|  |
| --- |
| Informationstechnologie |
| Dokumentenart | Arbeitsunterlage |
| Titel | **Apache-Schulung: Lektion 1** |
| Nummer | 1 |
| Autor/-in | Christian Folini |
| Kontaktangaben | +41 79 220 23 76christian.folini@post.ch |
| Ausgabestelle | IT222 |
| Geltungsbereich | Internet / WWW |
| Klassifizierung | intern |
| Version | X01.01 |
| Ausgabedatum | 16. Oktober 2015 |
| Ersetzt Ausgabe vom | 9. Oktober 2015 |

Apache Schulungszentrale: <https://bullnix.post.ch/foswiki/bin/view/BasicServices/ApacheSchulungen>

Quellen:
<https://github.com/Apache-Labor/labor/blob/master/labor-00/README.md>

<https://www.netnea.com/cms/apache_tutorial_1_apache_compilieren/>

## Kompilieren eines Apache Webservers

# Was machen wir?

Wir kompilieren einen Apache Webserver für ein Testsystem.

# Warum tun wir das?

Im professionellen Einsatz des Webservers geschieht es regelmässig, dass besondere Bedürfnisse (Security, zusätzliche Debug-Messages, spezielle Funktionalität dank eines neuen Patches etc.) einen zwingen, sich von den Distributionspaketen zu verabschieden und rasch eigene Binaries herzustellen. In diesem Fall ist es wichtig, dass die Infrastruktur bereit steht und man erste Erfahrungen mit dem Kompilieren und Produktivschalten der eigenen Binaries mitbringt. Zudem lässt sich in einem Laborsetup leichter mit selbst kompiliertem Apache arbeiten, was auch beim Debuggen von Vorteil ist.

# Schritt 1: Bereitmachen des Verzeichnisbaumes für den Sourcecode

Prinzipiell spielt es keine grosse Rolle, wo der Sourcecode liegt. Das Folgende ist ein Vorschlag, der sich am [File Hierarchy Standard](http://www.pathname.com/fhs/) orientiert. Der FHS definiert den Pfadbaum eines Unix-Systems; also die Ablagestruktur für sämtliche Dateien.

$> sudo mkdir /usr/src/apache

$> sudo chown `whoami` /usr/src/apache

$> cd /usr/src/apache

### **Schritt 2: Erfüllen der Vorbedingungen apr und apr-util**

Seit dem Erscheinen von Apache 2.4 wird der Apache Webserver ohne zwei wichtige Bibliotheken ausgeliefert, die vormals Teil der Distribution waren. Wir müssen nun apr und apr-util selbst installieren, bevor wir Apache kompilieren können. Bei apr handelt es sich um die Apache Portable Runtime Bibliothek. Sie erweitert den normalen Satz an C-Bibliotheken mit weiteren Funktionen, die von Server-Software typischerweise benötigt wird. Darunter fallen etwa Funktionen zur Verwaltung von Hash-Tabellen oder Listen. Diese Bibliothek wird nicht nur vom Webserver Apache, sondern auch von anderen Projekten der Apache Software Foundation verwendet, weshalb sie aus dem Source-Code von Apache herausgelöst wurde. Wie apr ist apr-util Teil der Portable Runtime Libraries, welche durch apr-util ergänzt werden.

Beginnen wir mit apr und laden das Paket herunter:

$> wget http://mirror.switch.ch/mirror/apache/dist/apr/apr-1.5.2.tar.bz2

Nun laden wir die Checksum der Sourcecodedatei direkt von Apache herunter. Leider bietet [*www.apache.org*](http://www.apache.org) nur eine md5-Checksum für apr an. Wir testen sie dennoch. Sicherheitshalber verwenden wir beim Herunterladen eine gesicherte Verbindung. Ohne https macht diese Überprüfung keinen grossen Sinn. Beide Files, der Sourcecode und die kleine Prüfsummendatei, sollten nebeneinander in /usr/src/apache liegen. Dann lässt sich die Prüfsumme testen:

$> wget https://www.apache.org/dist/apr/apr-1.5.2.tar.bz2.md5

$> md5sum --check apr-1.5.2.tar.bz2.md5

apr-1.5.2.tar.bz2: OK

Das Überprüfen sollte keine Probleme machen, OK. Wir können also mit dem Entpacken, Vorkonfigurieren und dem Kompilieren der apr fortfahren.

