

## 3 Erweiterte Installation

### 3.1 Anpassung der Installationspfade

Bevor ich nun auf die unterschiedlichen Anpassungsoptionen des Apache eingehe, möchte ich mich zunächst der Möglichkeit der individuellen Definition der Installationspfade widmen.

Der Apache stellt mehrere Installationslayouts zur Verfügung, d.h. eine Liste mit möglichen, vordefinierten Installationsverzeichnissen, jeweils optimiert für unterschiedliche Betriebssysteme und entsprechende Varianten. Diese Installations-schemata definieren also Zielinstallationsverzeichnisse und entsprechende Unter-verzeichnisse, wie sie standardmäßig auf einer Vielzahl der unterstützten Betriebssysteme Verwendung finden. Die Layouts werden definiert in der Datei *config.layout*, die sich im Hauptverzeichnis des entpackten Quellcodes des Apache befindet. Für den mir vorliegenden Apache 2.0.48 sieht diese Datei wie folgt aus:

```
#
# config.layout -- Pre-defined Installation Path
# Layouts
#
# Hints:
# - layouts can be loaded with configure's
# --with-layout=ID option
# - when no --with-layout option is given, the default
#   layout is `Apache'
# - a trailing plus character (`+') on paths is
#   replaced with a `/<target>' suffix where <target>
#   is currently hardcoded to 'apache2'.
#   (This may become a configurable parameter at some point.)
#
# Classical Apache path layout.
<Layout Apache>
  prefix:          /usr/local/apache2
  exec_prefix:     ${prefix}
  bindir:          ${exec_prefix}/bin
  sbindir:         ${exec_prefix}/bin
  libdir:          ${exec_prefix}/lib
  libexecdir:     ${exec_prefix}/modules
  mandir:          ${prefix}/man
  sysconfdir:     ${prefix}/conf
  datadir:         ${prefix}
  installbuilddir: ${datadir}/build
  errordir:        ${datadir}/error
  iconsdir:        ${datadir}/icons
  htdocsdir:       ${datadir}/htdocs
  manualdir:       ${datadir}/manual
```

```
cgidir:          ${datadir}/cgi-bin
includedir:      ${prefix}/include
localstatedir:  ${prefix}
runtimedir:     ${localstatedir}/logs
logfiledir:     ${localstatedir}/logs
proxycachedir:  ${localstatedir}/proxy
</Layout>

# GNU standards conforming path layout.
# See FSF's GNU project `make-stds' document for details.
<Layout GNU>
prefix:          /usr/local
exec_prefix:     ${prefix}
bindir:          ${exec_prefix}/bin
sbindir:         ${exec_prefix}/sbin
libdir:          ${exec_prefix}/lib
libexecdir:     ${exec_prefix}/libexec
mandir:          ${prefix}/man
sysconfdir:     ${prefix}/etc+
datadir:         ${prefix}/share+
installbuilddir: ${datadir}/build
erroridir:      ${datadir}/error
iconsdir:       ${datadir}/icons
htdocsdir:      ${datadir}/htdocs
manualdir:      ${datadir}/manual
cgidir:         ${datadir}/cgi-bin
includedir:     ${prefix}/include+
localstatedir:  ${prefix}/var+
runtimedir:     ${localstatedir}/run
logfiledir:     ${localstatedir}/log
proxycachedir:  ${localstatedir}/proxy
</Layout>

# Apache binary distribution path layout
<Layout BinaryDistribution>
prefix:          /usr/local/apache
exec_prefix:
bindir:          bin
sbindir:         bin
libdir:          lib
libexecdir:     libexec
mandir:          man
sysconfdir:     conf
datadir:
installbuilddir: build
erroridir:      error
iconsdir:       icons
htdocsdir:      htdocs
manualdir:      manual
```

```

cgidir:          cgi-bin
includedir:      include
localstatedir:  logs
runtimedir:     logs
logfiledir:     logs
proxycachedir:  proxy
</Layout>

```

```
# Mac OS X Server (Rhapsody)
```

```
<Layout Mac OS X Server>
```

```

prefix:          /Local/Library/WebServer
exec_prefix:     /usr
bindir:          ${exec_prefix}/bin
sbindir:         ${exec_prefix}/sbin
libdir:          ${exec_prefix}/lib
libexecdir:     /System/Library/Apache/Modules
mandir:          ${exec_prefix}/share/man
sysconfdir:     ${prefix}/Configuration
datadir:         ${prefix}
installbuilddir: /System/Library/Apache/Build
errordir:        /System/Library/Apache/Error
iconsdir:        /System/Library/Apache/Icons
manualdir:       /System/Library/Apache/Manual
htdocsdir:       ${datadir}/Documents
cgidir:          ${datadir}/CGI-Executables
includedir:      /System/Library/Frameworks/Apache.framework/Versions/2.0/Headers
localstatedir:  /var
runtimedir:     ${prefix}/Logs
logfiledir:     ${prefix}/Logs
proxycachedir:  ${prefix}/ProxyCache

```

```
</Layout>
```

```
# Red Hat Linux 7.x layout
```

```
<Layout RedHat>
```

```

prefix:          /usr
exec_prefix:     ${prefix}
bindir:          ${prefix}/bin
sbindir:         ${prefix}/sbin
libdir:          ${prefix}/lib
libexecdir:     ${prefix}/lib/apache
mandir:          ${prefix}/man
sysconfdir:     /etc/httpd/conf
datadir:         /var/www
installbuilddir: ${datadir}/build
errordir:        ${datadir}/error
iconsdir:        ${datadir}/icons
htdocsdir:       ${datadir}/html
manualdir:       ${datadir}/manual
cgidir:          ${datadir}/cgi-bin

```

```
    includedir:      ${prefix}/include/apache
    localstatedir:   /var
    runtimedir:      ${localstatedir}/run
    logfiledir:      ${localstatedir}/log/httpd
    proxycachedir:   ${localstatedir}/cache/httpd
</Layout>
```

```
# According to the /opt filesystem conventions
```

```
<Layout opt>
    prefix:          /opt/apache
    exec_prefix:     ${prefix}
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/sbin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/libexec
    mandir:          ${prefix}/man
    sysconfdir:     /etc${prefix}
    datadir:         ${prefix}/share
    installbuilddir: ${datadir}/build
    errordir:        ${datadir}/error
    iconsdir:        ${datadir}/icons
    htdocsdir:       ${datadir}/htdocs
    manualdir:       ${datadir}/manual
    cgidir:          ${datadir}/cgi-bin
    includedir:      ${prefix}/include
    localstatedir:   /var${prefix}
    runtimedir:      ${localstatedir}/run
    logfiledir:      ${localstatedir}/logs
    proxycachedir:   ${localstatedir}/proxy
</Layout>
```

```
# BeOS layout...
```

```
<Layout beos>
    prefix:          /boot/home/apache
    exec_prefix:     ${prefix}
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/bin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/libexec
    mandir:          ${prefix}/man
    sysconfdir:     ${prefix}/conf
    datadir:         ${prefix}
    installbuilddir: ${datadir}/build
    errordir:        ${datadir}/error
    iconsdir:        ${datadir}/icons
    htdocsdir:       ${datadir}/htdocs
    manualdir:       ${datadir}/manual
    cgidir:          ${datadir}/cgi-bin
    includedir:      ${prefix}/include
```

```

    localstatedir:    ${prefix}
    runtimedir:       ${localstatedir}/logs
    logfiledir:       ${localstatedir}/logs
    proxycachedir:    ${localstatedir}/proxy
</Layout>

# SuSE 6.x layout
<Layout SuSE>
    prefix:           /usr
    exec_prefix:      ${prefix}
    bindir:           ${prefix}/bin
    sbindir:          ${prefix}/sbin
    libdir:           ${prefix}/lib
    libexecdir:       ${prefix}/lib/apache
    mandir:           ${prefix}/man
    sysconfdir:       /etc/httpd
    datadir:          /usr/local/httpd
    installbuilddir:  ${datadir}/build
    errordir:         ${datadir}/error
    iconsdir:         ${datadir}/icons
    htdocsdir:        ${datadir}/htdocs
    manualdir:        ${datadir}/manual
    cgidir:           ${datadir}/cgi-bin
    includedir:       ${prefix}/include/apache
    localstatedir:    /var
    runtimedir:       ${localstatedir}/run
    logfiledir:       ${localstatedir}/log/httpd
    proxycachedir:    ${localstatedir}/cache/httpd
</Layout>

# BSD/OS layout
<Layout BSDI>
    prefix:           /var/www
    exec_prefix:      /usr/contrib
    bindir:           ${exec_prefix}/bin
    sbindir:          ${exec_prefix}/bin
    libdir:           ${exec_prefix}/lib
    libexecdir:       ${exec_prefix}/libexec/apache
    mandir:           ${exec_prefix}/man
    sysconfdir:       ${prefix}/conf
    datadir:          ${prefix}
    installbuilddir:  ${datadir}/build
    errordir:         ${datadir}/error
    iconsdir:         ${datadir}/icons
    htdocsdir:        ${datadir}/htdocs
    manualdir:        ${datadir}/manual
    cgidir:           ${datadir}/cgi-bin
    includedir:       ${exec_prefix}/include/apache
    localstatedir:    /var

```

```
    runtimedir:      ${localstatedir}/run
    logfiledir:      ${localstatedir}/log/httpd
    proxycachedir:   ${localstatedir}/proxy
</Layout>

# Solaris 8 Layout
<Layout Solaris>
    prefix:          /usr/apache
    exec_prefix:     ${prefix}
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/bin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/libexec
    mandir:          ${exec_prefix}/man
    sysconfdir:     /etc/apache
    datadir:         /var/apache
    installbuilddir: ${datadir}/build
    errordir:        ${datadir}/error
    iconsdir:        ${datadir}/icons
    htdocsdir:       ${datadir}/htdocs
    manualdir:       ${datadir}/manual
    cgidir:          ${datadir}/cgi-bin
    includedir:      ${exec_prefix}/include
    localstatedir:  ${prefix}
    runtimedir:     /var/run
    logfiledir:     ${datadir}/logs
    proxycachedir:  ${datadir}/proxy
</Layout>

# OpenBSD Layout
<Layout OpenBSD>
    prefix:          /var/www
    exec_prefix:     /usr
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/sbin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/lib/apache/modules
    mandir:          ${exec_prefix}/share/man
    sysconfdir:     ${prefix}/conf
    datadir:         ${prefix}
    installbuilddir: ${prefix}/build
    errordir:        ${prefix}/error
    iconsdir:        ${prefix}/icons
    htdocsdir:       ${prefix}/htdocs
    manualdir:       ${datadir}/manual
    cgidir:          ${prefix}/cgi-bin
    includedir:      ${exec_prefix}/lib/apache/include
    localstatedir:  ${prefix}
    runtimedir:     ${prefix}/logs
```

```
    logfiledir:      ${prefix}/logs
    proxycachedir:   ${prefix}/proxy
</Layout>
```

```
# Debian layout
```

```
<Layout Debian>
```

```
    prefix:
    exec_prefix:     ${prefix}/usr
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/sbin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/lib/apache2/modules
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/sbin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/lib/apache/modules
    mandir:          ${exec_prefix}/share/man
    sysconfdir:     ${prefix}/conf
    datadir:         ${prefix}
    installbuilddir: ${prefix}/build
    errordir:        ${prefix}/error
    iconsdir:        ${prefix}/icons
    htdocsdir:       ${prefix}/htdocs
    manualdir:       ${datadir}/manual
    cgidir:          ${prefix}/cgi-bin
    includedir:      ${exec_prefix}/lib/apache/include
    localstatedir:   ${prefix}
    runtimedir:      ${prefix}/logs
    logfiledir:      ${prefix}/logs
    proxycachedir:   ${prefix}/proxy
```

```
</Layout>
```

```
# Debian layout
```

```
<Layout Debian>
```

```
    prefix:
    exec_prefix:     ${prefix}/usr
    bindir:          ${exec_prefix}/bin
    sbindir:         ${exec_prefix}/sbin
    libdir:          ${exec_prefix}/lib
    libexecdir:     ${exec_prefix}/lib/apache2/modules
    mandir:          ${exec_prefix}/share/man
    sysconfdir:     ${prefix}/etc/apache2
    datadir:         ${exec_prefix}/share/apache2
    iconsdir:        ${datadir}/icons
    htdocsdir:       ${prefix}/usr/share/apache2/default-site/htdocs
    manualdir:       ${htdocsdir}/manual
    cgidir:          ${prefix}/usr/lib/cgi-bin
    includedir:      ${exec_prefix}/include/apache2
    localstatedir:   ${prefix}/var/run
```

```
runtimedir:      ${prefix}/var/run
logfiledir:     ${prefix}/var/log/apache2
proxycachedir:  ${prefix}/var/cache/apache2/proxy
infodir:        ${exec_prefix}/share/info
installbuilddir:  ${prefix}/etc/apache2/build
errordir:       ${datadir}/error
```

```
</Layout>
```

Die vordefinierten Layouts und die entsprechenden Installationsverzeichnisse benutzen Werte, die unter den gelisteten Betriebssystemen typisch sind. Dem einen oder anderen Leser werden die Zielverzeichnisse wahrscheinlich bekannt vorkommen. Einige Layouts (z. B. SuSE) verstreuen die unterschiedlichen Bestandteile des Apache in verschiedene Bereiche des Dateisystems, andere Layouts installieren alle Bestandteile in ein Verzeichnis. Beide Methoden haben sicherlich ihre Vor- und Nachteile, ich habe jahrelang mit dem SuSE-typischen Layout gearbeitet, inzwischen habe ich mich jedoch sehr an das klassische Layout unter `/usr/local/apache` bzw. `/usr/local/apache2` gewöhnt. Schauen wir uns zur Veranschaulichung beispielsweise das klassische Layout *Apache* einmal genauer an:

```
# Classical Apache path layout.
<Layout Apache>
  prefix:          /usr/local/apache2
  exec_prefix:     ${prefix}
  bindir:          ${exec_prefix}/bin
  sbindir:         ${exec_prefix}/bin
  libdir:          ${exec_prefix}/lib
  libexecdir:     ${exec_prefix}/modules
  mandir:          ${prefix}/man
  sysconfdir:     ${prefix}/conf
  datadir:         ${prefix}
  installbuilddir:  ${datadir}/build
  errordir:        ${datadir}/error
  iconsdir:        ${datadir}/icons
  htdocsdir:       ${datadir}/htdocs
  manualdir:       ${datadir}/manual
  cgidir:          ${datadir}/cgi-bin
  includedir:      ${prefix}/include
  localstatedir:   ${prefix}
  rundir:          ${localstatedir}/logs
  logfiledir:     ${localstatedir}/logs
  proxycachedir:  ${localstatedir}/proxy
</Layout>
```

Dieses Layout beginnt mit der Definition der Variable *prefix*, die auf das lokale Verzeichnis `/usr/local/apache2` verweist. Diese Variable gibt das Hauptinstallationsverzeichnis, in das der Apache mit allen Komponenten installiert werden soll, an.

Die folgende Variable *exec\_prefix* verweist ebenfalls auf dieses Verzeichnis und gibt den Speicherort für architekturabhängige Dateien an. In der Praxis ist dieses Ver-

zeichnis oft, wie hier im Beispiel zu sehen, mit dem in der Deklaration der Variable *prefix* gewählten Verzeichnis identisch.

Das bei *bindir* definierte Verzeichnis enthält zwei Shellskripte für den Zugriff auf die *Apache Portable Runtime* (*apr*) und die *Apache Portable Runtime Utils* (*apr-util*). Dieses Verzeichnis ist im Beispiel identisch mit dem Verzeichnis der Variable *sbindir*, was meiner Meinung nach durchaus sinnvoll ist. Diverse Zusatzprogramme für den Apache (z.B. *htpasswd*, *apxs* etc.) sowie das eigentliche Programm (*httpd*) lagern im *sbin*-Verzeichnis, hier befindet sich dieses Verzeichnis unter  $\${exec\_prefix}/bin$  (entspricht */usr/local/apache2/bin*).

Benötigte Bibliotheken (*apr*, *aprutil* etc.) werden im Unterverzeichnis *lib* des bei *prefix* angegebenen Verzeichnisses gespeichert (*/usr/local/apache2/lib*). Die Module werden, sofern diese als *Dynamic Shared Objects* (*DSO*) vorliegen, im durch die Variable *libexecdir* definierten Unterverzeichnis (z.B. *modules*) gespeichert.

Im Verzeichnis *mandir* werden die Manpages des Apache gespeichert, hier als Unterverzeichnis des durch die Variable *prefix* definierten Verzeichnisses.

*Sysconfdir* bezeichnet das Verzeichnis, welches alle Konfigurationsdateien des Apache (z.B. *httpd.conf*) enthält. Hier wird ein Unterverzeichnis namens *conf* unterhalb des Hauptverzeichnisses gewählt, viele Distributionen verwenden auch */etc* zur Speicherung dieser Dateien.

Das Verzeichnis *datadir* ist normalerweise den Verzeichnissen *installbuilddir*, *errordir*, *iconsdir*, *htdocsdir*, *manualdir* und *cgidir* übergeordnet und entspricht, wie hier im Beispiel, meist dem bei *prefix* angegebenen Verzeichnis (z.B. */usr/local/apache2*).

*Installbuilddir* enthält mehrere Skripte zur Installation diverser Programmteile (u.a. der *DSO*-Module) des Apache. Im Beispiel wird dieses Verzeichnis einfach *build* genannt.

Ruft ein Client z.B. eine nicht (mehr) existierende Datei auf, so erhält er eine entsprechende Fehlermeldung. Diese und andere Fehlermeldungen sind mehrsprachig im Verzeichnis *error* (Variable *errordir*) gespeichert und können, falls gewünscht, individuell angepasst werden. Abermals ist das Verzeichnis *error* im Layout ein direktes Unterverzeichnis des bei *prefix* angegebenen Verzeichnisses.

Die Variable *iconsdir* definiert ein Verzeichnis namens *icons*, in dem diverse kleine Bildchen (so genannte Icons) gespeichert werden sollen. Diese werden u.a. bei Verzeichnislistings angezeigt und zeigen teilweise die Art einer Datei an (z.B. *Icon.movie.gif* für diverse Filmformate).

Das bei *htdocsdir* angegebene Verzeichnis enthält die Inhalte (Bilder, HTML-Dateien etc.) die ins Internet gestellt und veröffentlicht werden sollen (hier: */usr/local/apache2/htdocs*).

Die kompletten Handbücher (eng. manuals) des Apache 2.x werden in ein Unterverzeichnis namens *manual* installiert (Verzeichnis *manualdir*), welches dem bei *datadir* angegebenen Verzeichnis unterstellt ist.

Die *Common Gateway Interface* (*CGI*)-Skripte werden im Verzeichnis *cgi-bin* gespeichert (Variable *cgidir*) und dort mit besonderen Rechten versehen. Ebenfalls ein Unterverzeichnis von *datadir*.

In der Programmiersprache C geschriebene Headerdateien des Apache werden im Verzeichnis *include* gespeichert (Variable *includedir*), welches auch ein Unterverzeichnis des zentralen Speicherortes *prefix* (z.B. */usr/local/apache2*) darstellt.

Das bei *localstatedir* angegebene Verzeichnis ist den Verzeichnissen *runtimedir*, *logfiledir* und *proxycachedir* übergeordnet und entspricht meist dem Hauptverzeichnis (*prefix*, hier: */usr/local/apache2*) des Apache.

Im Unterverzeichnis *logs* des bei *localstatedir* (= *prefix*) angegebenen Verzeichnisses werden neben den normalen Logfiles, bestimmt durch die Variable *logfiledir*, auch laufzeitspezifische Dateien (u.a. Lockfile) gespeichert, wie die Variable *runtimedir* definiert.

Die temporären Dateien, der so genannte Cache, des Proxymodules werden im Verzeichnis von *proxycachedir* gespeichert. Entspricht hier einem eigenen Verzeichnis namens *proxy* unterhalb des Basisinstallationsverzeichnisses des Apache.

### 3.1.1 Erstellung eines eigenen Installationslayouts

Falls Sie mit den vorgegebenen Layouts nicht zufrieden sind, können Sie sich sehr leicht ein eigenes Installationslayout erstellen. Editieren Sie die Datei *config.layout* und fügen Sie einen neuen Bereich ein, beispielsweise durch Kopieren eines bereits vorhandenen Layouts. Ich habe das Standardlayout des Apache kopiert, entsprechende Änderungen vorgenommen und dieses Layout *MeinIndianer* genannt:

```
# MeinIndianer
<Layout MeinIndianer>
  prefix:           /usr/local/apache2
  exec_prefix:      ${prefix}
  bindir:           ${exec_prefix}/bin
  sbindir:          ${exec_prefix}/bin
  libdir:           ${exec_prefix}/lib
  libexecdir:       ${exec_prefix}/modules
  mandir:           ${prefix}/man
  sysconfdir:       /etc/apache2
  datadir:          ${prefix}
  installbuilddir:  ${datadir}/build
  errordir:         /var/www/error
  iconsdir:         /var/www/icons
  htdocsdir:        /var/www
  manualdir:        ${datadir}/manual
  cgidir:           ${htdocsdir}/cgi-bin
  includedir:       ${prefix}/include
  localstatedir:    ${prefix}
  rundir:           ${localstatedir}/logs
  logfiledir:       ${localstatedir}/logs
  proxycachedir:    ${localstatedir}/proxy
</Layout>
```

Geändert habe ich die Variable *sysconfdir*, da ich Konfigurationsdateien gerne unter dem Verzeichnis */etc* speichere. Zudem habe ich mit *htdocs* den Speicherort für die zu veröffentlichenden Inhalte (html-Dateien, Bilder etc.) nach */var/www* geändert und die Variablen *errordir*, *iconsdir* und *cgidir* direkt diesem Verzeichniswechsel angepasst. Durch ein eigenes Layout können Sie die späteren Zielverzeichnisse der Apache-Installation beliebig bestimmen. Bedenken Sie jedoch bei der Wahl der entsprechenden Verzeichnisse, dass Sie an einigen Dateien des Apache eventuell Änderungen vornehmen und deshalb diese Dateien zu einem späteren Zeitpunkt wieder finden müssen! Ich finde es sehr hilfreich, wenn die Installation unterhalb eines Verzeichnisses vorgenommen wird, da man so sicherlich schnell alle benötigten Dateien und Verzeichnisse findet. Insbesondere wenn der Apache im Zuge eines Updates oder einer Neuinstallation gelöscht werden soll, ist eine auf mehrere Verzeichnisse verteilte Installation oftmals problematisch.

