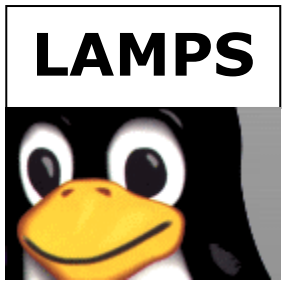
Anleitung zur Erstellung eines LAMPS-Systems / PhPeppershop

Andreas Jack Wegner

LAMPS Tutorial

Linux, Apache, MySQL, PHP und SSL



Februar 2003, Andreas Jack Wegner

LAMPS - Tutorial : Mit Linux, Apache, MySQL, PHP und SSL und dem PhPepperShop

Inhalt

Einleitung

Dieses kleine Tutorial soll helfen, einen LAMPS Server aufzusetzen und zu konfigurieren. Aufsetzen heist für dieses Tutorial, selber alle Software zu compilieren und installieren (Apache, PHP, MySQL usw). Das ist für die ersten Schritte mit PHP natrülich nicht unbedingt nötig. Bei den meisten Distributionen ist jetzt schon neben einem Apache ein mod_ssl und ein mod_php dabei. Damit kann man Anfangen.
Möchte man dann doch ein eigenes System, sollte diese Anleitung helfen. Voraussetzung ist, dass Linux schon installiert ist.

Was wird gebraucht?

Zuerst habe ich die Dateien auf meinem Windows Rechner heruntergeladen und auf eine CD gebrannt:

Linux

Ist ja, wie oben definiert, schon auf Deinem Rechner installiert. Diese Anleitung wurde mit Verwendung von SUSE 7.3 geschrieben. Neuere Versionen sollten aber keine merklichen Unterschiede aufweisen.
Praktisch ist, wenn Du schon jetzt darauf achtest die Pakete `flex`, `bison`, `xdevel` und `libpng` installiert zu haben - daß spart später eine Menge Ärger (PHP kann ohne `flex` z.B. nicht kompiliert werden).

Apache

Du solltest Dir die aktuellen Quellen runterladen, im Augenblick ist das Apache 1.3.27 (PHP für Apache2 noch im experimentellen Stadium).

Homepage: `http://httpd.apache.org`
Datei: `http://apache.serveftp.org/apache-site/dist/httpd/apache_1.3.27.tar.gz`

Mysql

Mysql als Datenbankserver. Im Augenblick ist die Version 3.23.55 die empfohlene Version.

Homepage: http://www.mysql.com
Datei: http://www.mysql.com/Downloads/MySQL-3.23/mysql-3.23.55.tar.gz
Mirror: http://ftp.gwdg.de/pub/misc/mysql/Downloads/MySQL-3.23/mysql-3.23.55.tar.gz

PHP

(Version 4.2.3) → PHP 4.3.0 siehe Anmerkung bei GD-Library
Homepage: http://www.php.net
Mirror: http://www.php3.de
Datei: http://www.php3.de/distributions/php-4.2.3.tar.gz
Dokumentation: http://php3.de/download-docs.php

Erweiterungen

Mit den oben genannten Grundkomponenten könnte man jetzt schon aufhören, aber es gibt noch ein paar nette Zusatzmodule, die man unbedingt mitinstallieren sollte (und für den PhPepperShop muss).

SSL-Zubehör

Damit der Webserver auch sicher verschlüsselte Kommunikation mit den Clients betreiben kann, sollte er SSL unterstützen. Dies bringt man ihm am besten mit dem Apache Modul `mod_ssl` bei. Das `mod_ssl` Modul setzt wiederum `openssl` voraus. `Openssl` liefert, soweit ich das verstehe, die Verschlüsselungsfunktionalität, `mod_ssl` integriert die Funktionalität in den Apache Webserver.

OpenSSL - in der aktuellen Version 0.9.6d

Homepage: http://www.openssl.org
Datei: http://www.openssl.org/source/openssl-0.9.6d.tar.gz

mod_ssl - man muß darauf achten, daß die Datei zur jeweiligen Apache-Version passt.

Homepage: http://www.modssl.org
Datei: http://www.modssl.org/source/mod_ssl-2.8.10-1.3.27.tar.gz

GD-Library

Möglicherweise möchtest Du mit PHP Buttons, Statistiken oder andere bunte Bildchen dynamisch erzeugen können. Hierzu brauchst Du die *GD-Library*, damit Du die "Image functions" auch benutzen kannst. Hierbei gibt es allerdings einen echten Sprung zwischen den Versionen: während Versionen bis einschließlich 1.62 nur GIFs erzeugen, können Versionen danach "nur" PNG und ab 1.8 auch JPEG. Also muß man wählen. Ich wähle 1.8.4, weil ich mit JPEG alles machen kann, was mich mit GIF auch machen wollte, nur daß ich damit endlich auch große und bunte Bilder bearbeiten / erzeugen kann. Auch der PhPepperShop ist für die GD-Library bis und mit 1.8.4 augelegt.
Für die GD-Library wird als "integraler Bestandteil" die *zlib* gebraucht. Darüber hinaus braucht man auch noch die IJG JPEG software (*libjpeg*), damit GD auch was von JPEG weiß. Außerdem ist es gerade für die Buttons noch sehr sinnvoll, und für die GD Library notwendig, sich die *freetype2* Library zu installieren. Damit kann man Textelemente mit TTF-Fonts in die Bilder einbauen - damit der Designer nicht immer so laut schreit. ;-)
Hier noch eine *Anmerkung zur neuen GD v.2.0x*. Diese bietet viele Neuerungen, aber die Funktions APIs haben sich geringfügig geändert. Wenn man z.B. *PHP 4.3.0* installiert (welches standardmässig die GD v.2.03 mitbringt, muss man im PhPepperShop Forum nach GD suchen gehen und die Aufrufe anpassen – sonst werden die Bilder nicht schön erstellt).