$> tar xvjf apr-1.5.2.tar.bz2

$> cd apr-1.5.2

$> ./configure --prefix=/usr/local/apr/

Wir begeben uns nach dem Entpacken in das neue Verzeichnis mit dem Sourcecode und starten configure. Damit konfogurieren wir den Compiler. Wir geben den Installationspfad vor und configure sucht sich dann eine Vielzahl von Informationen und Einstellungen zu unserem System selbst zusammen. Der Configure-Befehl beschwert sich oft über fehlende Komponenten. Ist klar: Ohne funktionierenden Compiler können wir nicht kompilieren und das Configure hat die Aufgabe nachzusehen, ob alles gut beisammen ist.

Sachen, die typischerweise fehlen, sind Folgende:

* build-essential
* binutils
* gcc

Das sind die Paket-Namen auf einer debian-basierten Distribution. Andernorts mögen die Pakete anders heissen. Das Fehlen lässt sich leicht beheben, indem man sie mit den Hilfsmitteln der eigenen Distribution nachinstalliert. Danach configure neu ausführen, eventuell nochmals etwas nachinstallieren und irgendwann läuft das Skript dann erfolgreich durch.

Wenn dies problemlos durchläuft, wovon wir ausgehen, ist es Zeit für das Kompilieren.

$> make

Das dauert einen Moment und wir erhalten die kompilierte apr, die wir umgehend installieren.

$> sudo make install

Wenn dies erfolgreich geschehen ist, verfahren wir mit den apr-util analog.

$> cd /usr/src/apache

$> wget http://mirror.switch.ch/mirror/apache/dist/apr/apr-util-1.5.4.tar.bz2

$> wget https://www.apache.org/dist/apr/apr-util-1.5.4.tar.bz2.md5

$> md5sum --check apr-util-1.5.4.tar.bz2.md5

apr-util-1.5.4.tar.bz2: OK

$> tar xvjf apr-util-1.5.4.tar.bz2

$> cd apr-util-1.5.4

$> ./configure --prefix=/usr/local/apr/ --with-apr=/usr/local/apr/

$> make

$> sudo make install

Wenn das in beiden Fällen geklappt hat, sind wir bereit für den Webserver selbst.

# Schritt 3: Herunterladen des Sourcecodes und Überprüfen der Checksum

Jetzt laden wir den Programmcode vom Netz herunter. Man kann das mit dem Browser direkt von [Apache](https://httpd.apache.org/) tun, oder man schont die Bandbreite des Apache Projektes und zieht ihn mittels wget von einem Mirror.

$> cd /usr/src/apache

$> wget http://mirror.switch.ch/mirror/apache/dist//httpd/httpd-2.4.17.tar.bz2

Der gepackte Sourcecode hat etwa eine Grösse von 5MB.

Nun laden wir die Checksum der Sourcecodedatei direkt von Apache herunter. Dankenswerterweise steht sie immerhin als sha1-Checksum zur Verfügung. Sicherheitshalber verwenden wir dazu wieder eine gesicherte Verbindung. Ohne https macht diese Überprüfung keinen Sinn.

$> wget https://www.apache.org/dist/httpd/httpd-2.4.17.tar.bz2.sha1

$> sha1sum --check httpd-2.4.17.tar.bz2.sha1

httpd-2.4.17.tar.bz2: OK

# Schritt 4: Entpacken und Compiler konfigurieren

Nach der Überprüfung können wir das Paket entpacken.

$> tar xvjf httpd-2.4.17.tar.bz2

Das ergibt etwa 38MB.

Wir gehen nun in das Verzeichnis und konfigurieren den Compiler mit unseren Eingaben und mit Informationen zu unserem System. Anders als bei apr sind unsere Eingaben sehr umfangreich.