Bitte achten Sie auch darauf, dass der Name eines manuell erstellten Layouts keine Leerzeichen enthalten darf! Sie können das von Ihnen erstellte Layout durch die Option `--enable-layout=Name` an das *configure*-Skript übergeben und damit aktivieren. Die Installation mit meinem selbst erstellten Layout erfolgt beispielsweise durch den Befehl

```
# ./configure --enable-layout=MeinIndianer
```

**Hinweis**

Diesem Befehl können Sie selbstverständlich noch weitere Optionen anfügen.

Die normale Installation des Apache 2 nimmt seinen Lauf und kann durch die Befehle *make* und *make install* abgeschlossen werden. Das eigene Layout wurde ohne Probleme erkannt und akzeptiert:

```
checking for chosen layout... MeinIndianer
checking for working mkdir -p... yes
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
```

## 3.2 Benutzerdefinierte Installation unter Unix/Linux

Die bereits beschriebene minimale Installation ist natürlich für erfahrene Benutzer nicht ausreichend, da diese oft sehr angepasste und individuelle Installationen und Konfigurationen wünschen, die z.B. durch die zielgerichtete Aktivierung von bestimmten Modulen und die Verwendung gewisser Laufzeitverhalten erreicht werden. Der Apache bietet deshalb eine schier unglaubliche Anzahl an Installationsoptionen, die sich in zahlreichen Variationen zusammenstellen lassen. Nachdem Sie

die Software dekomprimiert und in das neu entpackte Verzeichnis *httpd-2.0.48* gewechselt haben, können Sie die Übersicht der zur Verfügung stehenden Optionen durch Eingabe des folgenden Befehls aufrufen:

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

Die Anzahl der Konfigurationsoptionen ist wirklich enorm hoch. Falls Sie diese in einer angenehmen Form lesen möchten, können Sie auch den Befehl

```
# ./configure --help | less
```

nutzen, der Ihnen eine seitenweise Übersicht über die einzelnen Optionen gibt, indem er die Ausgaben des Befehls *configure* durch eine so genannte Pipe in das Programm *less* umleitet, welches schließlich die seitenweise Anzeige übernimmt. Dabei können Sie sich durch Betätigen der Leertaste jeweils die nächste Seite anzeigen lassen, oder mit der **Bild ↑**- und **Bild ↓**-Taste nach oben bzw. unten scrollen. Das Betätigen der Taste **Q** beendet die Auflistung der Optionen durch das Programm *less*.

Bevor Sie sich der Übersetzung einer individuell auf Ihre Wünsche und Bedürfnisse zugeschnittenen Version des Apache widmen können, möchte ich Ihnen zunächst eine Übersicht über die möglichen Konfigurationsoptionen sowie deren Bedeutung geben:

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
-h, --help	Zeigt eine Übersicht der möglichen Konfigurationsoptionen nebst kleinen Erläuterungen.
--help=short	Zeigt eine kurze (engl. short) Übersicht der Optionen, wobei diese sich nur auf die Apache-Module beziehen. Die Ausgabe entspricht ebenfalls der Option <i>--help=recursive</i> .
-V	Versionsinformationen werden angezeigt und das Programm wird wieder beendet.
-q, --quiet, --silent	Manche Benutzer empfinden die Tests, die <i>configure</i> während des Ablaufes durchführt, als störend und möchten diese nicht sehen. Durch Angabe dieses Parameters werden sie nicht mit angezeigt, allerdings geben diese Ausgaben bei Problemen oft sehr hilfreiche Hinweise auf mögliche Fehlerursachen.

*Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«)*

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--cache-file=Datei	Falls man die Ausgaben der durch <i>configure</i> ausgeführten Tests speichern möchte, um diese beispielsweise anderen Programmen oder Benutzern zur Verfügung zu stellen, kann man durch diese Option eine Datei festlegen, in der die Ausgaben gespeichert werden sollen.
-C, --config-cache	Diese Option speichert die Ausgabe der Tests von <i>configure</i> in der Datei <i>config.cache</i> (Standardverhalten).
-n, --no-create	Dadurch werden verschiedene Dateien wie Makefile etc. nicht erzeugt bzw. nicht gelesen (z.B. <i>config.status</i> ), falls diese schon vorhanden sind.
--srcdir=Verzeichnis	Macht eine Angabe über den Speicherort der Quellen des Apache.
--prefix=Präfix	Definition eines Verzeichnisses, in das der Apache installiert werden soll (Standard: <i>/usr/local/apache2</i> ). Siehe auch die Beschreibung der Datei <i>config.layout</i> .
--exec-prefix=Präfix	Architekturabhängige Dateien (z.B. <i>httpd</i> etc.) können in ein spezielles Verzeichnis kopiert werden, sofern dies gewünscht ist. Eigentlich ist dies jedoch nicht notwendig, sie werden unterhalb des mit <i>--prefix</i> angegebenen Verzeichnisses kopiert.
--bindir=Verzeichnis	Verzeichnis für zwei Shellskripte, die Kommandozeilenzugriff auf die <i>Apache Portable Runtime (apr)</i> und die <i>Apache Portable Runtime Utils (apr-utils)</i> ermöglichen. Standard: <i>/usr/local/apache2/bin</i>
--sbindir=Verzeichnis	In diesem Verzeichnis liegen alle ausführbaren Dateien des Apache (z.B. <i>htpasswd</i> , <i>apxs</i> etc.) inklusive des eigentlichen Programms ( <i>httpd</i> ). Dieses Verzeichnis sollte mit dem bei <i>--bin-dir</i> angegebenen Verzeichnis identisch sein, da sonst einige Skripte (z.B. Startskript <i>apachectl</i> ) nicht mehr ohne Benutzereingriff funktionieren.
--libexecdir=Verzeichnis	Verzeichnis für DSO-Module und Bibliotheken. Standard: <i>/usr/local/apache2/libexec</i>

**Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung**  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--datadir=Verzeichnis	Ein Verzeichnis für allgemeine Dateien des Apache, dem untergeordnet sind die Verzeichnisse <i>errordir</i> , <i>iconsdir</i> , <i>htdocsdir</i> , <i>manualdir</i> und <i>cgidir</i> . Entspricht in der Praxis oft dem bei <i>--prefix</i> angegebenen Verzeichnis (Standard: <i>/usr/local/apache2/share</i> ).
--sysconfdir=Verzeichnis	Enthält alle Konfigurationsdateien des Apache (z.B. <i>httpd.conf</i> ) sowie einige Beispiele (Standard: <i>/usr/local/apache2/etc</i> ).
--sharedstatedir=Verzeichnis	Veränderbare Daten, die nicht an eine bestimmte Architektur gebunden sind. In vielen Konfigurationen unbenutzt (Standard: <i>/usr/local/apache2/com</i> ).
--localstatedir=Verzeichnis	Dieses Verzeichnis ist den Verzeichnissen <i>runtime</i> , <i>logfiledir</i> und <i>proxycachedir</i> übergeordnet und entspricht meist dem Hauptverzeichnis ( <i>--prefix</i> ) des Apache (Standard: <i>/usr/local/apache2/var</i> ).
--libdir=Verzeichnis	In diesem Verzeichnis werden die Bibliotheken der <i>Apache Portable Runtime (apr)</i> und der <i>Apache Portable Runtime Utils (apr-util)</i> gespeichert (Standard: <i>/usr/local/apache2/lib</i> ).
--includedir=Verzeichnis	Die Includedateien des Apache, allesamt in C geschriebene Headerdateien, werden in diesem Verzeichnis gespeichert (Standard: <i>/usr/local/apache2/include</i> ).
--oldincludedir=Verzeichnis	Falls man einen anderen ANSI C-Compiler verwendet, als den gcc, so können die Include-dateien auch in diesem Verzeichnis (meist <i>/usr/include</i> ) gespeichert werden.
--infodir=Verzeichnis	Die komplette Dokumentation des Apache im HTML-Format (Standard: <i>/usr/local/apache2/info</i> ).
--mandir=Verzeichnis	Die Manualpages (Online-Handbücher) des Apache (Standard: <i>/usr/local/apache2/man</i> ).
--build=BUILD, --host=HOST, --target=TARGET	Auf manchen Systemen ist es unter Umständen nötig, Parameter der Rechnerarchitektur dem <i>configure</i> -Skript zu übergeben, falls dieses die entsprechenden Parameter nicht selbst anhand einiger lokaler Programme (z. B. <i>uname</i> , <i>arch</i> etc.)

**Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung**  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
	findet. Sollte in den wenigsten Fällen manuell gesetzt werden müssen. Auf meinen Systemen lautete dieser Wert z. B. <i>i686-pc-linux-gnu</i> bzw. <i>i386-pc-solaris2.8</i> .
--disable-FEATURE	Ein bestimmtes Feature abschalten, entspricht der Anweisung <i>--enable-FEATURE=no</i> .
--enable-FEATURE=yes	Ein bestimmtes Feature einschalten, Gegenstück zu <i>--disable-FEATURE</i> .
--enable-layout=LAYOUT	In der Datei <i>config.layout</i> werden verschiedene Installationslayouts vorgestellt und können durch diese Anweisung benutzt werden. Weitere Hinweise zum Aufbau dieser Datei sowie zur Erstellung eigener Layouts finden Sie im vorhergehenden Kapitel.
--enable-maintainer-mode	Schaltet den Entwicklermodus ein und gibt erweiterte Warnmeldungen aus.
--enable-modules=Modulliste	Aktiviert die mit <i>Modulleiste</i> angegebenen Module im Apache. Mehrere Module müssen mit Kommata voneinander getrennt und durch Anführungszeichen eingeschlossen werden.
--enable-mods-shared=Liste	Die hier angegebenen Module werden als <i>Dynamic Shared Objects</i> zur Verwendung mit <i>mod_so</i> übersetzt. Die Anweisung <i>--enable-mods-shared=max</i> sorgt dafür, dass alle Module außer <i>mod_so</i> und das Kernmodul <i>mod_core</i> als <i>Dynamic Shared Objects</i> übersetzt werden.
--disable-access	Deaktiviert (disable) bzw. aktiviert (enable) die von Hostnamen und IP-Adressen abhängige Zugriffskontrolle des Moduls <i>mod_access</i> .
--enable-access	
--disable-auth	Schaltet die Benutzerauthentifizierung durch das Modul <i>mod_auth</i> an bzw. ab.
--enable-auth	
--enable-auth-anon	Falls anonyme Zugriffe auf den Webserver gestattet werden sollen, müssen Sie dieses Modul aktivieren ( <i>--enable-auth-anon</i> ). Ich rate von der Verwendung dieses Moduls, welches vergleichbar ist mit dem unsicheren Anonymous FTP, strikt ab ( <i>--disable-auth-anon</i> ).
--disable-auth-anon	

**Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung**  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-auth-dbm --disable-auth-dbm	Unter Umständen ist die Authentifizierung der Benutzer via datenbankähnlicher DBM-Dateien gewünscht und kann durch <i>--enable-auth-dbm</i> aktiviert werden. Diese Dateien können durch das Programm <i>dbmmanage</i> verwaltet werden.
--enable-auth-digest --disable-auth-digest	Das Modul <i>mod_auth_digest</i> löst die veraltete Basisauthentifikation ( <i>basic authentication</i> ) ab und implementiert die neue <i>MD5-Digest Authentication</i> , die fester Bestandteil der HTTP/1.1 Spezifikation ist. Auch hier aktiviert <i>--enable-auth-digest</i> das Modul, <i>--disable-auth-digest</i> schaltet es ab.
--enable-file-cache --disable-file-cache	Früher wurde das Modul <i>mod_mmap_static</i> verwendet, um beim Start des Apache bestimmte statische Dateien in den Speicher zu laden und somit die Zugriffsgeschwindigkeit auf diese Informationen zu erhöhen. Das Modul wurde aus dem Apache 2 entfernt und durch <i>mod_file_cache</i> ersetzt. Falls Sie statische Informationen haben, auf die Sie schnell zugreifen müssen, aktivieren Sie dieses Modul, ansonsten nicht.
--enable-echo --disable-echo	Der Apache 2.0 bietet im Gegensatz zu den Vorgängerversionen die nötige Infrastruktur, um neben dem HTTP-Protokoll auch weitere Protokolle (z. B. FTP und POP3) zu unterstützen und zu verarbeiten! <i>mod_echo</i> wurde als Beispiel geschrieben und sollte auf einem Produktionssystem nicht installiert werden, da es lediglich zur Veranschaulichung dient. Es kann die Daten einer Clientanfrage gemäß des Echo-Protokolls direkt wieder an diesen zurückliefern. Ein konkretes Beispiel dazu wird im Laufe dieses Buches gegeben.
--enable-charset-lite --disable-charset-lite	Erlaubt Konvertierung von Inhalten in bestimmte Zeichensätze, die vorher definiert werden müssen. Gilt noch als experimentell, <i>--disable-charset-lite</i> und <i>--enable-charset-lite</i> schalten dieses Modul aus bzw. an.
--enable-cache --disable-cache	Falls Sie den Apache auch als Proxyserver betreiben, müssen Sie dieses Modul aktivieren ( <i>--enable-cache</i> ), da es die Cachingfunktionen enthält, die früher direkt in <i>mod_proxy</i> integriert

*Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)*

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-disk-cache --disable-disk-cache	waren. Ansonsten können Sie dieses Modul deaktivieren ( <i>--disable-cache</i> ).  Das Modul ist experimentell und implementiert einen mit RFC 2616 kompatiblen Inhaltsspeicher ( <i>http content cache</i> ) und bedarf zusätzlich mindestens einem Speichermanagementmodul wie <i>mod_mem_cache</i> . Gilt bisher als bedingt stabil.
--enable-mem-cache --disable-mem-cache	<i>Mod_mem_cache</i> implementiert ein RAM basierendes Speichermanagementmodul und ist das Gegenstück zu <i>mod_disk_cache</i> . Gilt ebenfalls noch als experimentell.
--enable-example --disable-example	Diese Option installiert ein Beispielmodul ( <i>mod_example</i> ) für den Zugriff auf die Apache-interne API (Application Programming Interface) auf dem Server. Dieses Modul ist für (Modul-) Entwickler gedacht und hat auf einem Produktivsystem nichts verloren.
--enable-ext-filter --disable-ext-filter	Im Gegensatz zu früheren Versionen kann der Apache ab Version 2.x Daten auch vor der Ausgabe an die Clients an ein externes Programm liefern, dort verarbeiten lassen und endgültig ausgeben. Als Ausgabefilter können beispielsweise Sortierungsprogramme oder Ähnliches verwendet werden, wobei es allgemein jedes Programm sein kann, welches Daten von der Standardeingabe einliest, verarbeitet und in die Standardausgabe schreibt. <i>--enable-ext-filter</i> aktiviert diese Funktionalität, <i>--disable-ext-filter</i> deaktiviert sie.
--enable-case-filter --disable-case-filter	Ein Beispiel für einen Filter, der Daten in Großbuchstaben umwandelt. Aktivierung mit <i>--enable-case-filter</i> , Deaktivierung mit <i>--disable-case-filter</i> .
--enable-case-filter-in --disable-case-filter-in	Ein Beispiel für einen Eingabefilter, der Eingaben in Großbuchstaben umwandelt. Aktivierung mit <i>--enable-case-filter-in</i> , Deaktivierung mit <i>--disable-case-filter-in</i> .
--enable-deflate --disable-deflate	<i>Mod_deflate</i> ist ein Ausgabefilter, der Ausgaben komprimiert, bevor diese an den Client gesendet werden. Mit <i>--enable-deflate</i> wird diese Erweiterung aktiviert, mit <i>--disable-deflate</i> wird sie deaktiviert.

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-include --disable-include	<p>Hinweis: Wenn Sie diesen Ausgabefilter verwenden möchten, müssen Sie ebenfalls das Modul <i>mod_ext_filter</i> aktivieren, da Sie sonst überhaupt keine externen Filter wie <i>mod_deflate</i> verwenden können!</p> <p>Mit diesem Modul können Dokumente auf dem Server ausgeführt werden, bevor diese an den Client gesendet werden. Diese Dokumente werden <i>Server Side Includes (SSI)</i> genannt, inzwischen gibt es für serverseitige Programmierung weitaus bessere Alternativen. Wird mit <i>--enable-include</i> aktiviert, <i>--disable-include</i> deaktiviert den Filter.</p>
--enable-log-config --disable-log-config	<p>Das Modul <i>log_common</i> wurde durch dieses Modul ersetzt. Es stellt u. a. Zugriffsstatistiken über die Art und Häufigkeit der aufgerufenen Webseiten zur Verfügung und zeichnet ebenso eventuell aufgetretene Fehler auf. Dieses Modul sollte unbedingt durch <i>--enable-log-config</i> aktiviert werden!</p>
--enable-env --disable-env	<p>Umgebungsvariablen aus der Shell können an den Apache bzw. an ein auf dem Server laufendes Skript übergeben werden. Ebenso können beliebige Variablen in der Serverkonfiguration gesetzt werden. Der Einsatz dieses Moduls ist dringend empfohlen, die Aktivierung erfolgt mit <i>--enable-env</i>. Falls Sie das Modul nicht benötigen, können Sie es mit <i>--disable-env</i> abschalten.</p>
--enable-mime-magic --disable-mime-magic	<p>Eine automatische Erkennung des korrekten MIME-Typen für eine Datei wird durch dieses Modul bereitgestellt. Sollten Sie auf jeden Fall aktivieren mit <i>--enable-mime-magic</i>.</p>
--enable-cern-meta --disable-cern-meta	<p>Es gibt eventuell Benutzer, die vom CERN HTTPD auf den Apache umgestiegen sind und ihre alten CERN-Meta-Files weiter benutzen möchten. Durch die Aktivierung dieses Moduls mit <i>--enable-cern-meta</i> können diese alten Dateien, die zusätzliche Header an die Clients senden, auch im Apache benutzt werden. Sollten</p>

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-expire --disable-expire	<p>Sie den CERN HTTPD nicht benutzt haben, schalten Sie dieses Modul durch <code>--disable-cern-meta</code> ab.</p> <p>Mit <code>mod_expire</code> kann die Dauer der Gültigkeit einer Information, praktisch deren Lebensdauer, definiert werden. Ein sehr sinnvolles Modul, wenn man Informationen veröffentlicht, die nur einen bestimmten Zeitraum Gültigkeit haben sollen (z.B. Nachrichten, Aktienkurse etc.). Es wird durch die Option <code>--enable-expire</code> eingeschaltet.</p>
--enable-headers --disable-headers	<p>Unter Umständen ist es nötig, die HTTP-Header beliebig zu verändern und Daten zu modifizieren, hinzuzufügen, zu ersetzen oder gar zu löschen. Falls gewünscht, wird diese Option mit <code>--enable-headers</code> eingeschaltet, <code>--disable-headers</code> schaltet sie ab.</p>
--with-gdbm=Pfad	<p>Einige Funktionen des Apache (z.B. <code>RewriteMap</code> von <code>mod_rewrite</code>) benutzen datenbankähnliche DBM-Dateien, die im Vergleich zu reinen Textdateien einen erheblichen Geschwindigkeitsvorteil darstellen. Mit dieser Option setzen Sie den Pfad zur lokalen GDBM-Installation, die für die Benutzung solcher DBM-Dateien vorhanden sein muss. Falls kein Pfad angegeben wird, sucht der Apache in den Suchpfaden des Systems nach der lokalen Installation. Alternativ kann auch einer der zwei anderen DBM-Datenbanktypen (z.B. NDBM) installiert sein.</p>
--with-ndbm=Pfad	<p>Mit dieser Option setzen Sie den Pfad zur lokalen NDBM-Installation, die für die Benutzung von DBM-Dateien vorhanden sein muss. Falls kein Pfad angegeben wird, sucht der Apache in den Suchpfaden des Systems nach der lokalen Installation. Alternativ kann auch einer der zwei anderen DBM-Datenbanktypen installiert sein.</p>
--with-berkeley-db=Pfad	<p>Die Option setzt den Pfad zur lokalen Installation der Berkeley DB, die für die Benutzung von DBM-Dateien vorhanden sein muss. Falls kein Pfad angegeben wird, sucht der Apache in den</p>

**Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung**  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-usertrack --disable-usertrack	Suchpfaden des Systems nach der lokalen Installation. Alternativ kann auch einer der zwei anderen DBM-Datenbanktypen installiert sein.  Früher als <i>mod_cookies</i> bekannt, versucht dieses Modul anhand so genannter Cookies die Aktionen des Benutzers zu verfolgen und aufzuzeichnen. Eigentlich nutzlos, denn viele Benutzer lehnen Cookies aufgrund ihres schlechten Rufes ab, kann mit <i>--disable-usertrack</i> deaktiviert werden.
--enable-unique-id --disable-unique-id	Mit Hilfe dieses Moduls kann für jede Clientanfrage eine einzigartige (engl. unique) Zahl, eine so genannte ID, erzeugt werden. Funktioniert zurzeit nur unter Unix/Linux, nicht unter Windows. Wird mit <i>--enable-unique-id</i> ein bzw. mit <i>--disable-unique-id</i> ausgeschaltet.
--enable-setenvif --disable-setenvif	Anhand der bei einer Clientanfrage zur Verfügung stehenden Informationen (u.a. Browserversion, IP-Adresse etc.) lassen sich mit diesem Modul Umgebungsvariablen setzen. Sollte auf jeden Fall mit <i>--enable-setenvif</i> aktiviert werden.
--enable-proxy --disable-proxy	Aktiviert bzw. deaktiviert die Proxyfunktionalitäten des Apache. Sofern man keinen Proxyserver betreiben möchte oder eine andere Software einsetzt (z. B. Squid), sollte man diese Funktion mit <i>--disable-proxy</i> deaktivieren. Die Aktivierung dieses Moduls integriert automatisch die Module <i>proxy_connect</i> , <i>proxy_ftp</i> und <i>proxy_http</i> in den Server. Unter Umständen benötigen Sie dieses Modul auch in Verbindung mit <i>mod_rewrite</i> .
--enable-proxy-connect --disable-proxy-connect	Dieses Modul implementiert die <i>Connect</i> -Methode für das Apache-Proxymodul. Kann gefahrlos deaktiviert werden mit <i>--disable-proxy-connect</i> , wenn man den Apache nicht als Proxyserver betreibt.
--enable-proxy-ftp --disable-proxy-ftp	FTP-Routinen für das Apache-Proxymodul. Können deaktiviert werden ( <i>--disable-proxy-ftp</i> ), wenn man den Apache nicht als Proxyserver betreibt.

