcshrc</i>
<i>sh, bash, ksh</i>	<i>.profile .bash_profile .bash_login, .bashrc</i>
DOS prompt	<i>C:\AUTOEXEC.BAT</i>

The following examples show how to set the `PATH` variable so that it includes the directory where the `mysql` program is installed. The examples assume there is an existing `PATH` setting in one of your startup files. If you have no `PATH` setting currently, simply add the appropriate line or lines to one of the files.

The `PATH` variable lists the pathnames for one or more directories. If an environment variable's value consists of multiple pathnames, it's conventional under Unix to separate them using the colon character (`:`). Under Windows, pathnames may contain colons, so the separator is the semicolon character (`;`) instead.

To set the `PATH` value for `csh` or `tcsh`, look for a `setenv PATH` command in your startup files, then add the appropriate directory to the line. Suppose your search path is set by a line like this in your `.login` file:

```
setenv PATH /bin:/usr/bin:/usr/local/bin
```

If `mysql` is installed in `/usr/local/mysql/bin`, add that directory to the search path by changing the `setenv` line to look like this:

```
setenv PATH /usr/local/mysql/bin:/bin:/usr/bin:/usr/local/bin
```

For a shell in the Bourne shell family such as `sh`, `bash`, or `ksh`, look in your startup files for lines that set and export the `PATH` variable. The lines will look something like this:

```
PATH=/bin:/usr/bin:/usr/local/bin
export PATH
```

Change them to this:

```
PATH=/usr/local/mysql/bin:/bin:/usr/bin:/usr/local/bin
export PATH
```

Under Windows, check for a line that sets the `PATH` variable in your `AUTOEXEC.BAT` file. It might look like this:

```
PATH=C:\WINDOWS;C:\WINDOWS\COMMAND
```

Change the `PATH` value to include the directory where `mysql` is installed. If this is `C:\mysql\bin`, the resulting `PATH` setting looks like this:

```
PATH=C:\mysql\bin;C:\WINDOWS;C:\WINDOWS\COMMAND
```

Under Windows NT, another way to change the `PATH` value is to use the System control panel (use its Environment panel). In other versions of Windows, you can use the Registry Editor application. Unfortunately, the name of the Registry Editor key that contains the path value seems to vary among versions of Windows. For example, on the Windows machines that I use, the key has one name under Windows ME, and a different name under Windows 98; under Windows 95, I couldn't even find it.

After setting an environment variable, you'll need to cause the modification to take effect. Under Unix, you can log out and log in again. Under Windows, restart the machine.