$> cd httpd-2.4.17

$> ./configure --prefix=/opt/apache-2.4.17 --with-apr=/usr/local/apr/bin/apr-1-config --with-apr-util=/usr/local/apr/bin/apu-1-config --enable-mpms-shared=event --enable-mods-shared=all --enable-nonportable-atomics=yes

Hier bestimmen wir das Zielverzeichnis für den zukünftigen Apache Webserver; wieder konform mit dem FHS. Darauf folgen zwei Optionen, um die beiden als Vorbedingung installierten Bibliotheken anzubinden. Mittels --enable-mpms-shared wählen wir ein sogenanntes Prozessmodell des Servers aus. Das ist – vereinfacht gesagt – so etwas wie der Motorentyp der Maschine: Benzin oder Diesel. In unserem Fall stehen event, worker, prefork und ein paar experimentelle Motoren zur Verfügung. Wir nehmen hier das Modell event, das unter 2.4 den neuen Standard darstellt und deutlich performanter ist als die übrigen Architekturen. In den Versionslinien 2.0 und 2.2 gab es bei diesem Entscheid deutlich mehr als nur die Performance zu bedenken, aber seit 2.4 hat sich die Problematik deutlich entschärft und wir fahren nun mit event am besten. Mehr Infos zu den verschiedenen Prozessmodellen (MPMs) liefert das Apache Projekt.

Dann bestimmen wir, dass wir alle (all) Module mitkompilieren möchten. Dabei ist zu berücksichtigen, dass all hier nicht wirklich alle bedeutet. Aus historischen Gründen meint all nur sämtliche Kern-Module, was auch schon eine ganze Menge ist. Das Schlüsselwort shared besagt, dass wir die Module separat kompiliert haben möchten, um sie dann einzeln als optionale Module einbinden zu können. Zu guter Letzt folgt mit enable-nonportable-atomics ein Compiler-Flag, das den Compiler instruiert, besondere Optionen zu verwenden, welche nur auf modernen x86-Prozessoren zur Verfügung stehen und sich günstig auf die Performance auswirken.

Beim configure-Befehl des Webservers kann es nun sein, dass weitere Pakete nachinstalliert werden müssen. Dazu zählen etwa die folgenden.

* libpcre3-dev
* libssl-dev
* zlibc
* zlib1g-dev

Je nach Distribution mag das eine oder andere Paket anders heissen.

Generell kommt es beim Kompilieren immer wieder vor, dass Bestandteile fehlen. Bisweilen ist das Nachinstallieren von Paketen schwieriger als in unserem Fall und im schlimmsten Fall kann es sein, dass Versionen inkompatibel sind. Oft findet man im Internet eine Lösung für das Problem, aber bisweilen muss man sich selbst ziemlich tief in das System eingraben, um die Wurzel der Schwierigkeiten zu beseitigen. In unserem einfachen Fall, sollte das aber kein Thema sein.

# Schritt 5: Kompilieren

Nach dem Durchlauf von configure sind wir bereit für den Compiler. Hier sollte nun nichts mehr schief gehen.

$> make

Das dauert wohl einige Zeit aus den 38MB werden knapp 100MB.

# Schritt 6: Installieren

Wenn das Kompilieren geklappt hat, dann installieren wir den selbst gebauten Apache Webserver. Wir müssen das Installieren durch den Superuser vornehmen lassen. Aber danach schauen wir gleich zu, dass wir wieder in Besitz des Webservers kommen. Für ein Testsystem ist das viel praktischer.

$> sudo make install

Auch die Installation dauert eine Weile.

$> sudo chown -R `whoami` /opt/apache-2.4.17

Und jetzt noch ein Kniff: Wenn man professionell mit Apache arbeitet, dann hat man oft mehrere verschiedene Versionen nebeneinander auf der Testmaschine. Verschiedene Versionen, verschiedene Patches, andere Module etc. führen zu recht mühsamen und langen Pfaden mit Versionsnummern und weiteren Beschreibungen. Ich mache es dann jeweils so, dass ich einen Softlink von /apache auf den aktuellen Apache Webserver lege. Dabei ist darauf zu achten, dass auch der Softlink uns und nicht dem root-User gehört (dies wird bei der Konfiguration des Servers wichtig).