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-proxy-http --disable-proxy-http	HTTP- und HTTPS-Routinen für das Apache-Proxymodul. Können ebenfalls deaktiviert werden ( <i>--disable-proxy-http</i> ), wenn man den Apache nicht als Proxyserver betreibt.
--enable-ssl --disable-ssl	Integriert die SSL-Unterstützung von OpenSSL in den Apache ( <i>--enable-ssl</i> ) bzw. schaltet diese aus ( <i>--disable-ssl</i> ). Wenn sensible Daten transferiert werden müssen (z. B. Bankdaten, Passwörter, Kreditkartendaten etc.), sollte diese Erweiterung zwingend aktiviert werden.
--enable-optional-hook-export --disable-optional-hook-export	Testmodul für Apache-interne Funktionen
--enable-optional-hook-import --disable-optional-hook-import	Testmodul für Apache-interne Funktionen
--enable-optional-fn-import --disable-optional-fn-import	Testmodul für Apache-interne Funktionen
--enable-optional-fn-export --disable-optional-fn-export	Testmodul für Apache-interne Funktionen
--enable-bucketeer --disable-bucketeer	Das Modul <code>mod_bucketeer</code> ist ein Test für einen Ausgabefilter. Es liest eine Textdatei mit speziellen Sonderzeichen (u. a. <code>^P</code> , <code>^F</code> und <code>^Ennn</code> ) ein, um bei Vorhandensein eines solchen Sonderzeichens die Möglichkeit der Ausführung einer Apache-internen Funktion (sog. Bucket) zu geben.
--enable-static-support --disable-static-support	Erzeugt statisch gelinkte Versionen der Supportprogramme ( <i>--enable-static-support</i> ). Bei statisch gelinkten Versionen sind die benötigten Bibliotheken in den fertigen Binaries enthalten, bei dynamisch gelinkten Programmen sind diese nicht im fertigen Programm enthalten, sondern können als Bibliothek von mehreren Programmen benutzt werden und sparen somit Festplatten- und Arbeitsspeicher.
--enable-static-httpasswd --disable-static-httpasswd	Generiert eine statisch gelinkte Version von <i>httpasswd</i> .
--enable-static-htdigest --disable-static-htdigest	Errichtet eine statisch gelinkte Version von <i>htdigest</i> .

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-static-rotatelog --disable-static-rotatelog	Generiert eine statisch gelinkte Version von <i>rotatelog</i> .
--enable-static-logresolve --disable-static-logresolve	Erzeugt eine statisch gelinkte Version von <i>logresolve</i> .
--enable-static-htdbm --disable-static-htdbm	Generiert eine statisch gelinkte Version von <i>htdbm</i> .
--enable-static-ab --disable-static-ab	Linkt <i>ab</i> (Apache-Benchmark) statisch.
--enable-static-checkgid --disable-static-checkgid	Generiert eine statisch gelinkte Version von <i>checkgid</i> .
--enable-http --disable-http	--enable-http aktiviert die HTTP-Protokoll-Unterstützung für den Apache, --disable-http deaktiviert diese Unterstützung (nicht sinnvoll!).
--enable-mime --disable-mime	<i>Mod_mime</i> setzt anhand der Dateiendung einer Datei den entsprechenden MIME-Typen. MIME heißt <i>Multipurpose Internet Mail Extension</i> und sorgt dafür, dass durch vorherige Kennzeichnung des Datentyps ein Client mit empfangenen Daten überhaupt umgehen kann. Dabei wird ein MIME-Typ (z. B. <i>text/html</i> ) übergeben und der Client weiß nun, welche Art von Daten er erhält. Dieses wichtige Modul wird mit <i>--enable-mime</i> aktiviert, deaktiviert wird es mit <i>--disable-mime</i> .
--enable-dav --disable-dav	Aktiviert bzw. deaktiviert die Unterstützung für <i>WebDav</i> ( <i>web-based distributed authoring and versioning</i> ), einem Standard, der von namhaften Firmen wie Microsoft, Netscape, Novell, IBM und Xerox ins Leben gerufen worden ist und der inzwischen von der unabhängigen IETF Web-Dav Working Group weiterentwickelt und überwacht wird ( <a href="http://www.webdav.org">http://www.webdav.org</a> ). Es handelt sich dabei um eine Erweiterung des HTTP-Protokolls (RFC 2518), die es ermöglicht, Dateien und Verzeichnisse direkt über HTTP auf einem Server zu bearbeiten.

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-status --disable-status	Integriert das Modul <i>mod_status</i> in den Apache bzw. lässt es weg. <i>Mod_status</i> gibt Statusinformationen über die Auslastung und die Aktivität des Servers aus.
--enable-autoindex --disable-autoindex	Mit dieser Option kann das Modul <i>mod_autoindex</i> , welches automatisch Verzeichnisindizes erzeugt (vergleichbar mit dem Befehl <i>ls</i> unter Unix/Linux und <i>dir</i> unter Dos/Windows), ein- ( <i>--enable-autoindex</i> ) bzw. abgeschaltet ( <i>--disable-autoindex</i> ) werden.
--enable-asis --disable-asis	Fügt <i>mod_asis</i> dem Apache hinzu ( <i>--enable-asis</i> ) bzw. entfernt ( <i>--disable-asis</i> ) dieses Modul. Es dient zur Sendung von selbstdefinierten HTTP-Headern an den Client.
--enable-info --disable-info	<i>Mod_info</i> gibt einen umfassenden Überblick über die aktuelle Serverkonfiguration, wobei nicht die laufende Konfiguration angezeigt wird, sondern die Konfiguration, die beim Start des Apache zur Verfügung stand. Ändert man die Konfiguration, so wird diese erst aktiv, wenn der Server neu gestartet worden ist bzw. die Konfigurationsdateien neu eingelesen worden sind. Erst nach Einlesen der Konfigurationsdateien werden diese Änderungen in der Ausgabe von <i>mod_info</i> sichtbar. <i>--enable-info</i> schaltet dieses Modul ein, <i>--disable-info</i> schaltet es ab.
--enable-suEXEC --disable-suEXEC	Ermöglicht es CGI-Skripten unter der Benutzerkennung eines beliebigen anderen Benutzers oder Gruppe zu laufen. Wird aktiviert mit <i>--enable-suEXEC</i> , <i>--disable-suEXEC</i> lässt das Modul weg.
--enable-cgid --disable-cgid	Unter Unix/Linux ist es mit <i>mod_cgid</i> möglich, CGI-Skripte mit einem externen CGI-Daemon auszuführen. Es verhält sich exakt wie <i>mod_cgi</i> , bietet eine zusätzliche Konfigurationsoption ( <i>ScriptSock</i> ) und wird automatisch gewählt, wenn während des Kompilierungs- und Konfigurationsprozesses ein thread-basierendes Multi Processing Module (MPM) wie <i>mpm_leader</i> o.Ä. gewählt wird.

**Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung**  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-cgi --disable-cgi	Dieses Basismodul ermöglicht die Ausführung von CGI-Skripten. Es wird aktiviert mit <i>--enable-cgi</i> , falls Sie inzwischen alternative Programmiersprachen zur Entwicklung von dynamischen Webseiten benutzen (z. B. PHP, Java), können Sie es mit <i>--disable-cgi</i> abschalten.
--enable-dav-fs --disable-dav-fs	<i>Mod_dav_fs</i> ist eine Schnittstelle für <i>mod_dav</i> , die es <i>mod_dav</i> erlaubt, Inhalte des Dateisystems zu verwalten. Es wird eingeschaltet mit <i>--enable-dav-fs</i> , abgeschaltet mit <i>--disable-dav-fs</i> und sollte immer zusammen mit <i>mod_dav</i> Verwendung finden.
--enable-negotiation --disable-negotiation	Mit diesem nützlichen Modul lassen sich lokale Gegebenheiten auf der Clientseite (z. B. Sprach-einstellungen) auswerten und zur Veröffentlichung von benutzeroptimierten Inhalten (z. B. Startseite in Deutsch) verwenden. Es wird aktiviert mit <i>--enable-negotiation</i> , falls Sie es deaktivieren möchten, können Sie dies mit <i>--disable-negotiation</i> tun. Benutzer- und sprachoptimierte Inhalte lassen sich auch mit dynamischen Skripten (z. B. in Perl oder PHP) entwickeln.
--enable-vhost-alias --disable-vhost-alias	Dynamisch konfigurierbares Massenhosing anhand von vorgefertigten Strukturen des lokalen Dateisystems wird durch dieses Modul bereitgestellt. Damit entfällt die massenweise Definition von VirtualHosts in der Konfigurationsdatei <i>httpd.conf</i> des Apaches und neue virtuelle Hosts lassen sich ohne Neustart des Apache aktivieren. Dieses Modul ist wohl nur für Massenhoster praktisch und wird mit <i>--enable-vhost-alias</i> aktiviert, <i>--disable-vhost-alias</i> schaltet es ab.
--enable-dir --disable-dir	<i>Mod_dir</i> stellt die Option <i>DirectoryIndex</i> zur Verfügung, mit der die Startdatei eines Webangebotes definiert wird. Außerdem leitet es unvollständige Clientanfragen an die richtige Adresse weiter. Dieses wichtige Modul wird mit <i>--enable-dir</i> aktiviert, <i>--disable-dir</i> schaltet es ab.

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-imap --disable-imap	Dieses veraltete Modul stellt serverseitige Imagemaps bereit. Es wird mit <i>--disable-imap</i> deaktiviert, <i>--enable-imap</i> aktiviert es. Dieses Modul findet praktisch keine Verwendung mehr.
--enable-actions --disable-actions	<i>Mod_actions</i> kann CGI-Skripten bestimmte MIME-Typen zuordnen und bei Zugriff auf Dateien dieses Typs diese automatisch ausführen. Mit <i>--enable-actions</i> wird dieses Modul aktiviert, <i>--disable-actions</i> deaktiviert es.
--enable-speling --disable-speling	<i>Mod_speling</i> versucht, durch menschliche Benutzer gemachte Fehler wie falsch eingegebene Internetadressen, nicht beachtete Groß- und Kleinschreibung etc. zu korrigieren und dennoch die richtige, ursprünglich gewünschte Seite zu finden. Das Modul wird mit <i>--enable-speling</i> eingeschaltet, <i>--disable-speling</i> schaltet es aus.
--enable-userdir --disable-userdir	Mit dem Modul <i>mod_userdir</i> können Benutzer unter einer vorher festgelegten Adresse (z. B. <i>www.domain.de/~benutzername</i> ) persönliche und individuelle Inhalte im Internet präsentieren. Benutzt werden kann dieses Modul mit <i>--enable-userdir</i> , <i>--disable-userdir</i> schaltet die Verwendung ab.
--enable-alias --disable-alias	Zur Umleitung von internen Pfaden auf andere Verzeichnisse (sog. Aliase) oder komplette Weiterleitungen an eine externe Webseite (sog. Redirect) kann <i>mod_alias</i> benutzt werden. Es wird mit <i>--enable-alias</i> verwendet, <i>--disable-alias</i> schaltet es ab.
--enable-rewrite --disable-rewrite	Unter Verwendung eines vorher definierten und auf regulären Ausdrücken basierenden Regelsatzes, können mit <i>mod_rewrite</i> eingehende Anfragen auf andere Ziele umgeleitet werden. Es wurde von dem Deutschen Ralf S. Engelschall ( <a href="http://www.engelschall.com">http://www.engelschall.com</a> ) entwickelt und wird mit <i>--enable-rewrite</i> eingeschaltet, <i>--disable-rewrite</i> schaltet die Benutzung von <i>mod_rewrite</i> ab.

Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--enable-so --disable-so	Diese extrem wichtige Option bestimmt die Verwendung von <i>Dynamic Shared Objects</i> und wird mit <i>--enable-so</i> eingeschaltet. <i>Dynamic Shared Objects</i> erlauben es dem Apache, nachträglich Module zu installieren und zu verwenden, ohne den Apache selbst neu zu kompilieren. Dadurch ist etwa eine Aktualisierung einer Zusatzsoftware wie PHP ohne Modifizierung und Neukompilierung des Apache möglich. Wichtig: Die als <i>Dynamic Shared Objects</i> zu übersetzenden Module müssen mit der Zusatzoption <i>--enable-mods-shared=Modulliste</i> angegeben werden, wobei auch die Verwendung von <i>--enable-mods-shared=min</i> bzw. <i>--enable-mods-shared=max</i> möglich ist! Falls Sie dennoch auf dieses Modul verzichten möchten, so können Sie dies durch die Option <i>--disable-so</i> machen.
--with-PACKAGE=yes	Durch Angabe von <i>--with-package=yes</i> wird die Benutzung von bestimmten Packages (gemeint sind hier Modulnamen) explizit angegeben. Beispielsweise aktiviert die Option <i>--with-example=yes</i> die Nutzung von <i>mod_example</i> .
--without-PACKAGE	Genauso wie Module explizit aktiviert werden können, können sie auch explizit deaktiviert werden. Die Option <i>--without-example</i> deaktiviert die Verwendung von <i>mod_example</i> und entspricht ebenfalls der Option <i>--with-example=no</i> .
--with-port=Portnummer	Gibt die Portnummer an, unter der der Apache später erreichbar sein soll. Der Standardport ist Port 80, die Angabe eines anderen Ports bedarf bei der Aufrufung einer Internetseite durch einen Client auch die Angabe des neuen Ports (z.B. <i>http://www.domain.tld:8000</i> ).
--with-z=Verzeichnis	Diese Option erlaubt die Verwendung einer bestimmten zlib-Datenkompressionsbibliothek. Das Installationsverzeichnis (z.B. <i>/usr/local</i> ) der alternativen zlib-Bibliothek muss dabei beim Aufruf übergeben werden mit <i>--with-z=/usr/local</i> .

*Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)*

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--with-ssl=Verzeichnis	Durch Angabe eines Verzeichnisses kann mit der Option ein SSL/TLS-Toolkit verwendet werden. Wurde ein solches Toolkit beispielsweise nach <code>/usr/local/ssl</code> installiert, so kann dieses durch die Option <code>--with-ssl=/usr/local/ssl</code> benutzt werden.
--with-mpm=MPM	Durch die Option <code>--with-mpm=MPM</code> wird der Apache angewiesen, ein bestimmtes <i>Multi-processing Modul (MPM)</i> , d.h. ein bestimmtes Laufzeitverhalten, zu benutzen. Unter Unix/Linux stehen zurzeit bereits die Module <code>prefork</code> , <code>perchild</code> , <code>worker</code> , <code>leader</code> und <code>threadpool</code> zur Verfügung. Unter Windows existiert bisher nur das MPM <code>winnt</code> , für OS/2 existiert <code>mpmt_os2</code> und Netware hat das MPM <code>netware</code> .
--with-module=Modultyp:Datei	Mit dieser Option können Module und deren Moduldateien explizit aktiviert werden, wenn diese vorhanden sind.
--with-program-name=Name	Hier kann ein alternativer Name für die eigentliche Programmdatei des Apache (z. B. <code>apache2</code> anstatt <code>httpd</code> ) gewählt werden. Die Konfigurationsdatei des Servers wird ebenfalls diesem Namen angepasst.
--with-suEXEC-bin=Pfad	Definiert den Speicherort von SuEXEC nach der Installation.
--with-suEXEC-caller=Benutzer	Bestimmt den Benutzer, der später das SuEXEC-Programm aufrufen darf, d.h. der Benutzer unter dessen Kennung der Apache läuft.
--with-suEXEC-userdir=Verzeichnis	Gibt das Verzeichnis an, in dem sich die Webseiten der Benutzer befinden.
--with-suEXEC-docroot=Verzeichnis	Entspricht der Einstellung der Anweisung <code>DocumentRoot</code> (vgl. <i>DocumentRoot</i> -Anweisung).
--with-suEXEC-uidmin=Benutzerid	Bestimmt die kleinste Benutzerkennung (numerisch), der es erlaubt ist, das Programm SuEXEC zu benutzen. Aus Sicherheitsgründen sollte diese Benutzerkennung immer größer sein, als die Kennung mit der der Apache läuft.
--with-suEXEC-gidmin=Gruppenid	Bestimmt die kleinste Gruppenkennung (numerisch), der es erlaubt ist, das Programm SuEXEC zu benutzen.

*Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)*

Option (Abkürzungen und Synonyme sind durch Komma getrennt)	Bedeutung
--with-suEXEC-logfile=Datei	Gibt den Speicherort einer Logdatei für SuEXEC an.
--with-suEXEC-safepath=Verzeichnis	Dieser Parameter bestimmt einen Systempfad für die durch SuEXEC aufgerufenen CGI-Skripte.
--with-suEXEC-umask=Umask	Definiert die umask für den SuEXEC-Prozess

*Tabelle 3.1 Konfigurationsoptionen und deren Bedeutung  
(abgeleitet von »./configure --help«) (Forts.)*

## 3.3 Modulübersicht

Bevor wir mit der Übersetzung und Installation des Apache beginnen, möchte ich in diesem Kapitel eine kleine Einführung in die modulare Struktur des Apache geben und die Funktionen der unterschiedlichen Module erklären. Ich halte dies durchaus für sinnvoll, denn bei der Kompilierung und Installation sind gewisse Grundlagen und Kenntnisse einfach nötig. Außerdem bekommt man hierdurch einen guten Einblick in die grundlegende Funktionsweise unseres Häuptlings.

Grundsätzlich besteht der Apache zunächst aus einem unabdingbaren Kernmodul (Quellcode: core.c), welches kompiliert etwa 1,26 MB groß ist und die grundlegendsten Funktionen definiert. Dieses Modul stellt bestimmte Konfigurationsoptionen (so genannte Direktive) bereit, die immer zur Verfügung stehen. Dazu kommen eine Reihe von Standardmodulen, die Teil der offiziellen Apache-Distribution sind und auch durch die Apache Software Foundation gepflegt werden. Den dritten und letzten Part stellen die Module der Drittanbieter, die teilweise unter <http://modules.apache.org> gelistet werden.

### 3.3.1 Standardmodule

Folgende Module gelten als Standardmodule:

Modulname	Funktion
mod_access	Mit diesem Modul kann der Zugriff auf Inhalte des Webservers auf Basis von Hostnamen und IP-Adressen kontrolliert werden. Dies ist beispielsweise sinnvoll, wenn man sensible Daten (Logfileauswertungen o. Ä.) schützen und nur ganz bestimmten Hosts den Zugriff gewähren will.