GD-Library Download

Homepage: http://www.boutell.com/gd/
Datei - Version 1.8.4: http://www.boutell.com/gd/http/gd-1.8.4.tar.gz

Zlib - Version 1.1.4

Homepage (eher Heimverzeichnis): http://www.gzip.org/zlib/
Datei: http://www.gzip.org/zlib/zlib-1.1.4.tar.gz

IJG JPEG Software - Version 6b

Homepage: www.ijg.org
Datei: ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz

Freetype - in der Version 2.1.2

Homepage: http://www.freetype.org
Datei: ftp://ftp.freetype.org/freetype/freetype2/ftdocs-2.1.2.tar.gz

Libpng

Homepage: http://www.libpng.org/pub/png/libpng.html
Datei: http://www.libpng.org/pub/png/libpng.html

PDFlib

Homepage: http://www.pdflib.com
Datei: http://www.pdflib.com/pdflib/download/index.html (Die Datei ganz unten)

PhPepperShop – in der Version 1.2 (aktuelle GPL-Version, Februar 2003)

Homepage: http://www.phpeppershop.com
Datei: http://www.phpeppershop.com (Unter Downloads)

Installation

Jetzt sind wir mit dem Herunterladen fertig, und können übergehen zum nächsten Schritt, derVorbereitungen.

Nach der Installation von Linux, müssen noch vier Pakete installiert werden. Diese können via SuSE Installationstool YaST abgerufen werden:

Folgende Pakete installieren (Pakete suchen, markieren, auf OK klicken):
"ncurces-devel"
"Bison"
"Flex"
"gdbm-devel"

Wir werden jetzt nach und nach die einzelnen Elemente installieren. Die Reihenfolge ergibt sich daraus, welche Software welche Libraries voraussetzt. Bei den Installationspfaden verfahre ich (bis auf einige Ausnahmen), nach dem Schema `/usr/local/'paketname'/'versionsnummer'`. Dann wird von `/usr/local/'paketname'/current` ein Link auf `/usr/local/'paketname'/'versionsnummer'` gesetzt, und im Weiteren verwendet. Schreibe ich aber jedes mal auch noch extra dabei. Einen Hinweis noch: Die ganzen Schritte hier habe ich als `'root'` ausgeführt (um `root` zu werden einfach `su` tippen und das Passwort eingeben).

Zuerst den Pfad erstellen wo die Dateien kopiert werden sollen:

```
cd /usr/local/src
mkdir lamp
```

als nächstes das CD-ROM mounten:

```
cd /
mount /cdrom
```

Jetzt müßen die Dateien von der CD in das oben genannte Verzeichnis kopiert werden. Zuerst in das Verzeichnis vom CD-ROM wechseln und die Datei kopieren:

```
cp apache_1.3.27.tar.gz /usr/local/src/lamp
```

Das selbe mit den anderen Dateien.
Jetzt müssen die noch entpackt werden, siehe unten:

Zunächst mal die gesamte Software auspacken:

```
1.   cd /usr/local/src/lamp
2.   tar -xzf apache_1.3.27.tar.gz
3.   tar -xzf freetype-2.1.2.tar.gz
4.   tar -xzf gd-1.8.4.tar.gz
5.   tar -xzf mod_ssl-2.8.10-1.3.27.tar.gz
6.   tar -xzf mysql-3.23.55.tar.gz
7.   tar -xzf openssl-0.9.6d.tar.gz
8.   tar -xzf php-4.2.2.tar.gz
9.   tar -xzf zlib-1.1.4.tar.gz
10.  tar -xzf jpegsrc.v6b.tar.gz
11.  tar -xzf libpng-1.2.4.tar.gz
12.  tar -xzf pdflib-4.0.3.tar.gz
```


MySQL-Datenbank installieren

Wechseln Sie in das Verzeichnis mit den MySQL-Source-Daten

```
cd /usr/local/src/lamp/mysql-3.23.55/
```

Legen Sie nun am besten die Ablagepfade für MySQL-Datenbanken an. In unserem Beispiel sind das:

```
mkdir /usr/local/mysql
mkdir /usr/local/mysql/data
```

und führen Sie den `configure`-Befehl mit dem Installationspfad `/usr/local/mysql/3.23.55` und dem Ablagepfad `/var/mysql/data` aus

```
./configure --prefix=/usr/local/mysql/3.23.55 --localstatedir=/usr/local/mysql/data
```