$> sudo ln -s /opt/apache-2.4.17 /apache

$> sudo chown `whoami` --no-dereference /apache

$> cd /apache

Unser Webserver hat nun also einen klaren Pfad, der ihn mit der Versionsnummer eindeutig beschreibt. Im Alltag verwenden wir aber einfach /apache für den Zugriff. Das erleichtert die Arbeit.

# Schritt 7: Starten

Dann wollen wir mal sehen, ob die Maschine anspringt. Das müssen wir für den Moment wieder durch den Superuser erledigen lassen:

$> sudo ./bin/httpd -X

Das ist wieder ein Kniff für den Testbetrieb: Apache ist eigentlich ein Daemon der im Hintergrund läuft. Für einfache Tests ist das aber eher nervig, da wir den Daemon andauernd starten, stoppen, neu laden und sonstwie manipulieren müssen. Mit der Option -X teilen wir Apache mit, dass er sich das mit dem Daemon erst mal sparen und dass er schön im Vordergrund als Single-Prozess/-Thread bleiben soll. Auch das hilft bei der Arbeit.

Vermutlich gibt es nun beim Start eine Warnung:

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message

Das ist nicht weiter schlimm und wir können die Warnung für den Moment ignorieren.

# Schritt 8: Ausprobieren

Die Maschine läuft jetzt. Aber funktioniert sie auch? Zeit für den Funktionstest: Wir sprechen den Apache mit dem Browser unter folgendem Link an:

<http://127.0.0.1>

Da erwarten wir dann Folgendes.



Im Browser zeigt der Apache ein erstes Lebenszeichen.

Super! Ziel erreicht: Der selbst kompilierte Apache läuft.

Zurück in die Shell und Abschalten des Servers mit STRG-C oder für uns Schweizer mit CTRL-C.

# Schritt 9 (Bonus): Ansehen des Binaries und der Module

Bevor wir die Lektion abschliessen möchten wir den Server etwas genauer ansehen. Sozusagen mit den Fingern über die Karosserie streichen und die Motorhaube öffnen. Informationen zu unserem Binary erhalten wir wie folgt:

$> sudo ./bin/httpd -V

Server version: Apache/2.4.17 (Unix)

Server built: Oct 16 2015 21:09:49

Server's Module Magic Number: 20120211:47

Server loaded: APR 1.5.2, APR-UTIL 1.5.4

Compiled using: APR 1.5.2, APR-UTIL 1.5.4

Architecture: 64-bit

Server MPM: event

 threaded: yes (fixed thread count)

 forked: yes (variable process count)

Server compiled with....

 -D APR\_HAS\_SENDFILE

 -D APR\_HAS\_MMAP

 -D APR\_HAVE\_IPV6 (IPv4-mapped addresses enabled)

 -D APR\_USE\_SYSVSEM\_SERIALIZE

 -D APR\_USE\_PTHREAD\_SERIALIZE

 -D SINGLE\_LISTEN\_UNSERIALIZED\_ACCEPT

 -D APR\_HAS\_OTHER\_CHILD

 -D AP\_HAVE\_RELIABLE\_PIPED\_LOGS

 -D DYNAMIC\_MODULE\_LIMIT=256

 -D HTTPD\_ROOT="/opt/apache-2.4.17"

 -D SUEXEC\_BIN="/opt/apache-2.4.17/bin/suexec"

 -D DEFAULT\_PIDLOG="logs/httpd.pid"

 -D DEFAULT\_SCOREBOARD="logs/apache\_runtime\_status"

 -D DEFAULT\_ERRORLOG="logs/error\_log"

 -D AP\_TYPES\_CONFIG\_FILE="conf/mime.types"

 -D SERVER\_CONFIG\_FILE="conf/httpd.conf"

Da wird die Version angegeben, wann wir kompiliert haben, apr kommt wieder zur Sprache und weiter unten der MPM event. Ganz unten finden wir übrigens den Hinweis auf das Standard-Konfigurationsfile des Webservers und etwas darüber den Pfad, unter dem wir das Default Errorlog finden können.

