Apache – eine kurze Einführung

Jörg Lehmann und Alexander Rahmeier

5. Mai 1998

1 Einleitung

Die Aufgabe eines WWW–Servers besteht im wesentlichen darin, auf Anfrage eines Clients, der eine URL (*Universal Resource Locator*, z.B. `http://www.augsburg-live.de/luga/`) übergibt, diese entweder in einen Dateinamen umzuwandeln und die entsprechende Datei (z.B. ein HTML–Dokument[1]) zurückzusenden oder ein passendes Programm, ein sogenanntes CGI–Skript[2], auszuführen, dessen Ausgabe dann an den Client übertragen wird.

Der erste Webserver überhaupt wurde am europäischen Zentrum für Hochenergiephysik CERN in Genf entwickelt. Wichtige Innovationen im Softwarebereich können also durchaus auch aus Europa kommen, und es sage keiner mehr, daß Grundlagenforschung nur viel Geld koste und im Endeffekt doch nichts bringe...

Der Webserver Apache selbst ist eine Weiterentwicklung des NCSA–httpd des National Center for Supercomputing Applications, ist also quasi eine „gepatchte" Version dieses Programmes, woher auch der Name rühren soll. Obwohl Apache kostenlos zur Verfügung gestellt wird, übernimmt seine Entwicklung eine ausgewählte Gruppe von Programmierern, die Apache Group.

Apache ist mit einem Anteil von fast 50% inzwischen der meistverwendete WWW–Server im Internet, wobei er für verschiedene Plattformen (diverse Unixe inklusive Linux, Windows u.v.m.) zur Verfügung steht.

2 Konfiguration

2.1 Allgemeines

Die Konfiguration des Apache gliedert sich in zwei Abschnitte. Zum einen muß man aufgrund der modularen Struktur des Apache vor seiner Übersetzung festlegen, welche Module man einbinden will. Dies ist in Abschnitt 2.2 näher beschrieben. Die notwendigen Verzeichnisse sowie die richtigen Zugriffsrechte werden in Abschnitt 2.3 erläutert. Die eigentliche Einrichtung erfolgt dann mittels einiger Konfigurationsdateien, die in Abschnitt 2.4 behandelt werden.

2.2 Übersetzung

Nachdem man die aktuellen Quellen des Apache von CD installiert oder von [1] bzw. diversen Mirrors heruntergeladen und entpackt hat (Betaversionen sind durch ein „b" im Namen besonders gekennzeichnet), sollte man sich in das Unterverzeichnis `src` des Apache–Pakets begeben und dort die Datei `Configuration` editieren. Falls diese nicht existiert, was nur bei Betaversionen der Fall sein sollte, muß man zunächst `Configuration.tmpl` nach `Configuration` kopieren.

[1] Hinweis: Wenn hier und im folgenden von HTML–Dokumenten die Rede ist, dann ist dies immer nur in Abgrenzung zu CGI–Skripten zu verstehen, d.h. es sind damit auch alle anderen Dateien (z.B. GIF–Bilder, Java–Programme usw.) gemeint, die der Server zur Verfügung stellen soll.

[2] CGI bedeutet *Common Gateway Interface* und stellt eine Schnittstelle für Programme dar, die auf Anfrage eines Clients auf dem Server ausgeführt werden.

Die Datei `Configuration` enthält fünf verschiedene Arten von Zeilen:

- Kommentare, die mit einem `#`–Zeichen beginnen;
- Regeln, gekennzeichnet durch das Schlüsselwort `Rule`, welche die Erzeugung des Makefiles (s.u.) steuern;
- Kommandos, die nicht besonders gekennzeichnet sind und die in das Makefile eingefügt werden;
- Modulauswahlzeilen, die mit `Module` beginnen und in denen die einzukompilierenden Module angegeben sind;
- optionale Modulauswahlzeilen beginnend mit `%Module` für Module, die zwar mit einkompiliert, aber zunächst nicht aktiviert werden sollen.

Wichtig sind vor allem die Modulauswahlzeilen, da mit ihnen der Funktionsumfang des Apache festgelegt wird. Dabei kann man zunächst durchaus großzügig bei der Auswahl der einzukompilierenden Module vorgehen, da eine große Zahl von eingebundenen Modulen außer einer größeren Binärdatei keine wahrnehmbaren Performanceeinbußen verursachen sollte. Die Reihenfolge, in der die Module angegeben werden, bestimmt die Präzedenz, mit der diese später ausgeführt werden, wobei das zuletzt angegebene Modul die höchste Priorität erhält. Die Proxyfunktionalität des Apache kann beispielsweise bei Bedarf durch Angabe der Zeile

```
Module proxy_module        modules/proxy/libproxy.a
```

aktiviert werden.

Nachdem man die gewünschten Module eingetragen hat, kann man mit

```
./Configure
```

aus `Makefile.tmpl` das eigentliche `Makefile` erstellen lassen. Die Übersetzung erfolgt wie gewohnt mit

```
make
```

und sollte als Resultat die ausführbare Datei `httpd` liefern, die man anschließend z.B. nach `/usr/local/bin` kopieren kann.