Dieser Vorgang kann einige Zeit in Anspruch nehmen. Sollte es zu Problemen kommen, überprüfen Sie bitte nochmals Ihre Eingabe auf Korrektheit. Wenn alles ohne Probleme konfiguriert wurde, können Sie die Distribution kompilieren:

```
make
```

auch das kann einige Zeit dauern. Haben Sie etwas Geduld. Wenn auch das Kompilieren ohne Fehlermeldung vonstatten gegangen ist, können Sie MySQL installieren:

```
make install
```

Als nächstes erstellen Sie einen (Soft-)Link:

```
ln -s /usr/local/mysql/3.23.55 /usr/local/mysql/current
```

Wechseln Sie in den MySQL Ordner und installieren Sie die initiale Test Datenbank:

```
cd /usr/local/mysql/current/bin
./mysql_install_db
```

nun müssen Sie noch die Rechte ändern (ev. auch nicht – ausprobieren):

```
chmod -R 777 /usr/local/mysql/
```

und starten Sie den Server:

```
/usr/local/mysql/current/bin/./safe_mysqld &

( ./ bedeutet dass dieser Befehl in dem Ordner gestartet werden soll)
( &  bedeutet dass der Prozess im Hintergrund laufen soll)
```

(Falls MySQL nicht startet, sondern sich selbst immer beendet dann muss die Rechtevergabe noch mal durchgeführt werden, vielleicht erst Apache installieren und es später noch mal versuchen.)

Zum Abschluss der Konfiguration müssen Sie einen `root`-user für MySQL aktivieren. Das geschieht, indem Sie ein Initialpasswort vergeben:

```
/usr/local/mysql/current/bin/./mysqladmin -u root password 'geheim'
```

Nun können Sie testen, ob MySQL richtig funktioniert, in dem Sie Zugriff auf die Steuerungs-Datenbank namens `mysql` nehmen:

```
/usr/local/mysql/current/bin/./mysql -u root -p
Enter password: 'geheim'
```

wenn alles richtig installiert wurde, sollten Sie nun folgende Meldung erhalten:

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.23.55
Type 'help' for help.
```

Zuerst die Datenbank MySQL auswählen:

```
use mysql;
```

und einen Konsolenzugriff bekommen. Testen Sie nun, ob alle Initialen Tabellen korrekt angelegt wurden, indem Sie auf der Konsole folgenden Befehl eingeben:

```
show tables;
```

Das Ergebnis dieser Abfrage sollte wie folgt aussehen:

```
+-----------------------+
| Tables in mysql       |
+-----------------------+
| columns_priv          |
|                       |
|                       |
+-----------------------+
6 rows in set (0.01 sec)
```

Sie sollten zusätzlich alle anderen Benutzer außer `root` löschen. Gehen Sie wie folgt vor:

Sie können sich alle User die existieren anzeigen lassen:

```
Select host, user, password from user;
```

Als nächstes löschen Sie alle Benutzer, wo kein Passwort gesetzt ist:

```
Delete from user where password='';
```

... und wo kein Benutzer definiert ist:


```
Delete from user where user='';
```

Schließlich können Sie sich die Tabelle noch mal anzeigen lassen. Es sollte nur noch ein Benutzer zu sehen sein, `root` !

```
Select host, user, password from user;
```

Wenn alles in Ordnung ist, können Sie die MySQL-Konsole wieder verlassen:

```
exit;
```

Als letztes sollen alle Befehle die man unter `/usr/local/mysql/current/bin/` finden kann, überall verfügbar machen:

```
cd /etc
vi profile
```

Dort den Eintrag `PATH=/usr/local/bin:/...` suchen und dahinter folgedes ergenzen:

```
:/usr/local/mysql/current/bin
```

Ab jetzt kann man das `./` vor dem Befehl weglassen, also `mysql -u root -p` und zwar im jeden Verzeichnis.

(Nicht vergessen bei einem Neustart von Linux, wenn man kein automatisches Starten von MySQL aktiviert hat, muß man MySQL vorher starten: `usr/local/mysql/3.23.55/bin/safe_mysqld &`**)**

Falls Sie möchten, dass MySQL auch beim hochfahren automatisch startet, dann kopieren Sie das Startup-File an die entsprechende Stelle (Für den PhPepperShop sinnvoll):

```
cd usr/local/src/lamp/mysql-3.23.55/support-files
cp mysql.server /etc/init.d
chmod 744 /etc/rc.d/mysql.server
cd /etc/rc.d/rc5.d
ln -s /etc/init.d/mysql.server S20mysql.server
ln -s /etc/init.d/mysql.server K20mysql.server
```

Fertig !

OpenSSL

OpenSSL wird hier unterhalb von `/usr/local/openssl/0.9.6d` installiert. OpenSSL wird ja, wie oben schon gesagt, von mod_ssl gebraucht, daher zuerst OpenSSL installieren.

