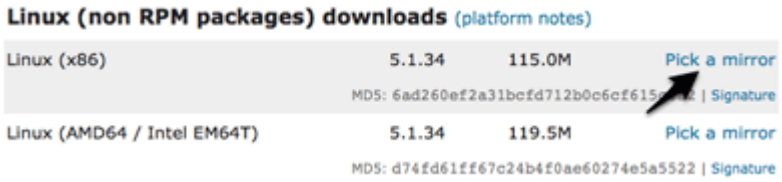
Installing MySQL

Start by downloading MySQL. Simply proceed to the **http://dev.mysql.com/downloads/** and click the Download link for the free MySQL Community Server. This will take you to a page with a long list of download links for the current recommended version of MySQL (as of this writing, it's MySQL 5.1).

Click the link near the top of the list to go to the Linux (non RPM packages). Now you need to choose the package that corresponds to your system architecture. If you're positive you're running a 64-bit version of Linux, go ahead and download the Linux (AMD64/Intel EM64T) package (about 120MB in size). If you're running a 32-bit version of Linux, download the Linux (x86) package (about 115MB)—it'll work even if it turns out you're running a 64-bit version of Linux. It may be a little unclear, but the Pick a mirror link shown in the figure below is the one you need to click to download the file.



Once you've downloaded the file, open a Terminal and log in as the root user:

```
user@machine:~$ sudo su
```

You will, of course, be prompted for your password.

Change directories to /usr/local and unpack the downloaded file:

```
root@machine:/home/user# cd /usr/local
root@machine:/usr/local# tar xfz ~user/Desktop/mysql-version-linux-platform.tar.gz
```

The second command assumes you left the downloaded file on your desktop, which is the Desktop directory in your home directory. You'll need to replace user with your username, version with the MySQL version you downloaded, and platform with the architecture and compiler version of the release you downloaded; this is so that the command exactly matches the path and filename of the file you downloaded. On my computer, for example, the exact command looks like this:

```
root@mythril:/usr/local# tar xfz ~kyank/Desktop/mysql-5.1.34-linux-x86_64-glibc23.tar.gz
```

After a minute or two, you'll be returned to the command prompt. A quick ls will confirm that you now have a directory named mysql-version-linux-platform. This is what it looks like on my computer:

```
root@mythril:/usr/local# ls
bin  games    lib  mysql-5.1.34-linux-x86_64-glibc23  share
etc  include  man  sbin                               src
```

Next, create a symbolic link to the new directory with the name mysql to make accessing the directory easier. Then enter the directory:

```
root@machine:/usr/local# ln -s mysql-version-linux-platform mysql
root@machine:/usr/local# cd mysql
```

While you can run the server as the root user, or even as yourself (if, for example, you were to install the server in your home directory), you should normally set up on the system a special user whose sole purpose is to run the MySQL server. This will remove any possibility of an attacker using the MySQL server as a way to break into the rest of your system. To create a special MySQL user, type the following commands (still logged in as root):

```
root@machine:/usr/local/mysql# groupadd mysql
root@machine:/usr/local/mysql# useradd -g mysql mysql
```

Now give ownership of your MySQL directory to this new user:

```
root@machine:/usr/local/mysql# chown -R mysql .
root@machine:/usr/local/mysql# chgrp -R mysql .
```

MySQL is now installed, but before it can do anything useful, its database files need to be installed, too. Still in the new mysql directory, type the following command:

```
root@machine:/usr/local/mysql# scripts/mysql_install_db --user=mysql
```

Now everything's prepared for you to launch the MySQL server for the first time. From the same directory, type the following command:

```
root@machine:/usr/local/mysql# bin/mysqld_safe --user=mysql &
```

If you see the message mysql daemon ended, then the MySQL server was prevented from starting. The error message should have been written to a file called hostname.err (where hostname is your machine's host name) in MySQL's data directory. You'll usually find that this happens because another MySQL server is already running on your computer.

If the MySQL server was launched without complaint, the server will run (just like your web or FTP server) until your computer is shut down. To test that the server is running properly, type the following command:

```
root@machine:/usr/local/mysql# bin/mysqladmin -u root status
```

A little blurb with some statistics about the MySQL server should be displayed. If you receive an error message, check the hostname.err file to see if the fault lies with the MySQL server upon starting up. If you retrace your steps to make sure you followed the process described above.