*Tabelle 3.2 Standardmodule des Apache*

Modulname	Funktion
mod_actions	Hiermit lassen sich CGI-Skripte bestimmten MIME-Typen zuordnen und bei Zugriff auf Dateien mit diesem Typ automatisch ausführen.
mod_alias	Mit mod_alias lassen sich URL-Pfade intern auf ein anderes Verzeichnis (Alias) oder komplett an eine andere URL umleiten (Redirect).
mod_asis	Dieses Modul stellt den Handler <i>send-as-is</i> bereit, der Dokumente so ausliefern kann »wie sie sind«, d.h. ohne Hinzufügung der normalen HTTP-Header. Damit lassen sich beliebige Inhalte und benutzerdefinierte HTTP-Header an den Client senden und somit beispielsweise Umleitungen (so genannte Redirects) realisieren.
mod_auth	Mod_auth stellt die drei Directiven <i>AuthGroupFile</i> , <i>AuthUserFile</i> und <i>AuthAuthoritative</i> zur Verfügung, die zur Konfiguration von passwortgeschützten Verzeichnissen und Inhalten benötigt werden. Das Modul arbeitet auf Basis von Textdateien und ist die einfachste Art eines Authentifizierungsmoduls.
mod_autoindex	Existiert in einem Verzeichnis keine Indexdatei, erzeugt dieses Modul einen Verzeichnisindex, der teilweise an die eigenen Wünsche angepasst werden kann.
mod_cgi	Damit der Apache CGI-Skripte (CGI – Common Gateway Interface) überhaupt ausführen kann, wird dieses Modul benötigt.
mod_cgid	Dieses Modul ist mit mod_cgi fast identisch, allerdings benutzt es einen externen Daemon zur Ausführung von CGI-Skripten und wird automatisch benutzt, wenn bei der Kompilierung des Apache ein multi-threaded MPM gewählt worden ist. Auf die verschiedenen Laufzeitmodelle werde ich später noch eingehen.
mod_dir	Das Module mod_dir stellt nur eine Konfigurationsoption namens <i>DirectoryIndex</i> zur Verfügung und diese legt fest, welche Datei dem Client gesendet wird, wenn dieser ein Verzeichnis aufruft. Dies ist üblicherweise eine Datei wie <i>index.html</i> oder <i>index.php</i> . Ruft ein Client beispielsweise die URL <i>http://www.beispiel.de/infos/</i> auf, wird automatisch versucht die als Directory-Index angegebene Datei zu finden und falls vorhanden an den Client zu senden. Außerdem vervollständigt dieses Modul Clientanfragen, wenn diese auf Verzeichnisse zugreifen, aber den abschließenden Slash (»/«) vergessen haben, und leitet sie auf die korrekte URL um.
mod_env	Dieses Modul sorgt dafür, dass Umgebungsvariablen aus der Shell an den Apache bzw. an ein auf dem Server laufendes Skript übergeben werden können. Ebenso können beliebige Variablen in der Serverkonfiguration gesetzt und ausgelesen werden.

Tabelle 3.2 Standardmodule des Apache (Forts.)

---

Modulname	Funktion
mod_imap	mod_imap erweitert den Indianer um die Möglichkeit der Nutzung von serverseitigen Image-Maps (so genannte Clickable Images). Unterstützt ein Clientbrowser keine Bilder (z.B. Textbrowser wie lynx), dann wird automatisch ein Textmenü erzeugt. Dieses Modul ist meiner Meinung nach überflüssig, da Imagemaps mit diversen Technologien heutzutage auf dem Client realisiert werden. Dies wird direkt in das HTML-Dokument eingebettet und teilweise sogar mit Flash oder ähnlich gelagerten Designmöglichkeiten umgesetzt.
mod_include	Durch dieses Module werden die sog. <i>Server-Side Includes</i> aktiviert, d.h. Dokumente können auf einschließende Verweise auf externe Datenquellen (z.B. Shellskripte, CGI-Skripte o. Ä.) untersucht werden. Falls solche externen Verweise vorhanden sind, werden diese ausgeführt und das Ergebnis wird an der Stelle des Verweises eingefügt. Nachdem alle SSI-Verweise ersetzt worden sind, wird das Dokument an den Client gesendet.
mod_log_config	Das Modul log_config hat das Modul log_common ersetzt, welches in früheren Versionen des Apache Verwendung fand. Es erzeugt u.a. Zugriffsstatistiken über die Art und Häufigkeit der aufgerufenen Webseiten. Diese Statistiken lassen sich mit diversen Programme (z.B. Webalizer, Webtrends etc.) auswerten und grafisch darstellen. Bei Bedarf lässt sich das Aussehen und der Inhalt der Logfiles an die eigenen Wünsche anpassen.
mod_mime	Basierend auf der Endung einer Datei (z.B. <i>.html</i> ) werden so genannte MIME-Header (Multipurpose Internet Mail Extensions) erzeugt, die an den Client gesendet werden. Ohne dieses Kennzeichen könnten die Clients mit den abgerufenen Daten nichts anfangen. Deshalb gehört dieses Modul zu den absolut notwendigen und sollte auf jeden Fall statisch oder als Dynamic Shared Object in den Apache integriert werden.
mod_negotiation	Anhand der HTTP-Header, die ein Client sendet, lassen sich mit diesem Modul transparent, d.h. für den Client nicht direkt ersichtlich, Inhalte bereitstellen, die an persönliche Gegebenheiten des Nutzers angepasst sind. Ein Beispiel dafür ist die bei der Installation des Apache standardmäßig vorhandene Startseite, die je nach Spracheinstellung des Clients eine entsprechende Variante (etwa in Deutsch) darstellt.
mod_setenvif	Mit diesem Modul lassen sich anhand der bei einer Clientanfrage zur Verfügung stehenden Informationen (z.B. IP-Adresse, Browserversion etc.) Umgebungsvariablen setzen. Es wird auch u.a. dazu verwendet, Funktionalitäten des Apache unter bestimmten Umständen auszuschalten bzw. zu umgehen.

---

Tabelle 3.2 Standardmodule des Apache (Forts.)

Modulname	Funktion
mod_so	Das mod_so integriert die Möglichkeit des dynamischen Ladens und Aktivierens von externen Modulen für den Apache, ohne dass dieser neu kompiliert werden muss. Diese externen Objekte nennt man <i>Dynamic Shared Objects</i> (DSO). Auf deren Besonderheiten werde ich im Laufe dieses Buches noch genauer eingehen.
mod_status	Wie der Name schon unschwer erkennen lässt, erzeugt dieses Modul einen Statusbericht der unter einer vorher definierten Adresse erreichbar ist und u.a. Auskunft gibt über die aktuelle CPU-Auslastung, die Anzahl der in Bearbeitung befindlichen Anfragen usw. Ein Beispiel für den Einsatz von mod_status finden Sie unter <a href="http://xml.apache.org/server-status?refresh=10">http://xml.apache.org/server-status?refresh=10</a> .
mod_userdir	Lokale User können mit diesem Modul Ihre privaten Verzeichnisse unter einer gewissen Adresse ( <a href="http://www.domain.tld/~username/">http://www.domain.tld/~username/</a> ) freigeben und ins Internet stellen.

Tabelle 3.2 Standardmodule des Apache (Forts.)

### 3.3.2 Module von Drittanbietern

Bekannte Module von Drittanbietern sind außerdem:

Modulname	Funktion
mod_auth_anon	mod_auth_anon ist in etwa vergleichbar mit <i>Anonymous FTP</i> , d.h. ein Benutzer kann unter Angabe des Benutzernamens <i>anonymous</i> und seiner E-Mail-Adresse authentifiziert werden und bestimmte (geschützte) Bereiche des Webservers erreichen. Die Zugriffe können in einem separaten Logfile gespeichert werden.
mod_auth_dbm	Authentifizierung via datenbankähnlicher DBM-Dateien, die mit dem Programm <i>dbmmanage</i> erstellt und aktualisiert werden können.
mod_auth_digest	Dieses Modul implementiert die <i>MD5-Digest Authentication</i> , die fester Bestandteil der HTTP/1.1-Spezifikation ist. Der Vorteil gegenüber der veralteten <i>Basic Authentication</i> ist, dass die Passwörter der Nutzer nicht mehr im Klartext übertragen werden. Es gilt noch als experimentell, löst aber das Modul mod_auth_digest und mod_digest ab. Alle Browser sollten diesen Standard inzwischen unterstützen.
mod_auth_ldap	Ab dem Apache 2.0.41 besteht die Möglichkeit, den Verzeichnisdienst LDAP zur einfachen Authentifikation eines Benutzers zu verwenden.

Tabelle 3.3 Bekannte Module von Drittanbietern

Modulname	Funktion
mod_cache	Ab Apache 1.3.x wurden die Cachingfunktionen aus dem Modul mod_proxy herausgenommen und in ein separates Modul (mod_cache) integriert.
mod_cern_meta	Falls Sie vom CERN HTTPD auf den Apache umgestiegen sind, können Sie dieses Modul dazu benutzen, Ihre CERN-Meta-Files weiter zu verwenden. Meta-Files sind Dateien, die ähnlich dem Modul mod_asis zusätzliche HTTP-Header enthalten, die an den Client gesendet werden können. Mod_header ist auf jeden Fall diesem Modul vorzuziehen und wenn Sie (wie ich) niemals einen anderen Webserver außer dem Apache benutzt haben, brauchen Sie diese Funktionalität nicht.
mod_charset_lite	Hiermit kann der Administrator Zeichensätze definieren, in die die Inhalte übersetzt werden sollen, bevor diese an den Client gesendet werden. Es befindet sich derzeit noch in der Entwicklung und gilt als experimentell.
mod_dav	<i>Distributed authoring and versioning</i> ist die Abkürzung dieses Moduls, zu Deutsch etwa <i>verteile und versionskontrollierte Verwaltung und Entwicklung</i> . DAV gilt inzwischen allgemein als Standard und wird von den meisten Authoringprogrammen wie Dreamweaver, Frontpage etc. unterstützt. Es bietet praktisch die Möglichkeit auf Verzeichnisse eines entfernten Webservers zuzugreifen, als sei dieser eine lokale Ressource. Ohne die Benutzung entsprechender Sicherheitsmechanismen kann man den Einsatz dieses Moduls nicht verantworten. Die Homepage dieses Moduls lautet <a href="http://www.webdav.org/mod_dav/">http://www.webdav.org/mod_dav/</a> .
mod_deflate	Mit dem Modul mod_deflate können Sie Informationen komprimieren, bevor diese an einen Client gesendet werden. Dabei ist das Modul als Ausgabefilter konzipiert und sorgt je nach Typ der komprimierten Daten zum Teil für große Komprimierungsraten (>30-50%). Für den Apache 1.3.x existieren unter dem Namen mod_gzip bzw. mod_gz ähnliche Module.
mod_echo	Dieses Modul veranschaulicht die im Apache 2.x enthaltene Multiprotokollunterstützung und implementiert einen einfachen Echo-Server.
mod_example	Dieses Modul dient ausschließlich Demonstrationszwecken und gibt interessierten Entwicklern Einblick in die Funktionsweise der Apache API (Application Programming Interface, Programmierschnittstelle). Es dient lediglich der Anschauung und sollte nicht im Produktionsbetrieb eingesetzt werden.
mod_expires	Durch dieses Modul kann man einem Dokument den HTTP-Header <i>Expires</i> hinzufügen, der die Lebensdauer eines Dokumentes definiert. Dies ist besonders dann sinnvoll, wenn man Dokumente hat, deren Inhalte sich häufig ändern und von denen

Tabelle 3.3 Bekannte Module von Drittanbietern (Forts.)

Modulname	Funktion
	immer die jeweils aktuellste Version an den Client gesendet werden soll. So kann man Proxyserver und lokale Zwischenspeicher auf dem jeweiligen Client umgehen.
mod_ext_filter	Durch das Modul <code>mod_ext_filter</code> können externe Programme, die Daten von der Standardeingabe lesen, diese verarbeiten und in die Standardausgabe schreiben, als Filter benutzt werden, bevor Daten an den Client geschickt werden. Somit lassen sich Daten beispielsweise sortieren oder anderweitig ver- oder bearbeiten, bevor diese an den Client geschickt werden.
mod_file_cache	Im Apache 1.3.x gab es das Modul <code>mod_mmap_static</code> , welches inzwischen durch dieses Modul ersetzt worden ist. Beim Start des Apache können dadurch Dateien direkt im Speicher abgelegt werden. Diesen Vorgang nennt man <i>Memory Mapping</i> und er erhöht die Zugriffszeit auf diese Dateien erheblich, da die Daten direkt aus dem Speicher kommen. Zusätzlich kann das Modul Dateien beim Start des Servers öffnen und deren geöffneten Dateihandler speichern. Allerdings werden Veränderungen an diesen Dateien nicht sofort bemerkt und erst nach einem Neustart des Servers angezeigt. Außerdem kann die Technik nicht auf CGI-Skripte u.Ä. angewendet werden, da das nur mit »normalen« Dateien funktioniert, die durch das Kernmodul des Apache bereitgestellt werden können. <code>Mod_file_cache</code> gilt bisher als experimentell und ist mit Vorsicht zu gebrauchen.
mod_headers	<code>Mod_headers</code> kann HTTP-Header beliebig modifizieren, hinzufügen, ersetzen oder auch löschen.
mod_info	Dieses Modul erzeugt eine umfassende Übersicht über die Konfiguration des Servers, die unter einer frei wählbaren URL erreichbar sein kann.
mod_isapi	Die Möglichkeit der Entwicklung und Nutzung von <i>Internet Server Application Programming Interface</i> (ISAPI) war bisher nur unter Windows mit dem Internet Information Server (IIS) möglich. Durch <code>mod_isapi</code> kann man auch mit dem Apache in den Genuss dieser Erweiterung kommen, allerdings nur unter Microsoft Windows.
mod_jserv	<code>Mod_jserv</code> ist eine vollständige Java Servlet Engine, die inzwischen jedoch nicht mehr eigenständig weiterentwickelt wird und mit dem Jakarta-Projekt ( <a href="http://jakarta.apache.org">http://jakarta.apache.org</a> ) verschmolzen ist. Nun wird Tomcat zur Ausführung von Java Server Pages und Java Servlets benutzt.
mod_ldap	Um die Geschwindigkeit von Webseiten zu verbessern, die auf Verbindungen zu LDAP-Verzeichnisdiensten (LDAP=Lightweight Directory Access Protocol) beruhen, wurde dieses Modul entwickelt. Es sorgt ab dem Apache 2.0.41 für eine gesteigerte

Tabelle 3.3 Bekannte Module von Drittanbietern (Forts.)

Modulname	Funktion
	Performance u. a. durch Zwischenspeicherung von Ergebnissen und Bereitstellung von so genanntem Connection Pooling. Diese Connection Pools sind persistenten Datenbankverbindungen ähnlich und erlauben den Aufruf mehrerer Abfragen über eine offene Verbindung zum LDAP-Server.
mod_mime_magic	Basierend auf dem Unix-Befehl <i>file</i> untersucht dieses Modul Dateien nicht anhand ihrer Endungen, sondern anhand ihrer Inhalte und versucht so, den richtigen MIME-Typen zu finden und diesen an den Client zu senden. Es ist besonders hilfreich, wenn Dateien nicht immer auch eine bestimmte oder bekannte Endung haben.
mod_perl	Mit <i>mod_perl</i> ist es möglich, die sehr mächtige Programmiersprache Perl zur Entwicklung eigener Apache-Module und Skripte zu benutzen. Bis zur Entwicklung von <i>mod_perl</i> war die Entwicklung neuer Module für den Apache nur in C möglich. Dabei wird der Perl-Interpreter speicherresident geladen. Eine sehr nützliche Funktion ist außerdem das so genannte Code-Caching, welches Skripte kompiliert, ausführt und danach im Cache speichert, bis der Server beendet wird.
mod_php	Ursprünglich von Rasmus Lerdorf entwickelte und unter dem Namen <i>Personal Homepage Tools</i> bekannt gewordene Erweiterung, die in C geschrieben ist. Inzwischen wird diese Erweiterung als <i>PHP: Hypertext Preprocessor</i> bezeichnet und hat in den letzten Jahren eine sehr weite Verbreitung gefunden. PHP wird, ebenso wie Perl, serverseitig ausgeführt und ist laut einer Umfrage von <a href="http://www.securityspace.com">http://www.securityspace.com</a> mit einem Nutzungsgrad von etwa 53% die beliebteste Apache-Erweiterung. Zum überragenden Erfolg von PHP haben sicherlich die Schnittstellen zu diversen Datenbanken (z. B. Oracle, MySQL, Interbase, Sybase etc.), die dynamische Erzeugung von Grafiken, die Netzwerkfunktionen und eine gute (inzwischen sogar mehrsprachige) Dokumentation beigetragen. Die Homepage dieser Erweiterung lautet <a href="http://www.php.net">http://www.php.net</a> , ein deutscher Mirror ist unter der Adresse <a href="http://www.php3.de">http://www.php3.de</a> erreichbar. Im Moment ist die Version 4.3.4 aktuell, die Version 5 steht jedoch kurz vor der Veröffentlichung.
mod_proxy	Dieses Modul erweitert den Apache um Proxyfunktionalitäten. Unterstützt wird die HTTP/0.9-, HTTP/1.0- und HTTP/1.1-Spezifikation sowie FTP und SSL. Ich persönlich bevorzuge als Proxyserver die Software Squid ( <a href="http://www.squid-cache.org">http://www.squid-cache.org</a> ), aber in Verbindung mit <i>mod_rewrite</i> kann dieses Modul unter Umständen durchaus Sinn machen.
mod_rewrite	Das Modul ist 1996 durch Ralf S. Engelschall entwickelt worden und bietet die Möglichkeit, ähnlich wie <i>mod_alias</i> , interne Aliase und externe Redirects zu realisieren, wobei diese im Gegensatz

Tabelle 3.3 Bekannte Module von Drittanbietern (Forts.)

Modulname	Funktion
	zu <i>mod_alias</i> auch auf Basis von regulären Ausdrücken ( <i>POSIX regular expressions</i> ) und externen Programmen möglich sind.
mod_speling	Mod_speling korrigiert Tippfehler der Benutzer und versucht dennoch die gewünschte Datei an den Client zu senden. Der Name <i>mod_speling</i> ist von den Autoren bewusst falsch geschrieben worden und deutet wohl auf die Funktion dieses Moduls hin.
mod_ssl	Basierend auf Apache-SSL entwickelte im April 1998 der Deutsche Ralf S. Engelschall mit mod_ssl sozusagen die Schnittstelle zwischen Apache und OpenSSL, einer frei verfügbaren SSL-Bibliothek zur sicheren Verschlüsselung von Daten. Da der Apache den Exportbestimmungen der USA unterliegt, ist die SSL-Unterstützung nicht direkt in den Apache eingebunden, sie muss durch dieses Zusatzmodul aktiviert werden.
mod_suEXEC	Normalerweise werden CGI- (Common Gateway Interface) und SSI-Skripte (Server Side Includes) unter dem Benutzer ausgeführt, unter dem auch der Apache läuft. Mit mod_suEXEC (su = switch user) können diese ab Apache 1.2 durch einen beliebigen anderen Benutzer oder eine andere Gruppe ausgeführt werden.
mod_unique_id	Wie der Name schon angibt, erzeugt dieses Modul eine Umgebungsvariable mit einer einzigartigen (englisch: <i>unique</i> ) Zahl (auch ID genannt), die für jede Clientanfrage neu und einzigartig erzeugt wird.
mod_usertrack	Ehemals mod_cookies genannt, bietet dieses Modul die Möglichkeit, mit Hilfe so genannter Cookies die Aktionen eines Benutzers zu verfolgen und aufzuzeichnen. Cookies sind kleine Textdateien, die auf dem Client gespeichert werden und durch den Server, der diese gesetzt hat, wieder ausgelesen werden können. Das Verfolgen und Aufzeichnen der Benutzeraktionen (auch <i>Usertracking</i> genannt) basiert auf einem einfachen Grundprinzip, ist aber in der Regel aufgrund der Tatsache, dass nicht jeder Browser Cookies unterstützt und diese auch durch den Benutzer abgelehnt werden können, praktisch nicht gezielt realisierbar.
mod_vhost_alias	Ein sehr hilfreiches Modul für die Bereitstellung von virtuellem Hosting. Es bezieht die Informationen für eine neue Domain nicht aus einem <i>&lt;VirtualHost&gt;</i> -Eintrag in der <i>httpd.conf</i> , sondern direkt aus dem Dateisystem und somit findet eigentlich ein Aliasing von Verzeichnisnamen auf fertige Domainnamen statt. Dadurch können ohne Neustart des Apache neue Domains angelegt werden. Auf die verschiedenen Möglichkeiten des virtuellen Hosting wird in späteren Kapiteln noch ausführlich eingegangen.

Tabelle 3.3 Bekannte Module von Drittanbietern (Forts.)

Teilweise werden Module von Drittanbietern unter <http://modules.apache.org> aufgelistet. Eine Suche ohne die Eingabe eines Suchbegriffs zeigt dort eine Übersicht aller registrierten Module der Drittanbieter. Falls man auf der Suche nach einem speziellen Module ist, hilft oft auch eine Suchanfrage bei <http://www.google.de>.

Bei der Vielzahl von Modulen die Übersicht zu behalten und nur die für die jeweiligen Bedürfnisse nötigen Module zu finden, ist sicherlich eine aufwendige Arbeit. Deshalb habe ich in der nachfolgenden Tabelle geeignete Modulkombinationen aufgelistet und diese verschiedenen Szenarien zugeordnet.

### 3.3.3 Empfohlene Module

Diese Module bilden die Grundlage für die nachfolgenden Fallbeispiele und sind so essenziell, dass sie immer in den Apache integriert werden sollten.

Name des Moduls	Grund für den Einsatz
mod_access	Sensible Inhalte einer Website sollen nur von bestimmten Hosts erreichbar sein und können durch dieses Modul geschützt werden.
mod_auth	Der Zugang zu bestimmten Verzeichnissen kann durch eine Authentifikation am Server mit Hilfe eines gültigen Benutzernamens und Passwortes gewährt werden.
mod_dir	Der Name der Startdatei wird durch die Konfigurationsoption <i>DirectoryIndex</i> bestimmt, den dieses Modul zur Verfügung stellt. Außerdem hängt dieses Modul automatisch einen Slash (»/«) an die vom Benutzer angegebene URL an, wenn dieser fehlt und der Benutzer eigentlich ein Verzeichnis aufrufen wollte.
mod_log_config	Um eine Statistik der aufgerufenen Webseiten zu bekommen, ist dieses Modul unbedingt notwendig.
mod_mime	Damit die Clients überhaupt wissen, welche Art von Inhalten eine Website zur Verfügung stellt, definiert dieses Modul so genannte MIME-Typen.