```
cd /usr/local/src/lamp/openssl-0.9.6d
./config --prefix=/usr/local/openssl/0.9.6d
make
make test
make install
ln -s /usr/local/openssl/0.9.6d /usr/local/openssl/current
```

Apache und SSL

Zunächst wird der Apache zusammen mit mod_ssl gebaut (kompiliert und gelinkt). Wir erstellen eine DSO-Version des Apache Webservers, was im wesentlichen heißt, dass Zusatzmodule wie z.B. PHP nicht fest in den Server mit eingebaut werden, sondern vom Server je nach Bedarf dynamisch hinzu geladen werden. Das hat in der Anwendung den Effekt, daß man beim Update von PHP nur noch PHP neu bauen muss, und nicht mehr den ganzen Apache.

- **mod_ssl und Apache vorbereiten**

  Hier werden zunächst die Quellen des Apache gepatcht. Praktischerweise werden gleichzeitig schon alle für den Apache-Configure-Aufruf notwendigen Parameter mitgegeben. D.h. es werden in einem Schritt die Apache-Quellen gepatcht und konfiguriert. (Alle Anweisungen ab dem Präfix sind für den Apache Webserver).

  Besonderheiten: Es wird mit `datadir` als Pfad für die Webdokumente `/usr/local/apache/htdocs` angegeben, das heisst die Dokumente, die über den Webserver erreichbar sein werden, müssen in diesem Verzeichnis liegen. Hier bei der Vorbereitung wird aber `--datadir=/usr/local/apache` angegeben, da Apache bei der Installation `htdocs` (hyper-text-documents) automatisch erstellt.

  ```
  cd /usr/local/src/lamp/mod_ssl-2.8.10-1.3.27
  ./configure --with-apache=../apache_1.3.27 \
  --with-ssl=../openssl-0.9.6d \
  --prefix=/usr/local/apache/1.3.27 \
  --datadir=/usr/local/apache
  --enable-module=most \
  --enable-shared=max \
  --enable-module=ssl
  ```

- **Apache bauen**

  Nun braucht nur noch der Apache gebaut werden. Ein Befehl, gut zu schaffen.

  ```
  cd /usr/local/src/lamp/apache_1.3.27
  make
  ```


- **Zertifikat erzeugen**

  Es wird ein Zertifikat zur späteren Verwendung für SSL-Sitzungen erzeugt. Dieses wird von der eigenen CA (’Behörde’ welche Zertifikate erstellen darf) unterschrieben, die in diesem Schritt gleich mit angelegt wird.

```
make certificate TYPE=custom
```

  Dabei wie folgt auf die Fragen antworten (Anführungszeichen weglassen), und natürlich was Sinnvolles einsetzen.

  Zuerst wird nach Angaben zu der CA gefragt (Schritte 1 - 4).

```
Signature Algorithm: R
Country Name: "DE"
State or Province: "NRW"
Localty Name: "Dortmund"
Organization Name: "Meinefirma"
Organizational Unit Name: "CA"
Common Name: "Meinefirma CA"
Email Address: "ca@meinefirma.de"
Certificate Validity: "365"

Certificate Version: 3
```

  Dann wird Schlüssel und selbssigniertes Zertifikat der CA erzeugt. Hiernach wird dann nach Angaben zum Serverzertifikat gefragt.

```
Country Name: "DE"
State or Province: "NRW"
Localty Name: "Dortmund"
Organization Name: "Meinefirma"
Organizational Unit Name: "Webmaster"
```

  Und jetzt ganz wichtig: den Namen angeben, unter dem später Dein Rechner per SSL erreichbar sein wird (bei mir: https://intern.mediathek.de).

```
Common Name: "testserver.meinefirma.de"
Email Address: "info@meinefirma.de"
Certificate Validity: "365"

Certificate Version: 3
```

  Zuletzt wird gefragt, ob die Schlüssel verschlüsselt werden sollen. Zunächst wird nach der CA gefragt, dann nach dem Serverschlüssel. Da es sich bei mir um ein Testsystem handelt,

wähle ich keine Verchlüsselung. Bei einem Produktionssystem sollte natürlich die Verschlüsselung eingeschaltet werden, wenngleich dann auch bei jedem Neustart des Apache das Passwort für den Server-Key eingeben werden muss. Dies lässt sich aber auch automatisieren.

Kein Test deshalb `yes`!

```
Encrypt the private key now? [Y/n]: y
read RSA key
writing RSA key
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
Fine, you're using an encrypted private key.
Encrypt the private key now? [Y/n]: y
read RSA key
writing RSA key
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
Fine, you're using an encrypted RSA private key.
```

```
RESULT: CA and Server Certification Files

o  conf/ssl.key/ca.key
   The PEM-encoded RSA private key file of the CA which you can
   use to sign other servers or clients. KEEP THIS FILE PRIVATE!

o  conf/ssl.crt/ca.crt
   The PEM-encoded X.509 certificate file of the CA which you use to
   sign other servers or clients. When you sign clients with it (for
   SSL client authentication) you can configure this file with the
   'SSLCACertificateFile' directive.

o  conf/ssl.key/server.key
   The PEM-encoded RSA private key file of the server which you configure
   with the 'SSLCertificateKeyFile' directive (automatically done
   when you install via APACI). KEEP THIS FILE PRIVATE!

o  conf/ssl.crt/server.crt
   The PEM-encoded X.509 certificate file of the server which you configure
   with the 'SSLCertificateFile' directive (automatically done
   when you install via APACI).

o  conf/ssl.csr/server.csr
   The PEM-encoded X.509 certificate signing request of the server file which
   you can send to an official Certificate Authority (CA) in order
   to request a real server certificate (signed by this CA instead
   of our own CA) which later can replace the conf/ssl.crt/server.crt
   file.