Man kann aber noch etwas mehr aus dem System rausholen und ihn etwa nach den Modulen fragen, welche fix in den Server hineinkompiliert sind:

$> sudo ./bin/httpd -l

Compiled in modules:

 core.c

 mod\_so.c

 http\_core.c

Diese und die obenstehenden Informationen helfen bei der Fehlersuche und wenn man einen Bugreport einsenden will. Dies sind typischerweise auch die ersten Fragen, welche von den Entwicklern gestellt werden.

Das Binary selbst (/apache/bin/httpd) ist übrigens ungefähr 2.0MB gross und die Liste der Module sieht folgendermassen aus:

$> ls -lh modules

total 8.8M

-rw-r--r-- 1 myuser dune73 14K Oct 16 21:09 httpd.exp

-rwxr-xr-x 1 myuser root 36K Oct 16 21:16 mod\_access\_compat.so

-rwxr-xr-x 1 myuser root 34K Oct 16 21:17 mod\_actions.so

-rwxr-xr-x 1 myuser root 49K Oct 16 21:17 mod\_alias.so

-rwxr-xr-x 1 myuser root 31K Oct 16 21:17 mod\_allowmethods.so

-rwxr-xr-x 1 myuser root 30K Oct 16 21:17 mod\_asis.so

-rwxr-xr-x 1 myuser root 47K Oct 16 21:16 mod\_auth\_basic.so

-rwxr-xr-x 1 myuser root 102K Oct 16 21:16 mod\_auth\_digest.so

-rwxr-xr-x 1 myuser root 79K Oct 16 21:16 mod\_auth\_form.so

-rwxr-xr-x 1 myuser root 30K Oct 16 21:16 mod\_authn\_anon.so

-rwxr-xr-x 1 myuser root 39K Oct 16 21:16 mod\_authn\_core.so

-rwxr-xr-x 1 myuser root 43K Oct 16 21:16 mod\_authn\_dbd.so

-rwxr-xr-x 1 myuser root 33K Oct 16 21:16 mod\_authn\_dbm.so

-rwxr-xr-x 1 myuser root 33K Oct 16 21:16 mod\_authn\_file.so

-rwxr-xr-x 1 myuser root 54K Oct 16 21:16 mod\_authn\_socache.so

-rwxr-xr-x 1 myuser root 70K Oct 16 21:16 mod\_authz\_core.so

-rwxr-xr-x 1 myuser root 46K Oct 16 21:16 mod\_authz\_dbd.so

-rwxr-xr-x 1 myuser root 37K Oct 16 21:16 mod\_authz\_dbm.so

-rwxr-xr-x 1 myuser root 41K Oct 16 21:16 mod\_authz\_groupfile.so

-rwxr-xr-x 1 myuser root 37K Oct 16 21:16 mod\_authz\_host.so

-rwxr-xr-x 1 myuser root 31K Oct 16 21:16 mod\_authz\_owner.so

-rwxr-xr-x 1 myuser root 31K Oct 16 21:16 mod\_authz\_user.so

-rwxr-xr-x 1 myuser root 129K Oct 16 21:17 mod\_autoindex.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_buffer.so