If you want your MySQL server to run automatically whenever the system is running, you'll have to set it up to do so. In the support-files subdirectory of the mysql directory, you'll find a script called mysql.server that can be added to your system startup routines to do this. For most versions of Linux, you can do this by creating a link to the mysql.server script in the /etc/init.d directory, then create two links to that: /etc/rc2.d/S99mysql and /etc/rc0.d/K01mysql. Here are the commands to type:

```
root@machine:/usr/local/mysql# cd /etc
root@machine:/etc# ln -s /usr/local/mysql/support-files/mysql.server init.d/
root@machine:/etc# ln -s /etc/init.d/mysql.server rc2.d/S99mysql
root@machine:/etc# ln -s /etc/init.d/mysql.server rc0.d/K01mysql
```

That's it! To test that this works, reboot your system, and request the status of the server with mysqladmin as you did above.

One final thing you might like to do for the sake of convenience is to place the MySQL client programs—which you'll use to administer your MySQL server later on—in the system path. To this end, you can place symbolic links to mysql, mysqladmin, and mysqldump in your /usr/local/bin directory:

```
root@machine:/etc# cd /usr/local/bin
root@machine:/usr/local/bin# ln -s /usr/local/mysql/bin/mysql .
root@machine:/usr/local/bin# ln -s /usr/local/mysql/bin/mysqladmin .
root@machine:/usr/local/bin# ln -s /usr/local/mysql/bin/mysqldump .
```

Once you've done this, you can log out of the root account. From this point on, you can administer MySQL from any directory on your system:

```
root@machine:/usr/local/bin# exit
user@machine:~$ mysqladmin -u root status
```

Installing PHP

As mentioned above, PHP is more a web server plugin module than a program. There are actually three ways to install the PHP plugin for Apache:

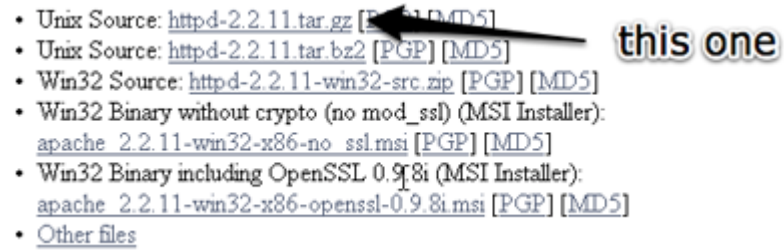
- as a CGI program that Apache runs every time it needs to process a PHP-enhanced web page
- as an Apache module compiled right into the Apache program
- as an Apache module loaded by Apache each time it starts up

The first option is the easiest to install and set up, but it requires Apache to launch PHP as a program on your computer every time a PHP page is requested. This activity can really slow down the response time of your web server, especially if more than one request needs to be processed at a time.

The second and third options are almost identical in terms of performance, but the third option is the most flexible, since you can add and remove Apache modules without having to recompile it each time. For this reason, we'll use the third option.

Assuming you don't already have Apache running on your computer, surf on over to the **http://httpd.apache.org/** and look for the version of Apache described as "the best available version" (as of this writing it's version 2.2.11, as shown below).

Once you get to the Download page, scroll down to find the links to the various versions available. The one you want is Unix Source, shown in the figure below. Both the .tar.gz or the .tar.bz2 are the same; just grab whichever archive format you're used to extracting.



What you've just downloaded is actually the source code for the Apache server. The first step, then, is to compile it into an executable binary installation. Pop open a Terminal, navigate to the directory where the downloaded file is located, then extract it, and navigate into the resulting directory:

```
user@machine:~$ cd Desktop
user@machine:~/Desktop$ tar xfz httpd-version.tar.gz
user@machine:~/Desktop$ cd httpd-version
```

The first step in compiling Apache is to configure it to your requirements. Most of the defaults will be fine for your purposes, but you'll need to enable dynamic loading of Apache modules (like PHP), which is off by default. Additionally, you should probably enable the URL rewriting feature, upon which many PHP applications rely (although it's unnecessary for the examples in this book). To make these configuration changes, type this command:

```
user@machine:~/Desktop/httpd-version$ ./configure --enable-so --enable-rewrite
```

A long stream of status messages will parade up your screen. If the process stops with an error message, your system may be missing some critical piece of software that's required to compile Apache. Some Linux distributions lack the essential development libraries or even a C compiler installed by default. Installing these should enable you to return and run this command successfully. Current versions of Ubuntu, however, should come with everything that's needed.