Tabelle 3.4 Grundkonfiguration der Module

#### Fallbeispiel 1

Minimale Konfiguration eines Webservers, der ausschließlich statische Informationen in Form von HTML-Dateien zur Verfügung stellt. Jeder Mitarbeiter soll ein individuelles Verzeichnis für seine Daten bekommen, die er im Web veröffentlichen will. Eventuell sollen die statischen Inhalte später durch dynamisch generierte Webseiten ersetzt werden.

Name des Moduls	Grund für den Einsatz
mod_autoindex	Benutzer, denen die Möglichkeit gegeben wird, unter einer bestimmten Adresse persönliche Inhalte im Web zu veröffentlichen, vergessen eventuell eine Startdatei wie <i>index.html</i> o.Ä. zu definieren. Das Modul erzeugt automatisch einen Index eines Verzeichnisses und zeigt diesen an.
mod_setenvif	Eventuell müssen Funktionalitäten des Apache aufgrund lokaler Gegebenheiten der Clients (z.B. stark veraltete Browserversion) abgeschaltet werden.
mod_so	Falls später die statischen Webseiten durch dynamisch erzeugte ersetzt werden sollen, so werden zusätzliche Module (z.B. PHP) benötigt, die eigentlich eine Neuinstallation des Apache erzwingen. Mit diesem Modul ist die nachträgliche Installation von Modulen ohne Neukompilierung des Apache möglich. Man spricht hier von so genannten <i>Dynamic Shared Objects</i> (DSO).
mod_userdir	Jeder lokale Benutzer soll unter einer festgelegten Adresse (z.B. <i>http://www.domain.tld/~benutzername</i> ) individuelle Informationen veröffentlichen können.

*Tabelle 3.5 Modulkonfiguration eines sehr einfachen Webservers, basierend auf der oben genannten Grundkonfiguration*

## Fallbeispiel 2

Konfiguration eines kompletten Webservers, der nur eine Website beheimatet. Die Inhalte werden sowohl statisch, als auch dynamisch generiert. Eingesetzt wird der Server als Intranet bzw. Internetserver.

Name des Moduls	Grund für den Einsatz
mod_actions	CGI-Skripte können einem bestimmten MIME-Typen zugewiesen und bei Zugriff auf Dateien diesen Typs automatisch ausgeführt werden.
mod_alias	Umleitung von internen Pfaden auf ein anderes Verzeichnis (sog. Alias) oder komplette Weiterleitung an eine externe Webseite (sog. Redirect).
mod_asis	Eventuell ist es nötig, eigene Header an den Client zu senden, ohne dass die normalen HTTP-Header hinzugefügt werden. Auch Redirects können damit realisiert werden.
mod_cache	Zur Zwischenspeicherung von häufig angeforderten Daten ist dieses Modul sicherlich nützlich.

*Tabelle 3.6 Modulkonfiguration eines Intranet/Internet Servers, basierend auf der oben genannten Grundkonfiguration sowie der Konfiguration aus dem ersten Szenario*

Name des Moduls	Grund für den Einsatz
mod_cgi	Auf vielen Webservern kommen so genannte <i>Common Gateway Interface</i> (CGI)-Skripte zum Einsatz, die in verschiedenen Programmiersprachen geschrieben werden können. Damit der Apache diese überhaupt ausführen kann, wird dieses Modul benötigt.
mod_env	Teilweise wird der Zugriff auf Umgebungsvariablen aus der Shell im Apache benötigt. Dieses Modul stellt die Verbindung zwischen Shell und Apache her und erlaubt die Definition eigener Variablen in der Serverkonfiguration.
mod_expires	Für Inhalte der Website kann es eventuell nötig sein, die Dauer der Gültigkeit anhand eines zusätzlichen HTTP-Headers zu beschränken.
mod_headers	Unter Umständen kann es notwendig sein, die HTTP-Header beliebig zu modifizieren, hinzuzufügen, zu ersetzen oder auch zu löschen.
mod_include	Falls <i>Server Side Includes</i> (SSI) benutzt werden, ist dieses Modul auf jeden Fall nötig. SSI-Skripte sind meiner Meinung nach eigentlich veraltet und daher ist die Benutzung dieses Moduls optional.
mod_mime_magic	Dieses Modul versucht anhand des Inhaltes einer Datei den Dateitypen zu erkennen und ihr einen entsprechenden MIME-Typen zuzuweisen.
mod_negotiation	Der Apache kann anhand von Daten, die der Client an den Server gesendet hat, Informationen in einem durch den Client gewünschten und akzeptierten Format übermitteln. Der Client sendet dabei Informationen über die von ihm gewünschten Datenformate, Spracheinstellungen, Zeichensätze und Codierung und der Apache liefert die Variante einer Information an den Client, die am besten auf die Wünsche des Client passt. Es findet praktisch eine Verhandlung über das Aussehen und das Format der zu liefernden Daten statt. Dieser Vorgang wird Content Negotiation (etwa: Inhaltsverhandlung) genannt und wird durch das Modul mod_negotiation bereitgestellt.
mod_perl	Mod_perl ist sozusagen das Urgestein aller Module und erfreut sich immer noch großer Beliebtheit. Es darf auf einem Webserver nicht fehlen und dient dazu, in Perl geschriebene Programme ausführen zu können.
mod_php	Dynamisch erzeugte Inhalte lassen sich hervorragend mit diesem Modul realisieren und es gibt wohl kaum einen Webserver, der es inzwischen nicht bereitstellt.

*Tabelle 3.6* Modulkonfiguration eines Intranet/Internet Servers, basierend auf der oben genannten Grundkonfiguration sowie der Konfiguration aus dem ersten Szenario (Forts.)

Name des Moduls	Grund für den Einsatz
mod_rewrite	Die Realisierung von internen und externen Weiterleitungen und Redirects anhand regulärer Ausdrücke ist sicherlich ein Feature, welches sehr oft Verwendung findet.
mod_speling	Oft geben Benutzer eine URL falsch oder unvollständig ein und vergessen beispielsweise einen Buchstaben. Mod_speling korrigiert solche Tippfehler und führt den Client dennoch zum gewünschten Ziel.
mod_ssl	Eventuell wird ein Shopsystem angeboten und es müssen sensible Daten (z. B. Kreditkartennummern etc.) transferiert werden. Mit mod_ssl lässt sich eine relativ sichere Übertragung dieser Daten mit dem SSL-Standard realisieren.
mod_suEXEC	Die Verwendung dieses Moduls ist optional, es ermöglicht die Ausführung von CGI- und SSI- Skripten unter der Kennung eines beliebigen lokalen Benutzers. Falls solche Funktionalitäten nicht benötigt werden, kann man dieses Modul ruhig weglassen.

*Tabelle 3.6* *Modulkonfiguration eines Intranet/Internet Servers, basierend auf der oben genannten Grundkonfiguration sowie der Konfiguration aus dem ersten Szenario (Forts.)*

### Fallbeispiel 3

Konfiguration eines kompletten Webservers, der einen großen Anteil von virtuellen Servern (so genannte VirtualHosts) beheimatet. Dies kann beispielsweise in einer ISP-Umgebung (Internet Service Provider) der Fall sein.

Name des Moduls	Grund für den Einsatz
mod_vhost_alias	Aufbauend auf Strukturen des lokalen Dateisystems kann dieses Modul neue virtuelle Server bereitstellen, ohne dafür neue Einträge in der Konfigurationsdatei des Apache vorzunehmen.

*Tabelle 3.7* *Zusätzliche Modulkonfiguration eines Internetservers mit einer Großzahl von virtuellen Servern, die auf der Standardkonfiguration eines Webservers (vgl. Tabelle ■■■) aufbaut*

## 3.4 ./configure bis zum Abwinken

Das *configure*-Skript des Apache bietet wirklich schier unendliche Kombinationsmöglichkeiten der einzelnen Konfigurationsoptionen. Generell können, wie bereits in der Auflistung und Erläuterung der einzelnen Optionen dargestellt, einzelne Module und bestimmte Parameter durch Nutzung der entsprechenden Option aktiviert bzw. deaktiviert werden. Die Verwendung der jeweiligen Optionen muss wirklich sehr individuell an die jeweiligen Gegebenheiten angepasst werden. Ich möchte nun sozusagen als kleine Entscheidungshilfe einige Fallbeispiele präsentieren, die Ihnen helfen sollen, bei der Installation des Apache die von Ihnen wirklich benötigten Optionen zu wählen.

### 3.4.1 Fallbeispiele

#### Fallbeispiel 1:

Der Apache soll unter Verwendung des Layouts *Apache* nach */usr/local/apache2* installiert werden. Es soll dasselbe Laufzeitverhalten verwendet werden, wie im Apache 1.3.x (*prefork*), und der Server soll die Standardmodule des Apache (*mod\_auth*, *mod\_access*, *mod\_log\_config*, *mod\_env*, *mod\_setenvif*, *mod\_mime*, *mod\_autoindex*, *mod\_asis*, *mod\_cgi*, *mod\_dir*, *mod\_userdir*, *mod\_actions*, *mod\_alias* und *mod\_so*) ohne *mod\_imap*, *mod\_include*, *mod\_negotiation* und *mod\_status* verwenden. Deshalb ergibt sich folgender *configure*-Aufruf:

```
# ./configure --enable-layout=Apache --with-mpm=prefork
--disable-status --disable-imap --disable-include
--disable-negotiation
```

Zuerst wird bei diesem Befehl ein Layout (*Apache*, siehe *config.layout*) sowie ein Laufzeitverhalten gewählt (*Prefork*, identisch mit dem Laufzeitverhalten des Apache 1.3.x) und die vier nicht gewünschten Module werden nacheinander deaktiviert durch den Aufruf *--disable-modulname*. Die Übersetzung dieser Konfiguration erfolgt nun durch den Befehl

```
# make
```

und

```
# make install
```

installiert die Software in das Installationsverzeichnis des Layouts *Apache* (*/usr/local/apache2*).

#### Fallbeispiel 2:

Der Apache soll unter Verwendung des Layouts *Apache* wieder nach */usr/local/apache2* installiert werden, wobei die Konfigurationsdateien jedoch unter */etc/httpd/* gespeichert werden sollen. Der Server soll auf Port 8080 horchen, das aus dem Apache 1.3.x bekannte Laufzeitverhalten (*Prefork*) benutzen und eine große

Anzahl an Modulen (Standard- und Zusatzmodule) als *DSO* (*Dynamic Shared Objects*) übersetzen. Der entsprechende *configure*-Aufruf sieht wie folgt aus:

```
# ./configure --enable-layout=Apache --with-mpm=prefork
--sysconfdir=/etc/httpd/ --with-port=8080
--enable-mods-shared=most
```

Sie können diesen Aufruf in einer Zeile eingeben oder Backslashes benutzen, da diese Zeilenumbrüche symbolisieren und den Aufruf insgesamt lesbarer gestalten. Der Befehl ähnelt dem aus Fallbeispiel 1, ist jedoch um die Option *--sysconfdir=/etc/httpd/* erweitert worden, die die Konfigurationsdateien des Apache (u.a. *httpd.conf*) in das Verzeichnis */etc/httpd/* installiert. Durch die Option *--with-port=8080* wird der Apache auf dem TCP-Port 8080 horchen und somit von der Standardeinstellung (TCP, Port 80) abweichen. Die letzte Einstellung *--enable-mods-shared=most* gibt an, dass eine große Anzahl an Standard- und Zusatzmodulen als so genannte *Dynamic Shared Objects* (*DSO*) übersetzt werden sollen. Dazu zählen die Module:

---

mod_access	mod_expires
mod_actions	mod_headers
mod_alias	mod_imap
mod_asis	mod_include
mod_auth	mod_info
mod_auth_anon	mod_log_config
mod_auth_dbm	mod_mime
mod_auth_digest	mod_negotiation
mod_autoindex	mod_rewrite
mod_cgi	mod_setenvif
mod_dav	mod_speling
mod_dav_fs	mod_status
mod_dir	mod_userdir
mod_env	mod_vhost_alias

---

Die Kompilierung und Installation erfolgt, wie bei jedem dieser Fallbeispiele und einer Vielzahl von Unix/Linux-Programmen, mit dem Befehl *make* und *make install*.

### Fallbeispiel 3:

Der Apache soll unter Verwendung eines selbst erstellten Layouts namens *MeinIndianer* installiert werden, wobei die größtmögliche Anzahl an Modulen als *Dynamic Shared Objects* (*DSO*) übersetzt werden sollen. Die Module *mod\_include* und *mod\_imap*, die eigentlich zu den Standardmodulen des Apache gehören, sollen jedoch komplett deaktiviert werden. Es ergibt sich daraus folgender *configure*-Befehl:

```
# ./configure --enable-layout=MeinIndianer --disable-imap
--disable-include --enable-mods-shared=max
```

Der Sinn dieses Befehls sollte klar sein, denn es wird ein Layout namens *MeinIndianer* benutzt, um die maximale Anzahl an Modulen (außer `mod_imap` und `mod_include`) als *Dynamic Shared Objects* zu übersetzen. Die Befehle `make` und `make install` schließen die Installation ab.

#### Fallbeispiel 4:

Unter Verwendung des Multi Processing Module (MPM) *Perchild* soll der Apache nach `/var/web` installiert werden und die Module `mod_rewrite`, `mod_deflate` sowie `mod_alias` sollen als *Dynamic Shared Objects* übersetzt werden. Der Apache soll außerdem zu Entwicklungszwecken `mod_example` anbieten. Es entsteht daraus folgender `configure`-Befehl:

```
# ./configure --prefix=/var/web --with-mpm=perchild
--enable-example --enable-mods-shared="rewrite deflate alias"
```

Die Installation des Häuptlings aller Webserver erfolgt nach `/var/web` und verwendet das Laufzeitverhalten *perchild*. Er bietet die Standardmodule sowie `mod_example` an und hat die Module `mod_rewrite`, `mod_deflate` und `mod_alias` als *Dynamic Shared Objects* übersetzt. Die Kompilierung und Installation wird schließlich durch `make` und `make install` durchgeführt.

#### Fallbeispiel 5:

In einer sehr sicherheitskritischen Umgebung soll ein möglichst minimalistischer Apache installiert werden, der nur über die zum Betrieb unbedingt notwendigen Module verfügt. Es entsteht beispielsweise folgender `configure`-Aufruf:

```
# ./configure --prefix=/usr/local/apache2 --disable-include
--disable-env --disable-status --disable-asis --disable-autoindex
--disable-imap --disable-actions --disable-userdir --enable-auth-dbm --enable-auth-
digest --enable-deflate
```

Dieser Aufruf würde den Apache in das Verzeichnis `/usr/local/apache2` installieren und nur die Module `mod_access`, `mod_auth`, `mod_auth_dbm`, `mod_auth_digest`, `mod_deflate`, `mod_log_config`, `mod_setenvif`, `mod_mime`, `mod_cgi`, `mod_negotiation`, `mod_dir`, `mod_alias` sowie `mod_so` fest in den Server integrieren. Dies ist sicherlich noch nicht die sicherste Modulkonfiguration des Apache, aber eine gute Grundlage. Selbstverständlich wird auch in diesem Beispiel die Kompilierung durch den Befehl `make` und die Installation durch den Befehl `make install` ausgeführt.

Sie haben nun einige Beispiele für den Aufruf des `configure`-Befehls bekommen und können sich nun anhand der o.g. Tabelle den Apache gemäß Ihren Wünschen und Bedürfnissen zusammenbauen und installieren. Nachfolgend möchte ich nun die Installation diverser Zusatzbibliotheken für den Apache und diverse Erweiterungen (u.a. Perl, PHP etc.) unter Unix/Linux erläutern und weitere `configure`-Beispiele präsentieren.

## 3.5 Installation diverser Zusatzsoftware unter Unix/Linux

Die Basisinstallation des Apache ist recht einfach und verlangt eigentlich keine besonderen Voraussetzungen. Alle von mir verwendeten Systeme (u. a. SuSE Linux, Debian Linux, FreeBSD, Sun Solaris, Microsoft Windows), auf denen ich den Apache installiert habe, nutzen durch das jeweilige Betriebssystem vordefinierte Standardinstallationen (ohne Besonderheiten). Die insbesondere in der fortgeschrittenen Installation verwendeten Erweiterungen beruhen oft auf speziellen Softwarepaketen, die nicht immer Teil der Standardinstallation eines Betriebssystems sind. Dieser Abschnitt erläutert deshalb die Installation von einigen Softwarepaketen und Bibliotheken, die Sie installieren müssen, wenn diese nicht von Ihrer Distribution mitgeliefert werden oder Sie diese nicht installiert haben. Hinweis: Falls Sie mit der manuellen Installation von Software unter Unix/Linux nicht sonderlich vertraut sind, sollten Sie dieses Kapitel überspringen und die entsprechenden Softwarepakete mit der Paketverwaltung Ihrer Distribution (z. B. Yast) installieren!

### 3.5.1 Installation von GNU Bison (optional, u. a. für PHP benötigt)

Laden Sie sich die aktuelle Version 1.875 von der Homepage der Bibliothek <http://www.gnu.org/directory/bison.html> herunter und entpacken Sie diese durch `tar xvzf bison-1.875.tar.gz` (ca. 1 MB).

Wechseln Sie in das Verzeichnis `bison-1.875` und rufen Sie das Konfigurationsskript auf, um die Installation vorzubereiten:

```
# ./configure
```

Starten Sie die Kompilierung der Software und installieren Sie diese:

```
# make && make install
```

Die Installation ist damit abgeschlossen. Erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu:

```
# ldconfig
```

### 3.5.2 Installation von GNU Flex (optional, u. a. für PHP benötigt)

Laden Sie sich die aktuelle Version 2.5.4a von der Homepage der Bibliothek <http://www.gnu.org/software/flex/flex.html> (oder von <ftp://ftp.gnu.org/non-gnu/flex/>) herunter und entpacken Sie diese durch `tar xvzf flex-2.5.4a.tar.gz` (373 KB).

Rufen Sie das Konfigurationsskript innerhalb von *flex-2.5.4* auf, um die Installation vorzubereiten:

```
# ./configure
```

Starten Sie die Kompilierung der Software und installieren Sie diese:

```
# make && make install
```

Die Installation von Flex nach */usr/local/bin* ist damit abgeschlossen, erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu:

```
# ldconfig
```

### 3.5.3 Installation von GNU GDBM (optional, u. a. für PostgreSQL benötigt)

Laden Sie sich die aktuelle Version 1.8.3 von <ftp://ftp.gnu.org/gnu/gdbm/> herunter und entpacken Sie diese durch *tar xvzf gdbm-1.8.3.tar.gz*. (223 KB)

Wechseln Sie in das Verzeichnis *gdbm-1.8.3*. Rufen Sie das Konfigurationsskript auf, um die Installation vorzubereiten:

```
# ./configure
```

Starten Sie die Kompilierung der Software und installieren Sie diese:

```
# make && make install
```

Die Installation ist damit abgeschlossen. Erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu:

```
# ldconfig
```

### 3.5.4 Installation von LibNCurses (optional, u. a. für Kernelkonfiguration und LibReadline benötigt)

Normalerweise wird die *ncurses* (*new curses*) Bibliothek bei vielen Distributionen standardmäßig mitinstalliert. Wenn Sie diese jedoch nicht installiert haben, müssen Sie sich diese von der Internetseite <http://www.gnu.org/software/ncurses/ncurses.html> herunterladen und entpacken mit dem Befehl *tar xvzf ncurses-5.3.tar.gz* (2 MB).

Starten Sie innerhalb des Verzeichnisses *ncurses-5.3* durch den Aufruf des mitgelieferten *configure*-Skriptes die Konfiguration der Software:

```
# ./configure
```

Sobald der Vorgang abgeschlossen ist, können Sie die Kompilierung der *ncurses*-Bibliothek starten:

```
# make
```

Installieren Sie die Software, erzeugen Sie die Liste der bekannten Bibliotheken neu und schließen Sie damit die Installation der Bibliothek ab:

```
# make install && ldconfig
```

### 3.5.5 Installation von FreeType (optional, u.a. für PHP benötigt)

Die Homepage der FreeType-Bibliothek lautet <http://www.freetype.org>, laden Sie sich von dort die aktuelle Version (zur Zeit 2.1.5) herunter und entpacken Sie diese durch Eingabe von `tar xzvf freetype-2.1.5.tar.gz` (1,1 MB).

Wechseln Sie in das neu erzeugte Verzeichnis `freetype-2.1.5` (`cd freetype-2.1.5`) und rufen Sie die Übersicht der zur Verfügung stehenden Konfigurationsoptionen auf:

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

Nach Durchsicht der Optionen habe ich mich für folgenden Aufruf entschieden:

```
# ./configure --prefix=/usr/local/freetype-2.1.5
```

#### Hinweis

Sofern Sie kein spezielles Installationsverzeichnis (z.B. `/usr/local/freetype-2.1.5`) wünschen, können Sie auch nur den Befehl `./configure` benutzen.