-rwxr-xr-x 1 myuser root 103K Oct 16 21:17 mod\_cache\_disk.so

-rwxr-xr-x 1 myuser root 229K Oct 16 21:17 mod\_cache.so

-rwxr-xr-x 1 myuser root 108K Oct 16 21:17 mod\_cache\_socache.so

-rwxr-xr-x 1 myuser root 118K Oct 16 21:17 mod\_cgid.so

-rwxr-xr-x 1 myuser root 68K Oct 16 21:17 mod\_charset\_lite.so

-rwxr-xr-x 1 myuser root 33K Oct 16 21:17 mod\_data.so

-rwxr-xr-x 1 myuser root 221K Oct 16 21:17 mod\_dav\_fs.so

-rwxr-xr-x 1 myuser root 83K Oct 16 21:17 mod\_dav\_lock.so

-rwxr-xr-x 1 myuser root 395K Oct 16 21:17 mod\_dav.so

-rwxr-xr-x 1 myuser root 71K Oct 16 21:17 mod\_dbd.so

-rwxr-xr-x 1 myuser root 100K Oct 16 21:17 mod\_deflate.so

-rwxr-xr-x 1 myuser root 36K Oct 16 21:17 mod\_dialup.so

-rwxr-xr-x 1 myuser root 37K Oct 16 21:17 mod\_dir.so

-rwxr-xr-x 1 myuser root 33K Oct 16 21:17 mod\_dumpio.so

-rwxr-xr-x 1 myuser root 34K Oct 16 21:17 mod\_echo.so

-rwxr-xr-x 1 myuser root 32K Oct 16 21:17 mod\_env.so

-rwxr-xr-x 1 myuser root 44K Oct 16 21:17 mod\_expires.so

-rwxr-xr-x 1 myuser root 74K Oct 16 21:17 mod\_ext\_filter.so

-rwxr-xr-x 1 myuser root 42K Oct 16 21:17 mod\_file\_cache.so

-rwxr-xr-x 1 myuser root 62K Oct 16 21:17 mod\_filter.so

-rwxr-xr-x 1 myuser root 73K Oct 16 21:17 mod\_headers.so

-rwxr-xr-x 1 myuser root 30K Oct 16 21:17 mod\_heartbeat.so

-rwxr-xr-x 1 myuser root 79K Oct 16 21:17 mod\_heartmonitor.so

-rwxr-xr-x 1 myuser root 163K Oct 16 21:17 mod\_include.so

-rwxr-xr-x 1 myuser root 85K Oct 16 21:17 mod\_info.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_lbmethod\_bybusyness.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_lbmethod\_byrequests.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_lbmethod\_bytraffic.so

-rwxr-xr-x 1 myuser root 52K Oct 16 21:17 mod\_lbmethod\_heartbeat.so

-rwxr-xr-x 1 myuser root 103K Oct 16 21:17 mod\_log\_config.so

-rwxr-xr-x 1 myuser root 43K Oct 16 21:17 mod\_log\_debug.so

-rwxr-xr-x 1 myuser root 37K Oct 16 21:17 mod\_log\_forensic.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_logio.so