After several minutes, the stream of messages should come to an end:

```
...
config.status: creating build/rules.mk
config.status: creating build/pkg/pkginfo
config.status: creating build/config_vars.sh
config.status: creating include/ap_config_auto.h
config.status: executing default commands
user@machine:~/Desktop/httpd-version$
```

You're now ready to compile Apache. The one-word command make is all it takes:

```
user@machine:~/Desktop/httpd-version$ make
```

Again, this process will take several minutes to complete, and should end with the following message:

```
...
make[1]: Leaving directory `/home/user/Desktop/httpd-version'
user@machine:~/Desktop/httpd-version$
```

To install your newly-compiled copy of Apache, type sudo make install (the sudo is required, since you need root access to write to the installation directory).

```
user@machine:~/Desktop/httpd-version$ sudo make install
```

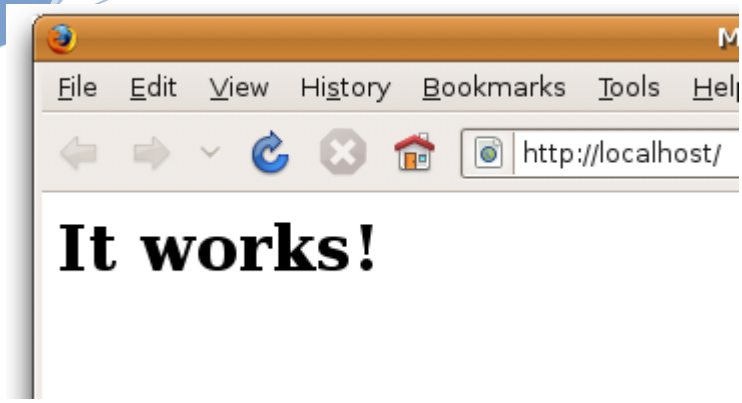
Enter your password when prompted.

As soon as this command has finished copying files, your installation of Apache is complete. Navigate to the installation directory and launch Apache using the apachectl script:

```
user@machine:~/Desktop/httpd-version$ cd /usr/local/apache2
user@machine:/usr/local/apache2$ sudo bin/apachectl -k start
```

You'll likely see a warning message from Apache complaining that it was unable to determine the server's fully qualified domain name. That's because most personal computers are without one. Don't sweat it.

Fire up your browser and type http://localhost into the address bar. If Apache is up and running, you should see a welcome message like the one below.



As with your MySQL server, you'll probably want to configure Apache to start automatically when your system boots. The procedure to do this is similar; just copy and link the apachectl script from your Apache installation:

```
user@machine:/usr/local/apache2$ sudo su
root@machine:/usr/local/apache2# cd /etc
root@machine:/etc# ln -s /usr/local/apache2/bin/apachectl init.d/
root@machine:/etc# ln -s /etc/init.d/apachectl rc2.d/S99httpd
root@machine:/etc# ln -s /etc/init.d/apachectl rc0.d/K01httpd
```

To test that this works, restart your computer and then hit the http://localhost page in your browser again.

With a shiny new Apache installation up and running, you're now ready to add PHP support to it. To start, download the PHP Complete Source Code package from the PHP Downloads page. Again, the .tar.gz and .tar.bz2 versions are identical; just download whichever you're used to extracting.

The file you downloaded should be called php-version.tar.gz (or .bz2). Pop open a new Terminal window, navigate to the directory containing the downloaded file, extract it, and move into the resulting directory:

```
user@machine:~$ cd Desktop
user@machine:~/Desktop$ tar xfz php-version.tar.gz
user@machine:~/Desktop$ cd php-version
```

To install PHP as an Apache module, you'll need to use the Apache apxs program. This will have been installed along with the Apache server if you followed the instructions above to compile it yourself; but if you're using the copy that was installed with your distribution of Linux, you may need to install the Apache development package to access Apache apxs. You should be able to install this package by using the package manager included with your Linux distribution. For example, on Debian Linux, you can use apt-get to install it as follows:

```
user@machine:~$ sudo apt-get install apache-dev
```

Now, to install PHP, you must be logged in as root:

```
user@machine:~/Desktop/php-version$ sudo su
[sudo] password for user: (type your password)
root@machine:/home/user/Desktop/php-version#
```

The first step is to configure the PHP installation program by telling it which options you want to enable, and where it should find the programs it needs to know about (such as Apache apxs and MySQL). The command should look like this (all on one line):

```
root@machine:/home/user/Desktop/php-version# ./configure --prefix=/usr/local/php --with-apxs2=/usr/local/apache2/bin/apxs --
with-mysqli=/usr/local/mysql/bin/mysql_config
```

The --prefix option tells the installer where you want PHP to be installed (/usr/local/php is a good choice).

The --with-apxs2 option tells the installer where to find the Apache apxs program mentioned above. When installed using your Linux distribution's package manager, the program is usually found at /usr/sbin/apxs. If you compiled and installed Apache yourself as described above, however, it will be in the Apache binary directory, at /usr/local/apache2/bin/apxs.