Die Konfigurierung der Software beginnt, diverse Abhängigkeiten (u.a. `libtool` etc.) werden geprüft und sobald das Skript beendet ist, kann die Kompilierung von FreeType 2.1.5 beginnen:

```
# make
```

Wenn die Kompilierung ohne Fehlermeldung abgeschlossen ist, können Sie die Software nach `/usr/local/freetype-2.1.5` installieren:

```
# make install
```

Die Installation sollte ohne Probleme klappen. Damit das System die erzeugten Bibliotheken finden kann, sollten Sie diese entweder in das Verzeichnis `/lib` kopieren oder die Systempfade für Bibliotheken entsprechend erweitern. Wenn Sie die Bibliotheken nach `/lib` kopieren möchten, hilft Ihnen der folgende Befehl:

```
# cp /usr/local/freetype-2.1.5/lib/* /lib
```

Alternativ können Sie die Suchpfade für Systembibliotheken, die in der Datei `/etc/ld.so.conf` definiert werden, um das Unterverzeichnis `lib` Ihrer FreeType-Installation erweitern. Nutzen Sie dazu folgenden Einzeiler:

```
# echo /usr/local/freetype-2.1.5/lib >> /etc/ld.so.conf
```

Der Befehl leitet die Ausgabe des Kommandos *echo* in die Datei */etc/ld.so.conf* um und fügt dadurch dieser Datei einen Verweis auf das Verzeichnis */usr/local/freetype-2.1.5/lib* hinzu. Schließlich muss das System über das Vorhandensein und den Speicherort der neuen Bibliotheken informiert werden. Diesen Part erledigt folgender Befehl, den Sie immer ausführen müssen, wenn Sie neue Bibliotheken auf Ihrem System installiert haben:

```
# ldconfig
```

Die Installation von FreeType 2.1.5 ist damit abgeschlossen.

### 3.5.6 Installation von zlib (optional, u. a. für PHP und GD-Lib benötigt)

Die Homepage der freien Datenkompressionsbibliothek lautet <http://www.zlib.org>, die aktuelle Version ist 1.2.1. Laden Sie sich die Software von dieser Seite oder einem ihrer vielen Mirrors herunter und extrahieren Sie diese durch Eingabe von *tar xvzf zlib-1.2.1.tar.gz* (338 KB).

Wechseln Sie in das neu erzeugte Verzeichnis *zlib-1.2.1* und rufen Sie die Übersicht der Konfigurationsoptionen auf:

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

Die Liste ist bei dieser Software extrem kurz, die Wahl der Konfigurationsoptionen fällt deshalb sehr leicht:

```
# ./configure --prefix=/usr/local/zlib-1.2.1 --shared
```

#### Hinweis

Wenn Sie kein spezielles Installationsverzeichnis definieren möchten, können Sie auch nur den Befehl *./configure --shared* verwenden.

Die Konfiguration der Software ist extrem schnell fertig, übersetzen Sie nun die Software:

```
# make
```

Der Kompilierungsvorgang sollte ohne Fehler durchlaufen. Installieren Sie nun die Software mit dem Befehl

```
# make install
```

Als Nächstes müssen Sie das System darüber informieren, dass Sie neue Bibliotheken installiert haben. Der bekannte Einzeiler erledigt dies mal wieder:

```
# echo /usr/local/zlib-1.2.1/lib >> /etc/ld.so.conf
```

Alternativ können Sie die Dateien des Unterverzeichnisses *lib* Ihrer zlib-Installation in einen dem System bekannten Pfad für Bibliotheken wie z. B. */lib* kopieren. Lassen Sie das System nun die Liste der Bibliotheken neu erzeugen:

```
# ldconfig
```

So einfach ist das: Die freie Datenkompressionsbibliothek zlib ist nun vollständig installiert.

### 3.5.7 Installation von JPEGsrc (optional, u. a. für PHP benötigt)

JPEGsrc ist freie Bibliothek zur Kompression und Dekompression von JPEG-Bildern. Die Homepage dieser Bibliothek lautet <http://www.ijg.org>, laden Sie sich die jeweils aktuelle Version (momentan 6b) herunter und entpacken Sie diese durch Eingabe von `tar xvzf jpegsrc.v6b.tar.gz` (599 KB).

Rufen Sie im neu erzeugten Verzeichnis *jpeg-6b* die Liste der zur Verfügung stehenden Konfigurationsoptionen auf und studieren Sie die möglichen Optionen:

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

Ich habe mich nach Durchsicht der möglichen Optionen für folgenden Aufruf des *configure*-Skriptes entschieden:

```
# ./configure --prefix=/usr/local/jpegsrc-6b --enable-shared
```

#### Hinweis

Auch hier ist die Angabe eines Installationsverzeichnisses (`--prefix=/usr/local/jpegsrc-6b`) optional und kann weggelassen werden.

Langsam wird es zur Gewohnheit: Die Übersetzung der Software erfolgt durch den Aufruf von

```
# make
```

Dem Makefile von JPEGsrc fehlen leider die entsprechenden Aufrufe, um die nötigen Installationsverzeichnisse zu erzeugen. Ich habe auch diesen Fehler (bzw. Feature) den Autoren der Bibliothek gemeldet und eventuell sind diese bereinigt worden, wenn Sie das Buch in den Händen halten. Falls die Fehler nicht behoben worden sind, müssen Sie halt die benötigten Verzeichnisse per Hand erstellen:

```
# mkdir /usr/local/jpegsrc-6b
# mkdir /usr/local/jpegsrc-6b/lib
# mkdir /usr/local/jpegsrc-6b/include
# mkdir /usr/local/jpegsrc-6b/bin
# mkdir /usr/local/jpegsrc-6b/man
# mkdir /usr/local/jpegsrc-6b/man/man1
```

Nun können Sie die Software, gemäß Ihren Angaben beim Aufruf des *configure*-Skriptes, erfolgreich unter */usr/local/jpegsrc-6b* installieren durch Ausführen des Befehls

```
# make install
```

Da Sie eine Bibliothek installiert haben, müssen Sie das System über das Vorhandensein einer neuen Bibliothek informieren. Fügen Sie dazu eine Zeile der Datei */etc/ld.so.conf* hinzu, die auf den Speicherort der neu installierten Bibliothek von JPEGSrc verweist:

```
# echo /usr/local/jpegsrc-6b/lib >> /etc/ld.so.conf
```

Erzeugen Sie nun die Liste der dem System bekannten und verwendeten Bibliotheken neu mit dem Befehl:

```
# ldconfig
```

Kopieren Sie außerdem die Include-Dateien mit folgendem Kommando:

```
# cp /usr/local/jpegsrc-6b/include/* /usr/local/include
```

Die Installation dieser JPEG Bibliothek ist abgeschlossen.

### 3.5.8 Installation von LibPNG (optional, u. a. für PHP und GD-Lib benötigt)

Die LibPNG wird von der GD-Lib benötigt und ist im Internet unter <http://www.libpng.org> zu erreichen. Laden Sie sich die aktuelle Version (z. Z. 1.2.5) herunter und entpacken Sie diese durch das Kommando *tar xvzf libpng-1.2.5.tar.gz* (495 KB).

Es existiert für LibPNG kein *configure*-Skript, welches für uns die Erzeugung des *Makefiles* übernimmt. Im Verzeichnis *libpng-1.2.5* existiert ein Unterverzeichnis namens *scripts*, welches *Makefiles* für diverse Plattformen enthält. Wir kopieren uns deshalb ein passendes *Makefile* durch Eingabe von

```
# cp scripts/makefile.linux makefile
```

Editieren Sie nun diese Datei und setzen Sie in Zeile 16 ein späteres Installationsverzeichnis. Ich habe mich dabei für das Verzeichnis */usr/local/libpng-1.2.5* entschieden, demnach sieht diese Zeile wie folgt aus:

```
prefix=/usr/local/libpng-1.2.5
```

Speichern Sie das Makefile und erzeugen Sie das von Ihnen gewünschte Installationsverzeichnis per Hand:

```
# mkdir /usr/local/libpng-1.2.5
```

Starten Sie schließlich die Kompilierung von LibPNG 1.2.5 mit:

```
# make
```

Die Installation dieser Bibliothek erledigt der Befehl:

```
# make install
```

Erweitern Sie die Liste der dem System bekannten Pfade für Bibliotheken oder kopieren Sie die neu installierten Bibliotheken in einen bereits bekannten Pfad. Wenn Sie den Suchpfad für Bibliotheken erweitern möchten, hilft Ihnen dieser Befehl:

```
# echo /usr/local/libpng-1.2.5/lib >> /etc/ld.so.conf
```

Erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu:

```
# ldconfig
```

Die Installation der nächsten Bibliothek ist damit abgeschlossen.

### 3.5.9 Installation von GD-Lib (optional, u. a. für PHP benötigt)

Die GD-Lib ist eine ANSI-C Bibliothek zur dynamischen Erzeugung von Bildern und Grafiken in diversen Formaten (u. a. PNG und JPEG). Die Homepage dieser Software lautet <http://www.boutell.com/gd/> und ist aktuell in der Version 2.0.17. Laden Sie sich die aktuelle Version (540 KB) herunter und entpacken Sie diese mit `tar xvzf gd-2.0.17.tar.gz`.

Wechseln Sie nun in das neu entpackte Verzeichnis `gd-2.0.17` (`cd gd-2.0.17`) und starten Sie die Vorbereitung der Software durch Eingabe von:

```
# ./configure
```

**Hinweis**

Unter Umständen kann es nötig sein, dass Sie die Installationsverzeichnisse der FreeType-, LibPNG- und LibJPEG-Bibliothek durch die Optionen `--with-freetype=Verzeichnis`, `--with-png=Verzeichnis` und `--with-jpeg=Verzeichnis` angeben müssen. Optional ist die Wahl eines speziellen Installationsverzeichnisses mit `./configure --prefix=/usr/local/gd-2.0.17`.

Starten Sie daraufhin die Kompilierung der Software:

```
# make
```

Schließlich können Sie die Installation der Software durch das Kommando

```
# make install
```

abschließen. Sofern Sie ein spezielles Installationsverzeichnis für die Software gewählt haben (z. B. `/usr/local/gd-2.0.17`) müssen Sie die Liste der bekannten Pfade für Bibliotheken erweitern:

```
# echo /usr/local/gd-2.0.17/lib >> /etc/ld.so.conf
```

Erzeugen Sie daraufhin die Liste der bekannten Bibliotheken neu:

```
# ldconfig
```

Die Installation ist damit abgeschlossen.

### 3.5.10 Installation der LibTiff (optional, u. a. für PHP und PDFLib benötigt)

Laden Sie sich von <http://www.libtiff.org> die aktuelle Version (momentan 3.5.7) herunter und entpacken Sie diese durch die Eingabe von `tar xvzf tiff-v3.5.7.tar.gz` (929 KB). Schauen Sie sich (in `tiff-v3.5.7`) nun die Übersicht der möglichen Konfigurationsoptionen an:

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

Da die Anzahl der möglichen Optionen wirklich sehr gering ist, habe ich mich für folgenden Aufruf entschieden:

```
# ./configure --prefix=/usr/local/libtiff-3.5.7
```

#### Hinweis

Die Angabe eines separaten Installationsverzeichnis (`--prefix=...`) ist optional.

Die Software wird für die Kompilierung und Installation vorbereitet und es wird Ihnen eine Übersicht der späteren Installationsverzeichnisse aufgelistet. Bestätigen Sie diese Übersicht durch Eingabe von `yes` und die Software wird entsprechend vorbereitet. Starten Sie die Kompilierung mit:

```
# make
```

Erzeugen Sie nun das Installationsverzeichnis der LibTiff 3.5.7 unterhalb des Verzeichnisses `/usr/local`:

```
# mkdir /usr/local/libtiff-3.5.7
```

Installieren Sie die Software durch folgenden Befehl:

```
# make install
```

Da es sich bei LibTiff um eine Bibliothek handelt, die auch von anderen Programmen verwendet wird, muss der Speicherort dieser Bibliothek dem System bekannt gegeben werden, sofern ein spezielles Installationsverzeichnis (z.B. `/usr/local/libtiff-3.5.7`) verwendet wird. Die Liste der Verzeichnisse, in denen das System Bibliotheken sucht bzw. findet, steht in der Datei `/etc/ld.so.conf`. Tragen Sie deshalb den Pfad zu Ihrer LibTiff-Installation in diese Datei ein:

```
# echo /usr/local/libtiff-3.5.7/lib >> /etc/ld.so.conf
```

Erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu und beenden Sie damit die Installation der LibTiff:

```
# ldconfig
```

### 3.5.11 Installation der PDFLib (optional, u.a. für PHP benötigt)

Falls Sie aus Ihren Programmen dynamisch PDF-Dateien erzeugen möchten, benötigen Sie die PDFLib von Thomas Merz. Laden Sie sich die aktuelle Version (moment 5.0.2) von <http://www.pdflib.com> herunter und entpacken Sie diese mit *tar xvzf PDFlib-Lite-5.0.2-Unix-src.tar.gz* (2,6 MB). Achtung: Die Bibliothek ist in ihrem Funktionsumfang abgespeckt und nur bedingt kostenlos. Sie muss, insbesondere bei gewerblichem Einsatz, lizenziert werden. Weitere Informationen finden Sie auf der Homepage der PDFLib!

#### Hinweis

Es gibt zwei verschiedene Versionen der PDFLib, die vorkompilierte und nicht vorkompilierte, d.h. im Quelltext vorliegende Variante. Achten Sie darauf, dass Sie die im Quelltext verfügbare Version herunterladen (siehe <http://www.pdflib.com/products/pdflib/download-source.html>).

Wechseln Sie in das neu entpackte Verzeichnis *PDFlib-Lite-5.0.2-Unix-src* und schauen Sie sich die zur Verfügung stehenden Konfigurationsoptionen an:

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

Ich würde gerne in Java, C++ und Perl dynamisch PDF-Dateien erzeugen und habe mich deshalb für folgenden Konfigurationaufruf entschieden (eine Zeile!):

```
# ./configure --prefix=/usr/local/pdflib-5.0.2 --enable-shared
--with-java=/usr/local/java --with-perl=/usr/bin/perl --enable-cxx
```

#### Hinweis

Sofern Sie die verschiedenen Optionen (z.B. *--with-java*) nicht benötigen, können Sie diese auch weglassen.

Sie erhalten eine Zusammenfassung der von Ihnen gewählten Optionen und können nun die Kompilierung der PDFLib starten:

```
# make
```

Sobald diese erfolgreich abgeschlossen ist, können Sie die Software installieren:

```
# make install
```

Die reine Installation der PDFLib ist abgeschlossen, erweitern Sie zusätzlich den Suchpfad für Systembibliotheken um den Pfad zu Ihrer PDFLib-Installation, damit

diese Erweiterung dem System bekannt ist und von verschiedenen Programmen genutzt werden kann. Benutzen Sie dazu folgenden Befehl:

```
# echo /usr/local/pdflib-5.0.2/lib >> /etc/ld.so.conf
```

**Hinweis**

Dieser Schritt ist nur bei Verwendung eines eigenen Installationsverzeichnis (z. B. `/usr/local/pdflib-5.0.2`) notwendig.

Dieser flotte Einzeiler fügt der Datei `/etc/ld.so.conf`, die die Liste der Verzeichnisse enthält, in denen das System Bibliotheken suchen soll, einen Eintrag hinzu, der auf das Unterverzeichnis `lib` der soeben getätigten PDFLib-Installation verweist und somit die in diesem Verzeichnis installierte Bibliothek `libpdf.so` dem System zur Verfügung stellt. Erzeugen Sie nun die Liste der dem System bekannten Bibliotheken neu und beenden Sie damit die Installation der PDFLib:

```
# ldconfig
```

Alternativ können Sie die neu installierte Bibliothek auch einfach in ein dem System bereits als Suchpfad für Bibliotheken bekanntes Verzeichnis kopieren:

```
# cp /usr/local/pdflib-5.0.2/lib/libpdf.so /lib
```

Ein einfacher symbolischer Verweis erledigt diesen Job natürlich auch:

```
# ln -s /usr/local/pdflib-5.0.2/lib/libpdf.so /lib/libpdf.so
```

### 3.5.12 Installation von GhostScript (optional)

Ich werde Ihnen in einem späteren Kapitel vorstellen, wie Sie beliebige Inhalte mit Linux-Bordmitteln in PDF-Dateien umwandeln können. Besorgen Sie sich dazu von <http://www.cs.wisc.edu/~ghost/> die aktuellste Version von GhostScript (momentan 7.07) und entpacken Sie diese durch Eingabe von `tar xvfz ghostscript-7.07.tar.gz` (4,7 MB).

Wechseln Sie in das neu entpackte Verzeichnis von GhostScript und schauen Sie sich die zur Verfügung stehenden Konfigurationsoptionen an:

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

Zur Installation von GhostScript nach `/usr/local/ghostscript-7.07` habe ich folgenden `configure`-Befehl gewählt:

```
# ./configure --prefix=/usr/local/ghostscript-7.07
```

**Hinweis**

Die Angabe eines besonderen Installationsverzeichnis ist optional.

Sobald dieser Befehl abgeschlossen ist, können Sie die Kompilierung der Software starten:

```
# make
```

Die Installation der Software erledigt dieses Kommando schnell:

```
# make install
```

Die Installation von GhostScript ist damit abgeschlossen. Sie benötigen jedoch zusätzlich noch ein Programm namens *a2ps*, welches Sie unter <http://www.gnu.org/software/a2ps/a2ps.html> herunterladen müssen. Entpacken Sie auch dieses Programm (aktuelle Version 4.13b) mit Hilfe des Befehls `tar xvzf a2ps-4.13b.tar.gz` (1,85 MB) und rufen Sie im Verzeichnis *a2ps-4.13* die Konfigurationsoptionen auf:

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

Nach Durchsicht der verfügbaren Optionen habe ich mich für folgenden Aufruf entschieden:

```
# ./configure --prefix=/usr/local/a2ps-4.13
```

**Hinweis**

Wie immer ist die Definition eines Installationsverzeichnis optional.

Sobald dieser Befehl abgeschlossen ist, können Sie die Kompilierung der Software starten:

```
# make
```

Die Installation der Software ist flott erledigt:

```
# make install
```

Die Installation von *a2ps* ist damit abgeschlossen.

## 3.6 Apache 2 in einer Chroot-Umgebung

Ich habe in diesem Buch bereits mehrfach die Vor- und Nachteile einer Chroot-Umgebung vorgestellt (u. a. im Kapitel `mod_security`). Im folgenden Kapitel möchte ich die Installation des Apache 2 in einer solchen Umgebung veranschaulichen, wobei ich die relativ aufwändige Installation von Drittanbietermodulen (z.B. `mod_php`) nicht besprechen werde.

**Hinweis**

Die Basis dieses Kapitels bildet eine Anleitung, die ursprünglich von Jan Mattila geschrieben und unter <http://www.cgisecurity.com/webserver/apache/chrootapache2-howto.html> veröffentlicht worden ist.

## 3.6.1 Grundinstallation des Apache 2

### Hinweis

Wenn Sie bereits über eine funktionsfähige lokale Installation des Apache 2 verfügen, können Sie dieses Kapitel überspringen. Erstellen Sie aber dennoch einen separaten Benutzer zur Ausführung des Apache (vgl. Ende dieses Kapitels).

Die Grundinstallation des Apache 2 erfolgt bei der Verwendung einer Chroot-Umgebung wie gewohnt. Laden Sie sich von <http://httpd.apache.org> die aktuellste Version (2.0.48) des Apache 2 herunter (6,25 MB) und entpacken Sie diese mit

```
# tar xvzf httpd-2.0.48.tar.gz
```

Wechseln Sie in das neu erzeugte Verzeichnis `httpd-2.0.48` (`cd httpd-2.0.48`) und rufen Sie eine Übersicht der zur Verfügung stehenden Optionen auf:

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

Nach Durchsicht der verfügbaren Optionen können Sie die Vorbereitung der Installation beispielsweise wie folgt starten (Standardkonfiguration, spätere Installation nach `/usr/local/apache2`):

```
# ./configure --prefix=/usr/local/apache2
```

### Hinweis

Falls Sie weitere Optionen verwenden möchten, können Sie diese dem oben genannten Befehl einfach anhängen. Aus Sicherheitsgründen ist die Deaktivierung der Module `mod_actions`, `mod_asis`, `mod_autoindex`, `mod_imap`, `mod_include`, `mod_status`, `mod_userdir`, `mod_auth_anon`, `mod_cern_meta`, `mod_digest`, `mod_dav`, `mod_info` und `mod_proxy_ratsam`, sofern diese nicht unbedingt benötigt werden.