-rwxr-xr-x 1 myuser root 467K Oct 16 21:17 mod\_lua.so

-rwxr-xr-x 1 myuser root 56K Oct 16 21:17 mod\_macro.so

-rwxr-xr-x 1 myuser root 88K Oct 16 21:17 mod\_mime\_magic.so

-rwxr-xr-x 1 myuser root 60K Oct 16 21:17 mod\_mime.so

-rwxr-xr-x 1 myuser root 184K Oct 16 21:16 mod\_mpm\_event.so

-rwxr-xr-x 1 myuser root 136K Oct 16 21:16 mod\_mpm\_worker.so

-rwxr-xr-x 1 myuser root 117K Oct 16 21:17 mod\_negotiation.so

-rwxr-xr-x 1 myuser root 198K Oct 16 21:17 mod\_proxy\_ajp.so

-rwxr-xr-x 1 myuser root 139K Oct 16 21:17 mod\_proxy\_balancer.so

-rwxr-xr-x 1 myuser root 59K Oct 16 21:17 mod\_proxy\_connect.so

-rwxr-xr-x 1 myuser root 40K Oct 16 21:17 mod\_proxy\_express.so

-rwxr-xr-x 1 myuser root 77K Oct 16 21:17 mod\_proxy\_fcgi.so

-rwxr-xr-x 1 myuser root 40K Oct 16 21:17 mod\_proxy\_fdpass.so

-rwxr-xr-x 1 myuser root 131K Oct 16 21:17 mod\_proxy\_ftp.so

-rwxr-xr-x 1 myuser root 114K Oct 16 21:17 mod\_proxy\_html.so

-rwxr-xr-x 1 myuser root 121K Oct 16 21:17 mod\_proxy\_http.so

-rwxr-xr-x 1 myuser root 66K Oct 16 21:17 mod\_proxy\_scgi.so

-rwxr-xr-x 1 myuser root 357K Oct 16 21:17 mod\_proxy.so

-rwxr-xr-x 1 myuser root 59K Oct 16 21:17 mod\_proxy\_wstunnel.so

-rwxr-xr-x 1 myuser root 33K Oct 16 21:17 mod\_ratelimit.so

-rwxr-xr-x 1 myuser root 34K Oct 16 21:17 mod\_reflector.so

-rwxr-xr-x 1 myuser root 41K Oct 16 21:17 mod\_remoteip.so

-rwxr-xr-x 1 myuser root 48K Oct 16 21:17 mod\_reqtimeout.so

-rwxr-xr-x 1 myuser root 40K Oct 16 21:17 mod\_request.so

-rwxr-xr-x 1 myuser root 210K Oct 16 21:17 mod\_rewrite.so

-rwxr-xr-x 1 myuser root 144K Oct 16 21:17 mod\_sed.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_session\_cookie.so

-rwxr-xr-x 1 myuser root 53K Oct 16 21:17 mod\_session\_dbd.so

-rwxr-xr-x 1 myuser root 61K Oct 16 21:17 mod\_session.so

-rwxr-xr-x 1 myuser root 47K Oct 16 21:17 mod\_setenvif.so

-rwxr-xr-x 1 myuser root 32K Oct 16 21:17 mod\_slotmem\_plain.so

-rwxr-xr-x 1 myuser root 59K Oct 16 21:17 mod\_slotmem\_shm.so

-rwxr-xr-x 1 myuser root 52K Oct 16 21:17 mod\_socache\_dbm.so

-rwxr-xr-x 1 myuser root 40K Oct 16 21:17 mod\_socache\_memcache.so

-rwxr-xr-x 1 myuser root 82K Oct 16 21:17 mod\_socache\_shmcb.so

-rwxr-xr-x 1 myuser root 43K Oct 16 21:17 mod\_speling.so

-rwxr-xr-x 1 myuser root 897K Oct 16 21:17 mod\_ssl.so

-rwxr-xr-x 1 myuser root 80K Oct 16 21:17 mod\_status.so

-rwxr-xr-x 1 myuser root 48K Oct 16 21:17 mod\_substitute.so

-rwxr-xr-x 1 myuser root 35K Oct 16 21:17 mod\_unique\_id.so

-rwxr-xr-x 1 myuser root 37K Oct 16 21:17 mod\_unixd.so

-rwxr-xr-x 1 myuser root 34K Oct 16 21:17 mod\_userdir.so

-rwxr-xr-x 1 myuser root 44K Oct 16 21:17 mod\_usertrack.so

-rwxr-xr-x 1 myuser root 27K Oct 16 21:17 mod\_version.so

-rwxr-xr-x 1 myuser root 40K Oct 16 21:17 mod\_vhost\_alias.so

-rwxr-xr-x 1 myuser root 54K Oct 16 21:17 mod\_watchdog.so

-rwxr-xr-x 1 myuser root 69K Oct 16 21:17 mod\_xml2enc.so

Das sind alle Module, welche von Apache zusammen mit dem Server verteilt werden; bekanntlich haben wir bei den zu kompilierenden Modulen die Option all ausgewählt. Weitere Module gibt es von Drittanbietern. Alle unsere Module brauchen wir kaum, aber einige will man fast immer dabei haben: Sie stehen von Beginng weg für eine Einbindung bereit.

# Verweise

* Apache: <http://httpd.apache.org>
* File Hierarchy Standard: <http://www.pathname.com/fhs/>
* Apache ./configure documenation: <http://httpd.apache.org/docs/trunk/programs/configure.html>