Congratulations that you establish your server with real certificates.
```

- **Apache installieren**

  Nun noch schnell den Apache Webserver installieren.

```
make install
ln -s /usr/local/apache/1.3.27 /usr/local/apache/current
```


Libraries für Zusatzfunktionen in PHP bauen

Zunächst erstellen wir einmal die von PHP benötigten Programme und Libraries. Meiner Erfahrung nach kommt es hier eher zu Stress als nachher bei der eigentlichen Erstellung von PHP selbst. Was bedeutet, dass man hier auch gut einen Teil überspringen kann, wenn man ihn nicht braucht. Im Augenblick ist die Entscheidung eher übersichtlich - wir bauen nur die GD-Library. Es kann auch gut sein, dass diese schon bei deiner Distro (Linux Distribution) dabei ist, und du die vorhandene Version verwenden möchtest. Dann überspringe einfach diesen ganzen Schritt.

- **Libpng**

  Im `libpng` Source Unterverzeichnis `scripts` muss man sich zuerst ein passendes `Makefile` aussuchen und ins `libpng` Source Verzeichnis kopieren. Hier ein Beispiel für MMX-Prozessoren (so ziemlich alle ab Intel Pentium 200MHz):

  ```
  cd /usr/local/src/lamp/libpng-1.2.4
  cp ./scripts/makefile.gcmmx ./makefile
  make
  make install
  ```

- **PDFlib**

  Im PDFlib Source Verzeichnis:

  ```
  cd /usr/local/src/lamp/pdflib-4.0.3
  ./configure --prefix=/usr/local
  make
  make install
  ```

- **Freetype**

  Installation unter `/usr/local` . Freetype wird bei der Installation von GD gebraucht (ermöglicht das Benutzen von TrueType Schriftarten – kann auch weggelassen werden, wenn man diese Funktionalität nicht benötigt).

  ```
  cd /usr/local/src/lamp/freetype-2.1.2
  ./configure --prefix=/usr/local
  make
  make install
  ```

  Kommt es an dieser Stelle zu einem Abbruch bei dem Meldungen in der Form "`X11/cursorfont.h: No such file or directory`" auftauchen, so liegt es daran, dass die notwendigen X-Pakete nicht installiert sind. Ein guter Kandidat ist das `'xdevel'-rpm`. Solltest Du dieses nachinstallieren, musst Du im …`/lamp/freetype`… Verzeichnis die Dateien `config.status` und `config.cache` löschen, bevor Du mit '`make clean`' und '`./configure`' von vorne beginnst.

- **Zlib**

  Hier wird die zlib unter `/usr/local` installiert.

  ```
  cd /usr/local/src/lamp/zlib-1.1.4
  ```



```
./configure

make
make install
```

- **IJG JPEG software**

  Die libjpeg wird unter `/usr/local` installiert. Stört sich eigentlich auch nicht mit der von SUSE installierten libjpeg.

```
cd /usr/local/src/lamp/jpeg-6b
./configure --enable-shared --enable-static --prefix=/usr/local

make
make test
make install
```

- **GD**

  Wichtig: Das `Makefile` ist wahrscheinlich schon fertig, nur überprüfen!!!

```
cd /usr/local/src/lamp/gd-1.8.4
vi Makefile
```

  Hier muss das `Makefile` angepasst werden. Wichtig sind die Zeilen mit `CFLAGS` und `LIBS`. Die Defaulteinstellung benutzt nicht libjpg, libpng, freetype - was wir aber genau benutzen wollen. Daher die vorgegebenen Zeilen kommentieren, und die Alternativen auskommentieren.

  Bei den `INCLUDEDIRS` und den `LIBDIRS` sollte man jeweils einen Zusatz in der Form:

  `-I/usr/local/include` beziehungsweise `-L/usr/local/lib oder -I/usr/local/include/freetype2` machen.

  Nun sollte noch überprüft werden, ob auch folgende Pfade vorkommen:

```
INSTALL_LIB=/usr/local/lib
INSTALL_INCLUDE=/usr/local/include
INSTALL_BIN=/usr/local/bin
```

```
make
make install
```


PHP vorbereiten und kompilieren

Hier ist wichtig, daß der config-file-path auf `/etc` gesetzt ist. Man sollte sich diesen Ort unbedingt merken, sonst wird man später verzweifeln.

```
cd /usr/local/src/lamp/php-4.2.2
./configure \
--with-apxs=/usr/local/apache/current/bin/apxs  \
--with-mysql=/usr/local/mysql/current  \
--with-zlib \
--with-ftp \
--with-gd \
--with-jpeg-dir=/usr/local/lib \
--with-png-dir=/usr/local/lib \
--with-pdflib \
--enable-versioning \
--enable-track-vars=yes \
--enable-url-includes \
--enable-sysvshm=yes \
--enable-sysvsem=yes \
--with-config-file-path=/etc

make
```

PHP installieren, Restarbeiten und Apache starten

Nun wird das PHP-Modul installiert, und das wichtige PHP-Handbuch noch in den Pfad für die Webdokumente kopiert. Außerdem wird eine `php.ini` (Konfigurationsdatei für PHP) nach `/etc` kopiert.