Sobald dieser Befehl abgeschlossen ist, können Sie die Kompilierung des Apache 2 starten:

```
# make
```

Schließlich kann die Installation der Software nach `/usr/local/apache2` durch Eingabe des Befehls

```
# make install
```

erfolgen. Aus Sicherheitsgründen sollte danach eine separate Gruppe (`apache2`) und ein separater Benutzer erstellt werden, die ausschließlich zur Ausführung des Apache 2 verwendet werden. Die Erstellung einer neuen Gruppe erfolgt durch den folgenden Befehl:

```
# groupadd apache2
```

Dieser Gruppe fügen wir einen einzigen Benutzer namens *apache2* hinzu, der später zur Ausführung des Apache verwendet wird:

```
# useradd -g apache2 -d /dev/null -s /bin/false apache2
```

Das Heimatverzeichnis des Benutzers wird auf das virtuelle Datennirwana */dev/null* gesetzt. Durch die Verwendung des Programms */bin/false* als Loginshell wird es ein Angreifer, der eventuell durch einen gezielten Angriff gegen den Apache Zugriff auf das lokale System erhalten hat, sehr schwer haben, diese Zugriffsmöglichkeit zu missbrauchen. Widmen wir uns nun dem Aufbau der eigentlichen Chroot-Umgebung:

### 3.6.2 Einrichtung der Chroot-Umgebung

Als Basisverzeichnis für die Chroot-Umgebung habe ich mich für das Verzeichnis */chroot/httpd* entschieden. Sie können prinzipiell jedes beliebige Verzeichnis dazu benutzen. Sofern Sie sich für ein anderes Verzeichnis entschieden haben, müssen Sie die nachfolgend aufgeführten Befehle entsprechend anpassen. Bitte beachten Sie außerdem, dass es aus einer gesicherten Chroot-Umgebung heraus unter normalen Umständen nicht möglich ist, auf Dateien und Verzeichnisse außerhalb dieser Umgebung auf Dateisystemebene zuzugreifen. Aufgrund dessen müssen alle Dateien, Geräte und Verzeichnisse, die der Apache zur Ausführung benötigt, manuell aus dem normalen Dateisystem in die Chroot-Umgebung kopiert beziehungsweise in der Chroot-Umgebung erneut erstellt werden. Dazu muss zunächst die Basisverzeichnisstruktur innerhalb der Chroot-Umgebung erstellt werden:

```
# mkdir -p /chroot/httpd/dev
# mkdir -p /chroot/httpd/etc
# mkdir -p /chroot/httpd/lib
# mkdir -p /chroot/httpd/usr/lib
# mkdir -p /chroot/httpd/usr/libexec
# mkdir -p /chroot/httpd/usr/local/apache2/bin
# mkdir -p /chroot/httpd/usr/local/apache2/logs
# mkdir -p /chroot/httpd/usr/local/apache2/conf
# mkdir -p /chroot/httpd/usr/local/apache2/lib
# mkdir -p /chroot/httpd/usr/local/apache2/htdocs
# mkdir -p /chroot/httpd/usr/share/zoneinfo/Europe
# mkdir -p /chroot/httpd/var/run
```

Diese Befehle ergeben eine minimale Verzeichnisstruktur unterhalb des als Chroot-Basisverzeichnis definierten Verzeichnisses */chroot/httpd*, in die später Dateien und Verzeichnisse aus dem »echten« Root-Verzeichnisbaum (*/*) kopiert werden.

Für die Ausführung des Apache ist das Vorhandensein des virtuellen Datennirwanas */dev/null* sowie des Gerätes */dev/urandom* notwendig, so dass wir diese innerhalb der Chroot-Umgebung erzeugen müssen:

```
# mknod /chroot/httpd/dev/null c 1 3
# mknod /chroot/httpd/dev/urandom c 2 4
```

Außerdem müssen wir für dieses Gerät spezielle Berechtigungen definieren:

```
# chown root.root /chroot/httpd/dev/null
# chmod 666 /chroot/httpd/dev/null
```

Nun folgt wohl der unangenehmste und zeitraubendste Teil dieses Kapitels: Da der Apache während der Ausführung in der Chroot-Umgebung nicht auf die benötigten (externen) Bibliotheken zugreifen kann, müssen diese Bibliotheken in die gesicherte Umgebung kopiert werden. Dazu müssen wir jedoch zunächst mit Hilfe des Befehls `ldd` herausfinden, welche Bibliotheken der Apache benötigt:

```
# ldd /usr/local/apache2/bin/httpd
```

Die Ausgabe dieses Befehls ist recht lang und zeigt die durch den Apache zur Ausführung verwendeten Bibliotheken:

```
libaprutil-0.so.0 => /usr/local/apache2.chroot/lib/libaprutil-0.so.0 (0x40015000)
libgdbm.so.2 => /usr/lib/libgdbm.so.2 (0x4003d000)
libdb-4.0.so => /usr/lib/libdb-4.0.so (0x40044000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x400dd000)
libapr-0.so.0 => /usr/local/apache2.chroot/lib/libapr-0.so.0 (0x40103000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40124000)
librt.so.1 => /lib/librt.so.1 (0x40175000)
libm.so.6 => /lib/libm.so.6 (0x40188000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x401aa000)
libnsl.so.1 => /lib/libnsl.so.1 (0x401d7000)
libdl.so.2 => /lib/libdl.so.2 (0x401ec000)
libc.so.6 => /lib/libc.so.6 (0x401ef000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

### Hinweis

Unter Umständen kann die Anzahl der benutzten Bibliotheken auf dem von Ihnen verwendeten System von dieser Ausgabe abweichen.

Die Bibliotheken müssen nun händisch in die Chroot-Umgebung kopiert werden, damit diese dem Apache auch innerhalb der Chroot-Umgebung zur Ausführung zur Verfügung stehen. Da wir recht bequem sind, können wir uns ein kleines Skript schreiben, welches diese Aufgabe für uns übernimmt:

```
# for i in `ldd /usr/local/apache2/bin/httpd | awk '{ print $3 }'`; do cp $i
/chroot/httpd/lib/`basename $i`; done
```

### Hinweis

Das ```-Zeichen ist ein so genannter Backtick, den Sie auf Ihrer Tastatur links neben der `←`-Taste (Rückschritttaste) finden. Um dieses Zeichen zu erhalten, drücken Sie die `⇧`-Taste und die Taste links neben dem `←` sowie danach die `⇧`. Innerhalb des `awk`-Befehls werden einfache Anführungszeichen verwendet, die Sie durch Betätigen der `⇧`-Taste sowie der Taste oberhalb der rechten `⇧`-Taste (`#`-Taste) erhalten.

Dieses kleine Shellskript führt den oben genannten Befehl aus und kopiert in einer for-Schleife jede verwendete Bibliothek in das Zielverzeichnis `/chroot/httpd/lib`. Der `awk`-Befehl sorgt dafür, dass aus der Ausgabe des Befehls `ldd` nur der volle Dateipfad einer Bibliothek (z.B. `/usr/local/apache2/lib/libaprutil-0.so.0`) in der Variablen `i` gespeichert wird. Diese Variable `i` wird mit dem Namen `$i` durch den Kopierbefehl `cp` angesprochen und an die korrekte Stelle in der Chroot-Umgebung (`/chroot/httpd/lib`) kopiert. Der Befehl `basename` extrahiert dabei den Dateinamen (z.B. `libaprutil-0.so.0`) aus der vollen Pfadangabe (z.B. `/usr/local/apache2/lib/libaprutil-0.so.0`). Alles klar?

In dem Verzeichnis `/chroot/httpd/lib` befinden sich nun alle Bibliotheken, die der Apache zur Ausführung benötigt. Leider benötigen diese Bibliotheken selbst wieder eine Reihe von Bibliotheken, wie der Befehl `ldd` herausfindet:

```
# ldd /chroot/httpd/lib/*
```

Die benötigten Bibliotheken sind:

```
/chroot/httpd/lib/ld-linux.so.2:
statically linked
/chroot/httpd/lib/libapr-0.so.0:
    libc.so.6 => /lib/libc.so.6 (0x40034000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
/chroot/httpd/lib/libaprutil-0.so.0:
    libc.so.6 => /lib/libc.so.6 (0x40029000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
/chroot/httpd/lib/libc.so.6:
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/libcrypt.so.1:
    libc.so.6 => /lib/libc.so.6 (0x40027000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/libdb-4.0.so:
    libc.so.6 => /lib/libc.so.6 (0x400ac000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
/chroot/httpd/lib/libdl.so.2:
    libc.so.6 => /lib/libc.so.6 (0x40027000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/libexpat.so.0:
    libc.so.6 => /lib/libc.so.6 (0x40039000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
/chroot/httpd/lib/libgdbm.so.2:
    libc.so.6 => /lib/libc.so.6 (0x4001a000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
/chroot/httpd/lib/libm.so.6:
    libc.so.6 => /lib/libc.so.6 (0x40027000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/libnsl.so.1:
    libc.so.6 => /lib/libc.so.6 (0x40027000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/libpthread.so.0:
```

```

libc.so.6 => /lib/libc.so.6 (0x40027000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/chroot/httpd/lib/librt.so.1:
libc.so.6 => /lib/libc.so.6 (0x40027000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40154000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

```

**Hinweis**

Unter Umständen variiert die Ausgabe auf dem von Ihnen benutzten System.

Wie die Ausgabe erkennen lässt, benötigen im Prinzip alle Bibliotheken dieselben drei Dateien, so dass wir diese schnell manuell in die Chroot-Umgebung kopieren können (eine Zeile):

```
# cp /lib/libc.so.6 /lib/ld-linux.so.2 /lib/libpthread.so.0 /chroot/httpd/lib/
```

Zusätzlich benötigt der Apache noch zwei weitere Bibliotheken, die wir ebenfalls in die Chroot-Umgebung kopieren müssen (eine Zeile):

```
# cp /lib/libnss_compat.so.2 /lib/libnss_files.so.2 /chroot/httpd/lib/
```

Neben den genannten Bibliotheken braucht der Apache zur erfolgreichen Ausführung eine Menge weiterer Dateien. Eine Liste dieser Dateien liefert uns unter anderem der Befehl *strace*, der in jeder modernen Unix- beziehungsweise Linuxvariante enthalten sein sollte und Details (Systemaufrufe, Signale etc.) über den Ablauf eines Programms ausgibt.

```
# strace -o apache_ablauf.txt /usr/local/apache2/bin/httpd
```

Durch Angabe des Parameters *-o* ist die Ausgabe des *strace*-Befehls in die Datei *apache\_ablauf.txt* geschrieben worden, damit wir diese manuell analysieren können. Bevor wir jedoch diese Ausgabe untersuchen können, müssen wir zunächst den Apache beenden, der durch das Programm *strace* gestartet worden ist:

```
# killall httpd
```

Damit wir die Dateien finden, die zur Ausführung des Apache 2 in der Chroot-Umgebung benötigt werden, müssen wir die Protokolldatei *apache\_ablauf.txt* nach allen Systemaufrufen durchsuchen, in denen Dateien geöffnet werden (engl.: open):

```
# cat apache_ablauf.txt | grep open
```

Dieser Befehl produziert folgende Ausgabe (gekürzt):

```

open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/usr/local/apache2/lib/i686/mmx/libaprutil-0.so.0", O_RDONLY) = -1 ENOENT (No
such file or directory)
open("/usr/local/apache2/lib/i686/libaprutil-0.so.0", O_RDONLY) = -1 ENOENT (No such

```

```

file or directory)
open("/usr/local/apache2/lib/mmx/libaprutil-0.so.0", O_RDONLY) = -1 ENOENT (No such
file or directory)
open("/usr/local/apache2/lib/libaprutil-0.so.0", O_RDONLY) = 3

[ ... ]

open("/usr/local/apache2/lib/libnss_compat.so.2", O_RDONLY) = -1 ENOENT (No such file
or directory)
open("/etc/ld.so.cache", O_RDONLY) = 4
open("/lib/libnss_compat.so.2", O_RDONLY) = 4
open("/etc/passwd", O_RDONLY) = 4
open("/etc/resolv.conf", O_RDONLY) = 4
open("/usr/local/apache2/lib/libnss_files.so.2", O_RDONLY) = -1 ENOENT (No such file
or directory)
open("/etc/ld.so.cache", O_RDONLY) = 4
open("/lib/libnss_files.so.2", O_RDONLY) = 4
open("/etc/host.conf", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/hosts", O_RDONLY) = 4
open("/usr/local/apache2/logs/access_log", O_WRONLY|O_APPEND|O_CREAT, 0666) = 4
open("/usr/local/apache2/logs/error_log", O_RDWR|O_APPEND|O_CREAT, 0666) = 7
open("/usr/local/apache2/conf/mime.types", O_RDONLY) = 8
open("/usr/local/apache2/conf/httpd.conf", O_RDONLY) = 5

```

Um den Apache erfolgreich in der Chroot-Umgebung zu betreiben, müssen wir alle Dateien, die die Ausgabe des oben genannten Befehls geliefert hat, in die gesicherte Umgebung kopieren. Allerdings haben wir eine Vielzahl der ausgeführten Dateien bereits aus dem Verzeichnis */lib* in die Chroot-Umgebung kopiert, so dass wir nur noch folgende Dateien aus dem Verzeichnis */etc* kopieren müssen (eine Zeile):

```
# cp /etc/resolv.conf /etc/nsswitch.conf /etc/hosts /chroot/httpd/etc/
```

Zusätzlich benötigen wir noch die Dateien */etc/passwd* und */etc/group*, wobei wir aus Sicherheitsgründen nur die Informationen über die Gruppe beziehungsweise den Benutzer *apache2* in die entsprechenden Dateien der Chroot-Umgebung schreiben:

```
# grep apache2 /etc/passwd > /chroot/httpd/etc/passwd
# grep apache2 /etc/group > /chroot/httpd/etc/group
```

Des Weiteren brauchen wir Informationen über die Zeitzone, in der sich der Server befindet. Dazu kopieren wir einfach die Datei */usr/share/zoneinfo/Europe/Berlin* in die Chroot-Umgebung:

```
# cp /usr/share/zoneinfo/Europe/Berlin /chroot/httpd/usr/share/zoneinfo/Europe/
```

Schließlich benötigen wir die Dateien *mime.types*, *error\_log* und *access\_log*, die wir ebenfalls in die Chroot-Umgebung kopieren müssen (je Befehl eine Zeile):

```
# cp /usr/local/apache2/logs/access_log /chroot/httpd/usr/local/apache2/logs/  
# cp /usr/local/apache2/logs/error_log /chroot/httpd/usr/local/apache2/logs/  
# cp /usr/local/apache2/conf/mime.types /chroot/httpd/usr/local/apache2/conf/
```

Damit ist der eigentliche Aufbau der Chroot-Umgebung endlich abgeschlossen.

### 3.6.3 Konfiguration des Apache und Test der Chroot-Umgebung

Die Konfiguration des Apache ist recht einfach:

Wir müssen den Apache anweisen, dass dieser unter der manuell angelegten Benutzer- beziehungsweise Gruppenkennung *apache2* laufen soll. Dazu müssen Sie die Datei */usr/local/apache2/conf/httpd.conf* editieren und etwa in Zeile 266-267 die folgenden Anweisungen definieren:

```
User apache2  
Group apache2
```

Speichern Sie die Datei *httpd.conf* und kopieren Sie den Inhalt des Unterverzeichnisses *htdocs* aus dem normalen Dateisystem in die Chroot-Umgebung:

```
# cp -R /usr/local/apache2/htdocs /chroot/httpd/usr/local/apache2/
```

#### Hinweis

Alternativ können Sie selbstverständlich auch Ihre eigenen Dokumente in das Dokumentenverzeichnis *htdocs* kopieren.

Außerdem müssen Sie nach der Änderung der Datei *httpd.conf* diese in die Chroot-Umgebung kopieren (eine Zeile):

```
# cp /usr/local/apache2/conf/httpd.conf /chroot/httpd/usr/local/apache2/conf/
```

Schließlich benötigen wir alle Dateien aus dem Unterverzeichnis *bin* der lokalen Apache-Installation in der Chroot-Umgebung:

```
# cp /usr/local/apache2/bin/* /chroot/httpd/usr/local/apache2/bin/
```

Jetzt ist es an der Zeit, die Chroot-Umgebung das erste Mal zu testen. Geben Sie dazu folgenden Befehl ein, um den Apache zu starten:

```
# chroot /chroot/httpd/ /usr/local/apache2/bin/httpd
```

Dieser Befehl erstellt unterhalb des Verzeichnisses */chroot/httpd* eine Chroot-Umgebung und sperrt den Apache sozusagen in dieser Umgebung ein. Des Weiteren wird innerhalb der Chroot-Umgebung der Befehl */usr/local/apache2/bin/httpd* ausgeführt, um den Apache zu starten. Bitte beachten Sie, dass es sich dabei nicht um das reale Verzeichnis */usr/local/apache2* handelt, sondern um das in der Chroot-Umgebung vorhandene Verzeichnis */chroot/httpd/usr/local/apache2*. Das reale Verzeichnis */usr/local/apache2* wird nun eigentlich nicht mehr benötigt.

Rufen Sie jetzt die Startseite des Apache auf Ihrem lokalen Server unter `http://localhost` auf. Sie werden feststellen, dass sich freudestrahlend der Indianer meldet: Die Einrichtung der Chroot-Umgebung war erfolgreich!

### 3.6.4 Abschlussarbeiten

Zum Abschluss möchte ich ein simples Startskript vorstellen, welches unter Linux zum Start beziehungsweise Stopp des Apache 2 in der genannten Chroot-Umgebung dienen kann. Erstellen Sie dazu die Datei `/etc/init.d/apache2.sh` und schreiben Sie die folgenden Befehle in diese:

```
#!/bin/sh

# Start-/Stoppskript fuer den Apache 2
# in Chroot-Umgebung

chroot_bin=`which chroot`

case $1 in
start)

    echo "Starte den Apache 2 in Chroot-Umgebung..."
    $chroot_bin /chroot/httpd/ /usr/local/apache2/bin/httpd

;;

stop)

    echo "Beende den Apache 2."
    killall httpd

;;

restart)

    echo "Starte den Apache 2 neu..."
    $0 stop
    $0 start

;;

*)

    echo "Benutzung: $0 [start|stop|restart]"
    exit 1

esac
```

*Listing 3.1 Beispielskript zum Start/Stop des Apache 2 in der Chroot-Umgebung*

Speichern Sie die Datei und machen Sie diese ausführbar:

```
# chmod 755 /etc/init.d/apache2.sh
```

Mit Hilfe dieses Skriptes können Sie den Apache nun nach Belieben starten

```
# /etc/init.d/apache2.sh start
```

und stoppen:

```
# /etc/init.d/apache2.sh stop
```

Der Aufbau und die Konfiguration der Chroot-Umgebung sowie der Test des Apache 2 ist damit abgeschlossen. Dieses Kapitel zeigt relativ anschaulich, wie aufwändig es ist, den Apache in einer Chroot-Umgebung zu betreiben. Die Installation und Einbettung von externen Drittanbietermodulen (z.B. PHP) sei als Aufgabe dem interessierten Leser überlassen. Viel Spaß :-)

## 3.7 Metux MPM

Ein elementarer Vorteil des Apache 2 gegenüber dem Apache 1.3.x ist die Unterstützung verschiedener Laufzeitverhalten (Multi Processing Modules, MPM), die der Administrator bei der Installation des Apache auswählen kann. Mit Hilfe des Moduls *perchild* sollte es deshalb erstmalig möglich sein, virtuelle Server mit beliebigen Benutzer- und Gruppenkennungen auszuführen. Obwohl dies in punkto Sicherheit und Performance eine der vielversprechendsten Erweiterungen des Apache überhaupt darstellt, haben die Entwickler des Apache aus mir nicht bekannten Gründen die Weiterentwicklung des Moduls stark eingeschränkt bzw. teilweise ganz aufgegeben.

Der Deutsche Enrico Weigelt (<http://www.metux.de>) war mit der schleppenden Entwicklung des Moduls *perchild* unzufrieden und hat im Laufe dieses Jahres (2003) mit der Entwicklung eines eigenen Laufzeitmoduls namens *metuxmpm* (<http://www.metux.de/mpm/>) auf Basis des Moduls *perchild*, begonnen. Ziel dieses Moduls, welches sich momentan (Februar 2004) noch im Beta-Stadium befindet, ist die Bereitstellung eines stabilen Laufzeitverhaltens, welches die Ausführung von virtuellen Server mit jeweils eigenen Benutzerkennungen ermöglicht. Inzwischen wird Enrico Weigelt von einer Vielzahl von Freiwilligen unterstützt, die miteinander über eine Mailingliste ([metuxmpm@metux.de](mailto:metuxmpm@metux.de)) kommunizieren und das Modul fortlaufend weiterentwickeln. Mit der Veröffentlichung einer ersten stabilen Version wird im Lauf des Jahres (2004) zu rechnen sein, die dann auch offiziell Teil des Apache werden soll. Nachfolgend möchte ich die Installation und Konfiguration des Moduls vorstellen, wobei noch mal darauf hingewiesen sei, dass das Modul noch nicht als stabil gilt und deshalb nicht auf einem Produktivsystem eingesetzt werden sollte.

## Hinweis

Ich möchte mich an dieser Stelle nochmals ganz herzlich bei Enrico Weigelt (<http://www.metux.de>), dem Autor des Moduls `metuxmpm`, für die Unterstützung bei der Dokumentation dieses Moduls bedanken. Eine Version des Moduls für den Apache 2.0.48 war bei der Drucklegung des Buches noch nicht verfügbar.

### 3.7.1 Installation

## Hinweis

Falls Sie das Modul `metuxmpm` nicht manuell installieren möchten, finden Sie unter <ftp://ftp.suse.com/pub/people/poehl/apache2/8.2-i386> und <http://debian.helasnet> vorgefertigte Pakete für SuSE bzw. Debian.

Um das Modul `metuxmpm` manuell zu installieren, müssen Sie sich zunächst von <http://httpd.apache.org> die aktuell unterstützte Version 2.0.47 des Apache 2 herunterladen und mit Hilfe des Befehls

```
# tar xvzf httpd-2.0.47.tar.gz
```

entpacken. Hinweis: Wechseln Sie noch nicht in das neu erzeugte Verzeichnis `httpd-2.0.47`, da Sie zunächst einen Patch installieren müssen.

Da es momentan für das Modul noch keine offizielle Downloadmöglichkeit gibt, hat einer der Entwickler (Asbjorn Sannes) unter <http://www.sannes.org/metuxmpm/> einige Informationen und Downloads zusammengestellt. Von dort müssen Sie sich deshalb die aktuelle Version des Moduls als Patch (<http://www.sannes.org/metuxmpm/r7/httpd-2.0.47-metuxmpm-r7.diff>) herunterladen. Daraufhin können Sie die Software mit folgendem Befehl in die aktuelle Version des Apache (2.0.47) integrieren:

```
# patch -p0 < httpd-2.0.47-metuxmpm-r7.diff
```

Die Ausgabe des Befehls `patch` zeigt nun die Dateien an, die durch die Integration des Moduls `metuxmpm` verändert worden sind:

```
patching file httpd-2.0.47/server/mpm/config.m4
patching file httpd-2.0.47/server/mpm/experimental/metuxmpm/Makefile.in
patching file httpd-2.0.47/server/mpm/experimental/metuxmpm/config5.m4
patching file httpd-2.0.47/server/mpm/experimental/metuxmpm/metuxmpm.c
patching file httpd-2.0.47/server/mpm/experimental/metuxmpm/mpm.h
patching file httpd-2.0.47/server/mpm/experimental/metuxmpm/mpm_default.h
patching file httpd-2.0.47/src/lib/apr/network_io/unix/sockets.c
```

Wechseln Sie in das Quellverzeichnis des Apache (`cd httpd-2.0.47`) und sorgen Sie für die Aktualisierung einiger, zur Kompilierung des Apache, benötigter Dateien:

```
# ./buildconf
```

**Hinweis**

Dieser Befehl verwendet u. a. die Programme *libtool* und *autoconf*. Sollten Sie die Programme nicht installiert haben, müssen Sie diese mit Hilfe der Softwareverwaltung Ihrer Distribution (z. B. YaST) oder manuell nachinstallieren.

Die Ausführung des Befehls *buildconf* ist notwendig, da der vorhin installierte Patch des *metuxmpm* den Quellcode des Apache verändert hat und somit u. a. eine Aktualisierung des *configure*-Skriptes notwendig ist. Daraufhin lässt sich die erfolgreiche Integration des *metuxmpm* in den Quellcode des Apache mit folgendem Befehl überprüfen:

```
# ./configure --help | grep -B 1 metuxmpm
```

Dieser Befehl gibt eine Liste der zur Kompilierung des Apache zur Verfügung stehenden Optionen aus und durchsucht diese nach dem Schlüsselwort *metuxmpm*. Der Parameter *-B 1* führt zusätzlich dazu, dass die vor der eigentlichen Fundstelle stehende Zeile ebenfalls ausgegeben wird. Es erscheint folgende Ausgabe.