The --with-mysqli option tells the installer where to find your MySQL installation. More specifically, it must point to the mysql_config program in your MySQL installation's bin directory (/usr/local/mysql/bin/mysql_config).

Again, a parade of status messages will appear on your screen. When it stops, check for any error messages and install any files it identifies as missing. On a default Ubuntu 8.10 installation, for example, you're likely to see an error complaining about an incomplete libxml2 installation. To correct this particular error, open Synaptic Package Manager, then locate and install the libxml2-dev package (libxml2 should already be installed). Once it's installed, try the configure command again.

After you watch several screens of tests scroll by, you'll be returned to the command prompt with the comforting message "Thank you for using PHP." The following two commands will compile and then install PHP:

```
root@machine:/home/user/Desktop/php-version# make
root@machine:/home/user/Desktop/php-version# make install
```

Take a coffee break: this will take some time.

Upon completion of the make install command, PHP will be installed in /usr/local/php (unless you specified a different directory with the --prefix option of the configure script above). Now you just need to configure it!

The PHP configuration file is called php.ini. PHP comes with two sample php.ini files called php.ini-dist and php.ini-recommended. Copy these files from your installation work directory to the /usr/local/php/lib directory, then make a copy of the php.ini-dist file and call it php.ini:

```
root@machine:/home/user/Desktop/php-version# cp php.ini* /usr/local/ php/lib/
root@machine:/home/user/Desktop/php-version# cd /usr/local/php/lib
root@machine:/usr/local/php/lib# cp php.ini-dist php.ini
```

You may now delete the directory from which you compiled PHP—it's no longer needed.

We'll worry about fine-tuning php.ini shortly. For now, we need to tweak Apache's configuration to make it more PHP-friendly. Locate your Apache httpd.conf configuration file. This file can usually be found in the conf subdirectory of your Apache installation (/usr/local/apache2/conf/httpd.conf).

To edit this file you must be logged in as root, so launch your text editor from the Terminal window where you're still logged in as root:

```
root@machine:/usr/local/php/lib# cd /usr/local/apache2/conf
root@machine:/usr/local/apache2/conf# gedit httpd.conf
```

In this file, look for the line that begins with DirectoryIndex. This line tells Apache which filenames to use when it looks for the default page for a given directory. You'll see the usual index.html, but you need to add index.php to the list:

```
<IfModule dir_module>
 DirectoryIndex index.html index.php
</IfModule>
```

Finally, go right to the bottom of the file and add these lines to tell Apache that files with names ending in .php should be treated as PHP scripts:

```
<FilesMatch \.php$>
 SetHandler application/x-httpd-php
</FilesMatch>
```

That should do it! Save your changes and restart your Apache server with this command:

```
root@machine:/usr/local/apache2/conf# /usr/local/apache2/bin/apachectl -k restart
```

If it all goes according to plan, Apache should start up without any error messages. If you run into any trouble, the helpful individuals in the SitePoint Forums (myself included) will be happy to help.

Post-Installation Set-up Tasks

Regardless of which operating system you're running, or how you set up your web server—once PHP is installed and the MySQL server is functioning, the very first action you need to perform is assign a root password for MySQL.

MySQL only allows authorized users to view and manipulate the information stored in its databases, so you'll need to tell MySQL who's authorized and who's unauthorized. When MySQL is first installed, it's configured with a user named root that has access to do most tasks without even entering a password. Your first task should be to assign a password to the root user so that unauthorized users are prohibited from tampering with your databases.

Why Bother?

It's important to realize that MySQL, just like a web server, can be accessed from any computer on the same network. If you're working on a computer connected to the Internet, then, depending on the security measures you've taken, anyone in the world could connect to your MySQL server. The need to pick a difficult-to-guess password should be immediately obvious!

To set a root password for MySQL, you can use the mysqladmin program that comes with MySQL. If you followed the instructions to install MySQL separately (as explained earlier in this chapter), the mysqladmin program should be on your system path. This means you can pop open a Terminal window (or in Windows, a Command Prompt) and type the name of the program without having to remember where it's installed on your computer.

```
mysqladmin -u root status
```

When you hit Enter you should see a line or two of basic statistics about your MySQL server, like this:

```
Uptime: 102261  Threads: 1  Questions: 1  Slow queries: 0  Opens: 15
Flush tables: 1  Open tables: 0  Queries per second avg: 0.0
```

If you're seeing a different message entirely, it's probably one of two options. First, you might see an error message telling you that the mysqladmin program was unable to connect to your MySQL server:

```
mysqladmin: connect to server at 'localhost' failed
error: 'Can't connect to MySQL server on 'localhost' (10061)'
Check that mysqld is running on localhost and that the port is 3306.
You can check this by doing 'telnet localhost 3306'
```

This message normally means that your MySQL server simply isn't running. If you have it set up to run automatically when your system boots, double-check that the setup is working. If you normally launch your MySQL server manually, go ahead and do that before trying the command again.