```
make install
cp /usr/local/src/lamp/php-4.2.2/php.ini-dist /etc/php.ini
```

Falls Sie möchten, dass Apache auch beim hochfahren automatisch startet:

```
ln -s /usr/local/apache/current/bin/apachectl /etc/init.d/apache

cd /etc/init.d/rc5.d
```

Eventuell wird noch ein alter Apache beim Start automatisch aufgerufen - daher bei Bedarf die entsprechenden alten Einträge löschen:

```
rm S20apache
rm K20apache
```

…und einen neuen eintragen:

```
ln -s /etc/init.d/apache S20apache
ln -s /etc/init.d/apache K20apache
```

„`S`“ steht für starten und „`K`“ für stoppen eines Prozesses. Die Nr. gibt die Reihenfolge der Prozesse, die gestartet werden.

Arbeitet man auf der Maschine lokal, und loggt sich grafisch ein, kann man die links entsprechend auch noch für den Runlevel 3 setzen (Runlevel mit grafischem Login = 5).

```
cd /etc/init.de/rc3.d

rm S20apache
rm K20apache

ln -s /etc/init.d/apache S20apache
ln -s /etc/init.d/apache K20apache
```

Zuletzt sollte man in der `/usr/local/apache/current/bin/apachctl` noch einen Parameter "`-DSSL`" in den httpd-Aufruf unterhalb der Abteilung start einbauen. Dies ist notwendig, damit der Apache auch mit SSL-Unterstützung angestartet wird.

```
cd /usr/local/apache/current/bin/
vi apachctl

start)
if [ $RUNNING -eq 1 ]; then
        echo "$0 $ARG: httpd (pid $PID) already running"
        continue
fi
#Original:
#if $HTTPD ; then
#gaendert, -DSSL eingefuegt:
if $HTTPD -DSSL; then
        echo "$0 $ARG: httpd started"
else
        echo "$0 $ARG: httpd could not be started"
        ERROR=3
Fi
;;
```

Konfiguration

Nun müssen eigentlich nur noch zwei Schritte unternommen werden: Konfiguration der `httpd.conf` für den Apache und Konfiguration der php.ini.

1. **httpd.conf**

   Editiere die `/usr/local/apache/current/conf/httpd.conf`, die zentrale Konfigurationsdatei für deinen Apache…

   ```
   cd /usr/local/apache/current/conf/

   vi httpd.conf
   ```

   …suchen Sie die Variable für den Servernamen, indem sie in VI den Slash / eingeben und dann "ServerName". Bestätigen sie das mit ENTER oder Return und `VI` sucht diese Variable. Wenn der Servername kommentiert, also mit einer Raute # versehen ist, entfernen sie diese mit der Taste x. Die Konfiguration an dieser Stelle sollte nun so aussehen:


```
ServerName ihrserver (webshopserver)
```

Damit ihr Apache-Server auch .php, .php3, .php4 und .phtml Endungen einer Index-Datei erkennt und sie auf ihre Webseiten ohne Angabe der Index-Datei zugreifen können, müssen sie dies dem DirectoryIndex hinzufügen (suchen Sie in VI nach "DirectoryIndex" und schalten sie in den Editiermodus mit der Taste i). Danach sollte dieser Parameter wie folgt aussehen:

```
DirectoryIndex index.html index.php index.php3 index.php4 index.phtml index.htm
```

Sie verlassen sie Editiermodus, indem Sie die ESC-Taste (oben links) drücken. Zum Abschluß der Konfiguration müssen sie dem WebServer noch beibringen, welche Endungen von dem PHP4-Modul interpretiert werden sollen. Suchen sie mit VI nach "AddType" und fügen Sie die gewünschten Änderungen wie hier zu sehen hinzu:

Was nicht zu finden ist, muss dazu geschrieben werden !

```
AddType application/x-tar .tgz
AddType application/x-httpd-php-source .phps
AddType application/x-httpd-php .php .phtml .php3 .php4 .html .htm
```

Als nächstes suchen Sie nach folgender Zeile:

```
<IfDefine SSL>
LoadModule ssl_module              libexec/libssl.so
LoadModule php4_module             libexec/libphp4.so
</IfDefine>
```

Aus dieser kopieren Sie die Zeile
`LoadModule php4_module libexec/libphp4.so`
und schreiben Sie noch mal außerhalb der `<IfDefine SSL>` klammern.

Dasselbe tun sie mit der „`AddModule mod_php4.c`“:

```
<IfDefine SSL>
AddModule mod_ssl.c
AddModule mod_php4.c
</IfDefine>
```

Verlassen Sie den VI-Editor nun, indem sie die Kombination: `wq!` Oder `ZZ` eingeben und somit die Änderungen speichern.

2. **php.ini**

Die /etc/php.ini sollte soweit eigentlich ganz in Ordnung sein. Trotzdem mal reinschauen. Erläuterungen findest Du im PHP3-Manual

(Nur für den PhPepperShop: In der `php.ini` den *Safe Mode* einschalten. Dies macht man, indem man einen Editor (z.B. vi, pico) nimmt und die Zeile sucht, in welcher `SafeMode=Off` steht und diesen dann auf `SafeMode=On` stellt.

Außerdem muß die Zeile 'register_globals = Off' auf 'register_globals = On' geändert werden. Danach muss der Webserver neu gestartet werden. Wenn auf deinem Server nur der PhPepperShop und PHP-Programme laufen, die (sicherheitsmässig) sauber programmiert sind, stellt die Erlaubnis, register_globals zu verwenden kein Sicherheitsproblem dar. Ab v.1.3 des PhPepperShops sind diese Einstellungen ohnehin hinfällig.

Starten

Und nun das hoffentlich finale Ereignis. Den Server starten.

```
/etc/init.d/./apache start
```

Falls die Apache mit SSL Unterstützung starten wollen:

```
/etc/init.d/./apache startssl
```

…und die Seite aufrufen

```
lynx localhost
```

Falls es nicht funktioniert, ist /usr/local/apache/current/logs/error_log die erste Anlaufstelle.

Wenn beim Start vom Apache eine Fehlermeldung der Form:

```
Cannot load /usr/local/apache/current/libexec/libphp4.so into server:
libmysqlclient.so.10: cannot open shared object file: No such file or directory
/etc/init.d/apachectl start: httpd could not be started
```

…erscheint, dann sollte man in der Datei /etc/ld.so.conf noch den Pfad

```
/usr/local/mysql/current/lib/mysql
```

hinzufügen und mit dem Aufruf von

```
ldconfig
```

die Bibliotheken verfügbar machen.

TEST

Zum Testen, ob alles funktioniert, brauchen Sie eine Datei mit dem phpinfo()-Befehl. Diese legen sie wie hier beschrieben an. Erstellen sie eine Datei test.php mit dem VI-Editor.

```
cd /usr/local/apache/htdocs
vi test.php
```

und schreiben Sie diese einfachen Zeilen hinein:

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


Speichern Sie nun die Datei wie beschrieben und verlassen sie den VI Editor. Falls der Server nicht automatisch gestartet wurde, können sie ihren Server starten, indem Sie folgendes Kommando benutzen:

```
/etc/init.d/apache start
```

Nun sollte Apache standardmäßig auf Port 80 laufen. Sie überprüfen ihre Installation nun, indem Sie einen Internetbrowser starten und die Adresse ihres Servers mit der `test.php` eingeben:

```
lynx http://localhost/test.php
```

Wenn Sie nun die PHP4-Infodatei sehen, haben Sie es *geschafft!*

Sie können noch die Apache-Testseiten aus dem Webverzeichnis löschen:

```
cd /usr/local/apache/htdocs
rm index.*
```

FERTIG !

Tipp: Suchen unter linux: `find / -name „Datei“ -print`

Installation des PhPepperShops

Entpacken des Archives in „/usr/local/apache/current/htdocs/".

```
cd /usr/local/apache/current/htdocs
tar xvzf /usr/local/src/lamp/phpeppershop_v_1_2.tar.gz
```

Beim entpacken erstellt das Programm einen Ordner, indem die entpackten Deteien kommen „/usr/local/apache/current/htdocs/phpeppershop_src".

Als nächstes in das Verzeichnis wechseln und das Installations-Script starten (Siehe auch PhPepperShop Manuals/ Installation):

```
cd /usr/local/apache/current/htdocs/phpeppershop_src
perl ./config.pl
```

```
                                                    |-----------------|
PhPepperShop Konfiguration (MySQL-DB)               |Teil 1 - Eingaben|
Abbruch mit CTRL+C                                  |-----------------|
====================================
```

```
ACHTUNG:
Sind Sie in ihrem Webverzeichnis? - wenn nicht, kopieren sie dieses Verzeichnis
zuerst dorthin. Ansonsten ist ihr Shop nach der Installation nicht per Web zugaenglich.

Name der Shop-Datenbank (wird auch in der Shop-URL stehen): shop

Geben Sie nun den Standort ihrer Datenbank ein. Ist die Datenbank auf dem gleichen
Rechner, auf welchem Sie diesen Shop installieren und betreiben, so geben Sie hier
localhost ein. Ist die Datenbank auf einem anderen Rechner, so geben Sie hier seine
Adresse an (z.B. unidb.web.ch).
Datenbank Hostname: localhost

Ein MySQL Monitor wurde unter /bin gefunden, wollen Sie diesen benutzen?
(Empfohlen: ja, j/n): j

Sollen fuer den Shop ein oder zwei Datenbank-User erstellt werden?

Wenn sie schon zwei Datenbank-User haben oder zwei erzeugen lassen wollen,
waehlen sie zwei. Wir empfehlen dies, da nur so unser Security Konzept
vollstaendig umgesetzt werden kann!
Ein oder zwei Datenbank-User benutzen? (1|2): 2

Sollen die User von den Skripten automatisch erzeugt (und bei der Deinstallation
wieder geloescht) werden?