```
--with-mpm=MPM          Choose the process model for Apache to use.  
                        MPM={beos|worker|prefork|mpmt_os2|perchild|leader|threadpool|metuxmpm}
```

Erfreulicherweise wurde die Liste der verfügbaren Laufzeitverhalten (MPM) um das manuell installierte *metuxmpm* erweitert und die eigentliche Installation des Apache 2 kann beginnen. Dabei könnte eine minimale Installation des Apache wie folgt aussehen:

```
# ./configure --prefix=/usr/local/apache2 --with-mpm=metuxmpm
```

**Hinweis**

Dies ist wirklich eine sehr minimalistische Variante, um den Apache zu installieren bzw. die Installation vorzubereiten. Selbstverständlich können Sie diesem Befehl weitere Optionen, die Sie durch Eingabe des Befehls *./configure --help* erhalten, hinzufügen. Lesen Sie diesbezüglich bitte auch die entsprechenden Erläuterungen zur Installation des Apache 2 in diesem Buch.

Die eigentliche Kompilierung des Apache wird durch den Befehl

```
# make
```

gestartet. Sobald dieser abgeschlossen ist, können Sie die Installation nach */usr/local/apache2* durch Eingabe des Kommandos

```
# make install
```

beenden. Widmen wir uns nun der Konfiguration des Moduls:

## 3.7.2 Konfiguration

Generell werden durch das Modul *metuxmpm* die folgenden drei Konfigurationsanweisungen bereitgestellt:

### Multiplexer

Startet einen Multiplexer-Kindprozeß unter der angegebenen Benutzer- und Gruppenkennung.

<b>Konfigurationsanweisung:</b>	Multiplexer
<b>Syntax:</b>	Multiplexer <i>Benutzerkennung Gruppenkennung</i>
<b>Standardwert (Default):</b>	nicht definiert
<b>Enthalten in Modul:</b>	Metuxmpm
<b>Kontext:</b>	Server-Kontext
<b>Anweisung aktiv:</b>	nein

Mit Hilfe der *Multiplexer*-Anweisung definieren Sie einen so genannten Multiplexer-Kindprozess, der mit einer von Ihnen zu definierenden Benutzer- und Gruppenkennung gestartet werden soll, wobei momentan nur der Einsatz eines Multiplexers sinnvoll ist. Der Multiplexer untersucht die eingehenden Clientanfragen auf ihre *Host*-Header und ordnet diese dementsprechend den virtuellen Servern zu. Ein Beispiel:

```
Multiplexer www-data www-data
```

#### Hinweis

Die Benutzer und Gruppen, die Sie zur Ausführung der virtuellen Server benutzen möchten, müssen auf Ihrem System real existieren!

Diese Anweisung startet einen Multiplexer mit der Benutzerkennung *www-data* und der gleichnamigen Gruppenkennung. Generell kann als Multiplexer dieselbe Benutzer- und Gruppenkennung verwendet werden, mit der auch der Apache ausgeführt wird. Sie erhalten die Benutzerkennung durch Eingabe von:

```
# grep -E '^User' /usr/local/apache2/conf/httpd.conf
```

Bei dem von mir verwendeten System liefert dieser Befehl folgende Ausgabe, wobei nur die erste Zeile für uns relevant ist:

```
User www-data
UserDir public_html
```

#### Hinweis

Auf vielen System wird die Benutzerkennung *nobody* verwendet.

Auch die Gruppenkennung, mit der der Apache ausgeführt wird, lässt sich leicht herausfinden (vgl. *User-* und *Group-*Anweisung):

```
# grep -E '^Group' /usr/local/apache2/conf/httpd.conf
```

Dabei ist der Name der Gruppenkennung, mit der der Apache auf meinem Testsystem ausgeführt wird, mit dem bereits genannten Benutzernamen identisch:

```
Group www-data
```

### Hinweis

Auf vielen Systemen wird die Gruppe *nogroup* zur Ausführung des Apache verwendet.

Falls noch kein separater Benutzer zur Ausführung des Apache existiert oder Sie die virtuellen Server aus Sicherheitsgründen mit einer speziellen Benutzerkennung betreiben möchten, können Sie diesen selbstverständlich auch manuell anlegen. Dazu sollten Sie jedoch zunächst eine eigene Gruppe erzeugen:

```
# groupadd apache2
```

Nun können Sie der neu erstellten Gruppe einen Benutzer namens *apache2* hinzufügen:

```
# useradd -g apache2 -d /dev/null -s /bin/false apache2
```

Die entsprechende Multiplexer-Anweisung sieht infolgedessen wie folgt aus:

```
Multiplexer apache2 apache2
```

### Processor

Der Processor verarbeitet die eigentlichen Clientanfragen.

---

<b>Konfigurationsanweisung:</b>	Processor
<b>Syntax:</b>	Processor <i>Benutzerkennung Gruppenkennung</i>
<b>Standardwert (Default):</b>	nicht definiert
<b>Enthalten in Modul:</b>	Metuxmpm
<b>Kontext:</b>	Server-Kontext
<b>Anweisung aktiv:</b>	nein

---

Während der Multiplexer die Clientanfragen nur entgegennimmt, werden diese durch den so genannten *Processor* verarbeitet. Auch diese Anweisung erwartet die Angabe einer Benutzer- und Gruppenkennung, die zur Ausführung verwendet werden soll, wobei die definierte Gruppenkennung mit der der *Multiplexer-*Anweisung übereinstimmen muss:

```
Processor sebastian apache2
```

Aufgrund dieser Konfiguration können Sie einen virtuellen Server anlegen, der mit der Benutzerkennung *sebastian* und der Gruppenkennung *apache2* betrieben wird. Prinzipiell ist auch die Ausführung mehrerer virtueller Server mit ein- und derselben Benutzer- und Gruppenkennung möglich.

### AssignUserID

Bindet einen virtuellen Server an eine bestimmte Benutzer- und Gruppenkennung.

<b>Konfigurationsanweisung:</b>	AssignUserID
<b>Syntax:</b>	AssignUserID <i>Benutzername</i> <i>Gruppenname</i>
<b>Standardwert (Default):</b>	nicht definiert
<b>Enthalten in Modul:</b>	Metuxmpm
<b>Kontext:</b>	Server-Kontext, Virtual-Host Kontext
<b>Anweisung aktiv:</b>	nein

Mit der *AssignUserID*-Anweisung können Sie einem virtuellen Server eine Benutzer- und Gruppenkennung zuweisen, mit der ein virtueller Server ausgeführt werden soll, wobei diese vorher durch eine Multiplexer- bzw. eine Processor-Anweisung definiert worden sein müssen. Ein Beispiel:

```
AssignUserID sebastian apache2
```

Durch diese Anweisung wird der entsprechende virtuelle Server mit der Benutzerkennung *sebastian* und der Gruppenkennung *apache2* ausgeführt.

### 3.7.3 Praxisbeispiel

Da die korrekte Verwendung der oben genannten Konfigurationsanweisungen sicherlich etwas verwirrend ist, möchte ich ein kleines Beispiel für den Einsatz des *metuxmpm* vorstellen. Hinweis: Die nachfolgende Konfiguration verwendet die Benutzerkennung *sebastian* und die Gruppenkennung *apache2*. Außerdem wird als Installationsverzeichnis des Apache 2 das Verzeichnis */usr/local/apache2* verwendet. Sollten diese Werte auf dem von Ihnen verwendeten System variieren, müssen Sie diese entsprechend anpassen.

Erstellen Sie zunächst testweise zwei neue Domains namens *example.com* und *example2.com*, indem Sie die entsprechende Zonen in Ihrem Nameserver anlegen. Sofern Sie keinen Zugriff auf einen Nameserver haben, können Sie diese Einträge temporär auch in der Datei */etc/hosts* (Unix/Linux) bzw. *C:\WinNT\System32\Drivers\Etc\hosts* (Windows 2000/XP) vornehmen. Die entsprechenden Einträge sehen beispielsweise wie folgt aus:

```
192.168.0.6 www.example.com example.com
192.168.0.6 www.example2.com example2.com
```

**Hinweis**

Die hier verwendete IP-Adresse 192.168.0.6 ist gemäß der von Ihnen verwendeten Adresse anzupassen.

Speichern Sie die Datei und fügen Sie der zentralen Konfigurationsdatei *httpd.conf* des Apache folgende Anweisungen hinzu:

```
Multiplexer apache2 apache2
Processor sebastian apache2
```

Speichern Sie die Datei und fügen Sie der zentralen Konfigurationsdatei *httpd.conf* des Apache folgende Anweisungen hinzu:

```
NameVirtualHost *
KeepAlive Off
```

Entfernen Sie eine eventuell vorhandene Auskommentierung (*>#<<*) und fügen Sie die folgenden Zeilen der Datei hinzu:

```
<VirtualHost *>
AssignUserID sebastian apache2
Servername www.example.com
DocumentRoot /usr/local/apache2/htdocs
ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"
</VirtualHost>

<VirtualHost *>
AssignUserID sebastian apache2
Servername www.example2.com
DocumentRoot /srv/www/htdocs
</VirtualHost>
```

**Hinweis**

Diese Konfiguration ist unvollständig, verkürzt und für die Praxis nicht sonderlich sinnvoll. Sie dient lediglich der Veranschaulichung.

Die oben genannte Konfiguration erstellt zunächst einen Multiplexer, der die Anfragen der Clients entgegennimmt und einen Processor. Dieser Processor dient zur Ausführung zweier virtueller Server (*www.example.com* und *www.example2.com*), wobei die Zuweisung des zuständigen Processors an den jeweiligen virtuellen Server mit Hilfe der *AssignUserID*-Anweisung erfolgt. Ermöglichen Sie nun außerdem die Ausführung von CGI-Skripten im Verzeichnis *cgi-bin* des ersten virtuellen Servers, indem Sie ca. in Zeile 608 der Datei *httpd.conf* die Option *ExecCGI* einfügen, so dass die entsprechende Stelle in der Datei wie folgt aussieht:

```
<Directory "/usr/local/apache2/cgi-bin">
AllowOverride None
Options ExecCGI
Order allow,deny
Allow from all
</Directory>
```

Speichern Sie die Änderungen an der Datei *httpd.conf* und starten Sie den Apache neu. Erstellen Sie zusätzlich im Verzeichnis */usr/local/apache2/cgi-bin* eine Datei namens *test.pl* mit folgendem Inhalt:

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "<html>";
print "<h2>Test von metuxmpm</h2>";

print "Hallo Welt<br>\n";

print "Meine Benutzer- und Gruppenkennung lautet: <br><b>\n";
system("/usr/bin/id");

print "</b>\n<br>";

print "\n Ich bin der Benutzer: <br><b>\n";
system("/usr/bin/whoami");

print "</b>\n<br>";

print "Fertig :-)<br>\n";

print "</html>";
```

Speichern Sie auch diese Datei und geben Sie ihr die Ausführberechtigung:

```
# chmod +x /usr/local/apache2/cgi-bin/test.pl
```

Starten Sie den Apache neu und rufen Sie die Datei unter der Adresse *http://www.example.com/cgi-bin/test.pl* auf und Sie werden feststellen, dass der virtuelle Server für die Domain *www.example.com* tatsächlich unter einer eigenen Benutzerkennung ausgeführt wird:

Demnach verrichtet das Modul *metuxmpm* erwartungsgemäß korrekt seinen Dienst, quod erat demonstrandum :-)

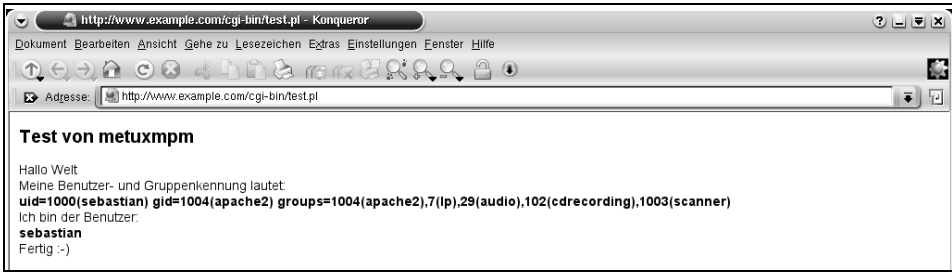


Abbildung 3.1 Die Testseite zeigt die ordnungsgemäße Funktionsweise des Moduls `metuxmpm`

## 3.8 Updates

Von Zeit zu Zeit bringt die Apache Software Foundation neuere Versionen des Apache Webservers heraus. Diese bringen teilweise neue Funktionen, beheben kritische Fehler oder Sicherheitsprobleme (Stichwort: *chunk encoding*, vgl. <http://www.cert.org/advisories/CA-2002-17.html> sowie [http://httpd.apache.org/info/security\\_bulletin\\_20020809a.txt](http://httpd.apache.org/info/security_bulletin_20020809a.txt), kritischer Fehler in der Win32, OS/2 und Netware-Variante des Apache, behoben in 2.0.42) und es ist gerade im Falle der Veröffentlichung einer kritischen Sicherheitslücke immer sinnvoll, ein Update zu machen. Zwar gilt für Updates die Systemadministrator-Maxime *Never change a running system*, aber falls es sich um ein sicherheitskritisches Update handelt, sollten Sie es in jedem Fall einspielen. Der Ablauf eines Updates deckt sich mit der Neuinstallation des Apache, allerdings sollten Sie vor dem Ausführen des Befehls `make install` den alten Server anhalten. Sie sollten außerdem vor dem Update eine Sicherung Ihrer alten Installation vornehmen, damit Sie diese, im Falle eines unerwartet auftretenden Problems, sicher wieder zurückspielen können. Wenn Sie eine vorkompilierte Version des Apache benutzt haben, können Sie die alte Version einfach löschen und die Neue einspielen (Konfigurationsdateien vorher sichern!). Sollten Sie eine durch Ihre jeweilige Distribution (SuSE, RedHat, Debian etc.) bereitgestellte Version des Apache benutzen, kann ein Update unter Umständen problematisch sein, denn nicht immer werden rasch aktualisierte Pakete durch die Distributoren bereitgestellt, d. h., es kann teilweise nötig sein (z. B. im Falle eines Sicherheitsproblems), das Update des Apache manuell durchzuführen. Leider ist es mit einigem Aufwand verbunden, die durch den Bereitsteller der zuvor verwendeten Version des Apache (z. B. SuSE) verwendeten Konfigurationsoptionen herauszubekommen, so dass man diese bei einem Update selber nutzen kann. Verwendet man die ursprünglich genutzten Optionen nicht, kann es passieren, dass man unter Umständen andere Verzeichnisse benutzt, als es ursprünglich der Fall war und somit das Update nicht vollständig und erfolgreich durchgeführt werden kann. Wenn Ihnen jedoch die ursprünglich verwendeten Konfigurationsoptionen vorliegen, sollte ein Update auch ohne Verwendung eines durch den jeweiligen Distributor (Hersteller, z. B. SuSE) zur Verfügung gestellten Paketes problemlos möglich sein.

Wenn Sie möchten, können Sie sogar die alte Konfiguration der vorherigen Version übernehmen. Kopieren Sie dazu die Datei `config.nice` aus dem Quellverzeichnis der

alten Version in das Quellverzeichnis der neueren Version und führen Sie diese Datei aus. Aus diesem Grunde sollten Sie bei der Installation des Apache zumindest die Datei *config.nice* sichern, damit Sie im Falle eines Updates die alte Konfiguration ohne Probleme wieder einspielen können. Im Apache 1.3.x hieß diese Datei übrigens *config.status* (diese Datei existiert zwar weiterhin, hat aber eine völlig neue Bedeutung bekommen). Für den von mir verwendeten Installationsaufruf der Version 2.0.47 sieht die Datei *config.nice* wie folgt aus:

```
#!/bin/sh
#
# Created by configure

"./configure" \
"--enable-layout=Apache" \
"$@"
```

Ich möchte Ihnen den Ablauf eines Updates anhand eines konkreten Beispiels von der Version 2.0.47 auf 2.0.48 vorstellen. In meinem Heimatverzeichnis */home/sebastian* habe ich die alte Version (Verzeichnisname: */home/sebastian/httpd-2.0.47*) komplett gesichert und die neue Version frisch vorliegen (Verzeichnis: */home/sebastian/httpd-2.0.48*). Die Installation der alten Version ist seinerzeit nach */usr/local/apache2* erfolgt, die Konfigurationsoptionen stehen in der Datei */home/sebastian/httpd-2.0.47/config.nice*. Ich kopiere also die Datei *config.nice* in das Verzeichnis der neuen Version:

```
/home/sebastian # cp httpd-2.0.47/config.nice ./httpd-2.0.48/
```

Ich wechsele in das Verzeichnis der Version 2.0.48 und führe das Konfigurationskript, welches ich ja bereits in der Version 2.0.47 benutzt habe, aus:

```
/home/sebastian/httpd-2.0.48 # ./config.nice
```

Die Konfiguration der neuen Version beginnt und sobald diese abgeschlossen ist, kann die Software auch schon kompiliert werden:

```
/home/sebastian/httpd-2.0.48 # make
```

Nun beende ich die alte Version des Häuptlings und spiele die neue Version ein:

```
/home/sebastian/httpd-2.0.48 # killall httpd
/home/sebastian/httpd-2.0.48 # make install
```

Keine Angst, es werden dabei keine Konfigurationsdateien der alten Version überschrieben. Der neue Apache muss sich nun ohne Probleme starten lassen:

```
# /usr/local/apache2/bin/httpd
```

Das eigentliche Update ist damit schon abgeschlossen, durch folgenden Aufruf kann man die korrekte Installation der neuen Version überprüfen:

```
# /usr/local/apache2/bin/httpd -v
```

Der Befehl sollte die Versionsnummer und das Kompilierungsdatum des Apache ausgeben:

```
Server version: Apache/2.0.48
Server built:   Oct 05 2002 11:15:30
```

Interessant ist, dass man dem Skript *config.nice* auch noch eigene Parameter mit angeben kann, die an den alten Aufruf des *configure*-Skriptes angehängt werden. In der Datei *config.nice* steht eine eventuell kryptisch anmutende Zeichenkette *\$@*, die dafür sorgt, dass der Parameter, der an das aufgerufene Shellskript *config.nice* übergeben worden ist, an den alten Aufruf von *configure* angehängt wird. Somit lassen sich nachträglich Änderungen an den ursprünglich verwendeten Konfigurationsoptionen vornehmen. Wenn Sie beispielsweise das Modul *mod\_imap* in Ihrer ursprünglichen Serverkonfiguration verwendet haben, dieses jedoch jetzt nicht mehr nutzen möchten, können Sie das Skript *config.nice* wie folgt aufrufen:

```
# ./config.nice --disable-imap
```

Es stehen Ihnen beim Aufruf von *config.nice* prinzipiell alle Optionen zur Verfügung, die es bei der Neuinstallation des Apache gibt (siehe vorherige Kapitel).

Und nicht vergessen: Das nächste Update ist immer das Schwerste :-)