Compiling PHP from source

- Assumptions
- Apache web server
- Obtaining additional libraries
- Build environment
- Obtaining and building PHP
- Configuring Apache and PHP
- Testing
- Troubleshooting
- Updating PHP
- See also

This page shows you how to build PHP from source on Ubuntu or other Debian based distributions. It could no doubt be used as a basis for compiling PHP on other Unix or Linux systems.

It is possible that you may need to do this because the latest versions of Moodle require reasonably new versions of PHP which may not be supported by the packages in your version of PHP - especially "long term support" versions.

This discussion was based on Ubuntu 10.04 which (at the time of writing) is the latest Ubuntu LTS. It does not carry a new enough PHP version for Moodle 2.1.

- 1 Assumptions
- 2 Apache web server
- 3 Obtaining additional libraries
- 4 Build environment
- 5 Obtaining and building PHP
- 6 Configuring Apache and PHP
- 7 Testing
- 8 Troubleshooting

- 9 Updating PHP
- 10 See also

Assumptions

It is assumed that you will have a basic install of Ubuntu (10.04 in this case) without (specifically) PHP installed and possibly no Apache web server either. If you already have PHP installed (as a package) you will need to remove it first:

```
sudo apt-get remove php5
```

Apache web server

You can either install Apache from the Ubuntu packages (recommended) or compile it from source. Compiling from source is simple but you end up (using default settings) with a directory structure that is completely different from the Ubuntu packaged version. This is not covered further here.

To install the package version, it's just

```
sudo apt-get install apache2 apache2-dev
```

The -dev package is required in order to build PHP.

Obtaining additional libraries

Moodle requires a significant number of optional PHP modules. Many of these require development libraries to be available on the system before PHP is compiled. This aspect is what makes building PHP from source tricky. If you would like a challenge, all of these can be downloaded as source packages and built from scratch but it is much easier to use the packaged versions. They are installed as follows...

```
sudo apt-get install \
    libxml2-dev \
    libcurl4-openssl-dev \
    libjpeg-dev \
    libpng-dev \
    libxpm-dev \
    libmysqlclient-dev \
```

```
    libpq-dev \
    libicu-dev \
    libfreetype6-dev \
    libldap2-dev \
    libxslt-dev \
    libssl-dev \
    libldb-dev
```

You may also need to create the following symbolic links as root on Linux:

```
ln -s /usr/lib/x86_64-linux-gnu/libldap.so /usr/lib/libldap.so
ln -s /usr/lib/x86_64-linux-gnu/liblber.so /usr/lib/liblber.so
```

Note that the above includes the clients for both the MySQL and PostgreSQL databases. You may not want to install both. You may need to include other libraries if you add further options to your PHP build. A Google search involving your Ubuntu version, the function and the word 'library' or 'development' will often turn up the correct package name.

Build environment

Ubuntu does not have all the compilers, linkers and libraries you need in a standard installation. If you have not compiled anything from source before this can be fixed by a single command...

```
sudo apt-get install build-essential
```

Obtaining and building PHP

The latest version can be downloaded from www.php.net. At the time of writing, this was 7.2.10 but new versions come out quite regularly. This should download as a .tar.gz file (e.g. php-7.2.10.tar.gz). Place this in a suitable location in (probably) your home folder and unpack the file.

```
tar -zxvf php-7.x.y.tar.gz
cd php-7.x.y
```

The next step is to run the 'configure' program. This digs around your system and creates specific make files based on your particular configuration. It also specified all the optional modules that will be compiled in. The minimum for Moodle 2 (and 1.9) is as follows...

```
./configure \
  --prefix=/usr/local/php \
  --with-apxs2=/usr/local/apache/bin/apxs \
  --enable-mbstring \
  --with-curl \
```

```
  --with-openssl \
  --with-xmlrpc \
  --enable-soap \
  --enable-zip \
  --with-gd \
  --with-jpeg-dir \
  --with-png-dir \
  --with-mysqli \
  --with-pgsql \
  --enable-embedded-mysqli \
  --with-freetype-dir \
  --with-ldap \
  --enable-intl \
  --with-xsl \
  --with-zlib
```

If you have special requirements you **may** need others. To get the full list of possibilities you can do...

```
./configure --help
```

This should complete without errors and finishes with an obvious copyright notice in a box. If you do get errors, it is most likely to be due to missing libraries. Make sure you have added all the libraries described above (with apt-get). Failing that, Google is your friend.

Once that bit is done, it's time to do the actual compiling and linking. Simply do...

```
make
```

This (depending on how fast your machine is) will take some time and will end up with the phrase `Build complete`.

It's then just a matter of installing the files...

```
sudo make install
```

Note: PHP_BINDIR is a value set on compile time. The path is set to the prefix used in configure (Linux equivalent: ./configure --prefix).

Configuring Apache and PHP

The first step is to copy the PHP configuration file (php.ini) from the source distribution to where it will be read on Apache startup. There are two supplied versions, one for production systems and one for development systems. The latter display many more errors and warnings and should (for security reasons) only be used in a development environment, as its name suggests. To copy your chosen file, do the following (from the source directory)..

```
sudo cp php.ini-production /usr/local/lib/php.ini
sudo ln -s /usr/local/lib/php.ini /etc
```

The second line is optional and makes a link to the ini file in /etc (you can find it at /etc/php.ini). At this point you might want to edit the ini file to customise it for your needs. A very likely change is the file upload limits and the php memory_limit setting.

Apache's configuration file will now require some changes in order to correctly handle PHP files. Ubuntu's APache configuration is a little strange and could be a whole discussion by itself. Essentially, you need to add the following lines:

```
LoadModule php5_module  modules/libphp5.so
AddType application/x-http-php  .php
DirectoryIndex  index.php  index.html
```

The first two can be stuck at the end of /etc/apache2/apache2.conf and the DirectoryIndex by checking mods-available/dir.conf and ensuring that index.php is in the list. Strictly speaking, you would create a file in *mods-available* and enable it with the *a2enmod* command but this is left as an exercise!

Once this has been completed, you just need to restart Apache...

```
sudo /etc/init.d/apache2 restart
```

Testing

The easiest way to test, is to create a small test file in the root of your web served directory (/var/www), calling it something like *test.php*...

```
  <?php phpinfo();
```

Then access the page from a web browser using http://hostname.of.your.server/test.php. You should see a lengthy list of the PHP setup (which you might want to check). If you just see the php code, the three settings in the Apache config have not worked for some reason and will require checked. If you get some other error, read on...

Troubleshooting

The best advice I can give is to look at the logs. The error log for Apache is in /var/log/apache/error.log with the latest errors at the end. Always look here first. Most problems are caused by incorrect file or directory permissions or incorrect settings in the Apache configuration.

Updating PHP

There are really two possibilities. Firstly, you may want to rebuild php with extra configuration settings, or you may want to build a newer version of PHP. You can follow the installation instructions exactly as above, right up to and including *sudo make install*. After that, simply restart the Apache web server and you are done.

For a new version, just use the same ./configure command as you did with the older version (not absolutely guaranteed to work, but very likely). If you can't remember the full command look in (in the old version) a file called config.log in the build directory or run the 'phpinfo' test above and you will see the config command used right at the top.

See also

- Installing Moodle 2.7 on CentOS 6.5 (with PHP 5.3.3)? Forum thread

---

Source: docs.moodle.org/503/en/Compiling PHP from source — MoodleDocs (GPL) · Imported 2026-11-10.

□□□ #3
□ cocobook-sync □□□ 2026-11-10 12:43:26 UTC
□ cocobook-sync □□□ 2026-11-10 23:18:15 UTC