Wenn sie keine Datenbank-User anlegen duerfen, waehlen sie n, ebenfalls wenn
die User schon existieren. Geben sie n für nein ein, ansonsten ein j: (j|n): j

MySQL Admin Loginname: phpshopadmin

MySQL Admin Passwort: 123456

MySQL Shopuser Loginname: shopuser

MySQL Shopuser Passwort: 123456

-> Alle Eingaben erfasst:      (Bitte aufschreiben!)
     Name der Shop Datenbank:    shop
     Anzahl DB-User:             2
     DB-Hostname:                localhost
     DB-Admin Login:             phpshopadmin
     DB-Admin Pwd:               123456
     Shopuser:                   shopuser
     Shopuser Pwd:               123456
     User automatisch erstellen: j
     Pfad zum MySQL Monitor:     /usr/local/mysql/current/bin

Um fortzufahren bitte Enter oder Return druecken
```



```
                                    |---------------------------------------|
PhPepperShop File-Struktur          |Teil 2 - Verzeichnisse/Dateien erstellen|
erstellen                           |---------------------------------------|


Es wird nun vom Shoptemplate ausgehend ihr Shop ('shop') erzeugt.
Dazu wird ein Verzeichnis shop angelegt, worin sich danach der
neue Shop befindet

-> Verzeichnis /usr/local/apache/current/htdocs/shop/ erstellt
-> Template-Shop kopiert, es folgt nun die Anpassung des Template-Shops



Um fortzufahren bitte Enter oder Return druecken







                                   |-------------------------------------|
PhPepperShop Konfiguration         |Teil 3 - Konfigurationsfiles erstellen|
==========================         |-------------------------------------|


-> Es werden nun folgende Dateien im Verzeichnis ..shop/database/ erzeugt:
   Diese drei Dateien erzeugen die personalisierte Shop-Datenbank.
   shop_del.sql
   shop_create.sql
-| shop_insert.sql

-> Die Dateien
   initialize.php        im Verzeichnis ..shop/shop/
   ADMIN_initialize.php  im Verzeichnis ..shop/shop/Admin/
-| erzeugen die Datenbank-Connection.

-> Die Datei
   ADMIN_backup.php      im Verzeichnis ..shop/shop/Admin
-| erzeugt ein Datenbank-Backup.
---------------------------------------------------------------------

-> shop_del.sql erzeugt!
-> shop_del_tables_only.sql erzeugt!
-> shop_create.sql erzeugt!
-> shop_insert.sql erzeugt!

-> initialize.php erzeugt!
-> ADMIN_initialize.php erzeugt!

-> remove.pl erzeugt!

-> ADMIN_backup.php erzeugt!


Um fortzufahren bitte Enter oder Return druecken
```



```
                                  |--------------------------------------|
PhPepperShop: chmod               |Teil 4 - Zugriffsrechte konfigurieren |
==================                |--------------------------------------|

Die Shop-Installation wird nun vervollständigt:
Datei und Verzeichnis Zugriffsrechte werden konfiguriert.

Alle Dateien ausfuehrbar machen (chmod 755):-> ausgeführt
Setze spezielle Datei-Attribute:
    ...'Shoproot'-Verzeichnis...
    ...database-Verzeichnis...
    ...shop-Verzeichnis...
    ...Frameset-Verzeichnis...
    ...Admin-Verzeichnis...
    ...Bilder-Verzeichnis...
    ...Buttons-Verzeichnis...
    ...Backups-Verzeichnis...
    ...Produktebilder-Verzeichnis...-> ausgeführt
CVS-Verzeichnisse entfernen:-> ausgeführt


Um fortzufahren bitte Enter oder Return druecken
```

```
                                  |----------------------------------------|
PhPepperShop: MySQL-Shopdb        |Teil 5 - Shop-Datenbank im MySQL erstellen|
==========================        |----------------------------------------|

Nun kommt der letzte Teil der Shop-Installation:

Soll versucht werden, die shop-Datenbank
ins MySQL-DBMS einzufuegen (wenn sie schon existiert,werden nur die Tabellen eingefuegt)? (j|n): j

Welcher MySQL-User soll die shop-Datenbank
ins MySQL-DBMS einfuegen (z.B. root): root

Passwort eingeben: geheim


**********************************************************************
*    ACHTUNG: das Admin-Verzeichnis muss jetzt noch per htaccess     *
* geschuetzt werden! (siehe auch demo_htaccess im Admin Verzeichnis) *
**********************************************************************

Ihr Shop (shop) sollte jetzt einsatzbereit sein.
(Die Shop-Dateien liegen im Verzeichnis ../shop/)
```

(also ein Verzeichnis zurück und dort in shop)

```
Sie können den Shop starten:

lynx localhost/shop/index.html



---Ende der Installation config.pl---
```

Jetzt muss nur noch der `.htaccess` Schutz erstellt werden! Hierzu existiert eine Anleitung auf http://www.phpeppershop.com/ unter der Rubrik Anleitungen.

Quellen

www.baach.de
www.phpeppershop.com