Second, if you're using MAMP on the Mac, you'll probably see this error message instead:

```
mysqladmin: connect to server at 'localhost' failed
error: 'Access denied for user 'root'@'localhost' (using password: NO)'
```

This error message means that the root user on your MySQL server already has a password set. It turns out that, with your security in mind, MAMP comes with a root password already set on its built-in MySQL server. That password, however, is root—so you're probably still going to want to change it using the instructions below.

One way or the other, you should now be able to run the mysqladmin program. Now you can use it to set the root password for your MySQL server:

```
mysqladmin -u root -p password "newpassword"
```

Replace newpassword with whatever password you'd like to use for your MySQL server. Make sure it's one you can remember, because if you forget your MySQL root password, you might need to erase your entire MySQL installation and start over from scratch! As we'll see in Chapter 10, MySQL Administration, it's usually possible to recover from such a mishap, but it's definitely a pain in the neck.

When you hit Enter, you'll be prompted to enter the current password for the root MySQL user. Just hit Enter again, since the root user has no password at this point, unless you've used MAMP to set up MySQL on your Mac; in this case you should type root, the default root MySQL password on MAMP.

Let me break this command down for you, so you can understand what each part means:

```
mysqladmin
```

This, of course, is the name of the program you wish to run.

```
-u root
```

This specifies the MySQL user account you wish to use to connect to your MySQL server. On a brand new server, there is only one user account: root.

```
-p
```

This tells the program to prompt you for the current password of the user account. On a brand new MySQL server, the root account has no password, so you can just hit Enter when prompted. It's a good idea, however, to make a habit of including this option, since most of the time you will need to provide a password to connect to your MySQL server.

```
password "newpassword"
```

This instructs the mysqladmin program to change the password of the user account to newpassword. In this example, whatever password you specify will become the new password for the root MySQL user.

Now, to try out your new password, request once again that the MySQL server tell you its current status at the system command prompt, but this time include the -p option:

```
mysqladmin -u root -p status
```

Enter your new password when prompted. As before, you should see a line or two of statistics about your MySQL server.

Since the root account is now password-protected, attempting to run this command without the -p switch will give you an "Access Denied" error.

You're done! With everything set up and running, you're ready to write your first PHP script. Before we do that, however, you might want to write a short email to your web host.

What to Ask Your Web Host

While you tinker with PHP and MySQL on your own computer, it might be good to start collecting the information you'll need when it comes time to deploy your first database driven web site to the public. Here's a rundown of the details you should be asking your web host for.

First, you'll need to know how to transfer files to your web host. You'll upload PHP scripts to your host the same way you normally send the HTML files, CSS files, and images that make up a static web site, so if you already know how to do that, it's unnecessary to bother your host. If you're just starting with a new host, however, you'll need to be aware of what file transfer protocol it supports (FTP or SFTP), as well as knowing what username and password to use when connecting with your (S)FTP program. You also have to know what directory to put files into so they're accessible to web browsers.

In addition to these, you'll also need to find out a few details about the MySQL server your host has set up for you. It's important to know the host name to use to connect to it (possibly localhost), and your MySQL username and password, which may or may not be the same as your (S)FTP credentials. Your web host will probably also have provided an empty database for you to use, which prevents you from interfering with other users' databases who may share the same MySQL server with you. If they have provided this, you should establish the name of that database.

Your First PHP Script

It would be unfair of me to help you install everything—but stop short of giving you a taste of what a PHP script looks like until http://www.sitepoint.com/article/mysql-3-getting-started-php/ So here's a little morsel to whet your appetite.

Open your favorite text or HTML editor and create a new file called today.php. Type this into the file:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>Today&rsquo;s Date</title>
  <meta http-equiv="content-type"
     content="text/html; charset=utf-8"/>
 </head>
 <body>
  <p>Today&rsquo;s date (according to this web server) is
   <?php

    echo date('l, F dS Y.');

    ?>
  </p>
 </body>
</html>
```

Editing PHP Scripts in Windows with Notepad

Windows users should note that, to save a file with a .php extension in Notepad, you'll need to either select All Files as the file type, or surround the filename with quotes in the Save As dialog box; otherwise, Notepad will unhelpfully save the file as today.php.txt, which will fail to work.