2.3 Verzeichnisse und Zugriffsrechte

Zur weiteren Einrichtung des Apache muß man nun einige Verzeichnisse für die Konfigurations- und Logdateien und natürlich auch für die Webseiten anlegen. Im weiteren verwenden wir folgende Verzeichnisstruktur:

`/usr/local/etc/httpd`	Wurzelverzeichnis der Konfigurations- und Logdateien
`/usr/local/etc/httpd/conf`	Verzeichnis für Konfigurationsdateien
`/usr/local/etc/httpd/log`	Verzeichnis für Logdateien
`/var/httpd`	Wurzelverzeichnis der HTML–Dokumente und CGI–Skripten
`/var/httpd/htdocs`	Wurzelverzeichnis der HTML–Dokumente
`/var/httpd/cgi-bin`	Wurzelverzeichnis der CGI–Skripten
`/var/httpd/icons`	Einige Icons für die Darstellung von Indexseiten

Alle Verzeichnisse und die darin enthaltenen Dateien sollten dabei `root` und der Gruppe `root` gehören, zumindest falls der Verwalter der WWW–Seiten zugleich Systemverwalterrechte besitzt. Sollte dies nicht der Fall sein, muß man sich von Fall zu Fall eine etwas andere Regelung überlegen.

Nun benötigt man noch einen Benutzer und eine Gruppe, mit deren Rechten schließlich der `httpd` laufen soll.[3] In unserem Fall haben wir hierfür einen Benutzer `wwwrun` und die Gruppe `www` eingerichtet.[4] Wichtig ist hierbei nur, daß in `/var/httpd` und Unterverzeichnissen keine Datei und kein Verzeichnis `wwwrun` oder `www` gehören darf und daß nur der Inhaber (und evtl. auch die Gruppe) einer Datei bzw. eines Verzeichnisses Schreibrechte besitzt. Auf diese Weise kann verhindert werden, daß ein möglicher Eindringling, der sich auf irgendeine Weise die Rechte des Apache–Prozesses verschafft hat, an den WWW–Seiten oder, was noch schlimmer wäre, an den CGI–Skripten Veränderungen vornehmen kann. Für eine HTML–Datei würden in unserem Fall also typischerweise folgende Rechte gesetzt werden:

```
-rw-r--r--   1 root     root          516 May  1  1997 index.html
```

Die Verzeichnisse, in denen die HTML–Dateien bzw. die CGI–Skripten liegen, und alle im Pfad darüberliegenden Verzeichnisse müssen natürlich „andere“ Benutzer betreten (Recht `x`) können, also beispielsweise

```
drwxr-xr-x   3 root     root         1024 May  1 14:41 htdocs/
```

Hinweis: Das hier noch gesetzte Leserecht für andere Benutzer erlaubt bekanntlicherweise das Lesen des Inhaltsverzeichnisses des entsprechenden Unterverzeichnisses, was nicht notwendig und zumeist auch nicht erwünscht ist. Vergibt man nämlich kein Leserecht, so ist ein Betrachter der Webseiten an die von diesen festgelegte Struktur gebunden und kann nicht (bzw. nur durch Raten des Dateinamens) beliebige Dokumente laden.

2.4 Konfigurationsdateien

Die Konfiguration geschieht mit Hilfe von drei, im Verzeichnis `/usr/local/etc/httpd/conf` abgelegten Dateien.[5] Im den folgenden Abschnitten sind diese Dateien zusammen mit den wichtigsten Anweisungen anhand einer Beispielkonfiguration beschrieben. Bessere, vor allem ausführlich kommentierte Vorlagen für die Einrichtung des Apache findet man im Unterverzeichnis `conf` der Apache–Quellen unter den Namen `httpd.conf-dist`, `access.conf-dist` und `srm.conf-dist`. Eine Dokumentation aller Anweisungen im HTML–Format findet man in [3] oder in gedruckter Form in [2]. Wie dort haben wir bei der Beschreibung der Anweisungen folgendes Format gewählt:

Anweisung

Syntax:	`Beispiel` *`Parameter`*
Voreinstellung:	`Beispiel Standardparameter`
Kontext:	`Wo verwendet`

Unter der Rubrik Kontext ist dabei angegeben, an welchen Stellen innerhalb der Konfigurationsdateien die jeweilige Anweisung verwendet werden darf, wobei im Rahmen dieses Vortrages nur die Kontexte `Serverkonfiguration`, `Verzeichnis`, `virtuelle Hosts` und `.htaccess` auftauchen. Wichtig sind davon im weiteren auch nur die ersten beiden, deren Bedeutung dann auch später klar werden sollte.

2.4.1 Die Datei `httpd.conf`

Mittels dieser Datei werden generelle Eigenschaften des Servers, wie z.B. die Portnummer und die Wurzelverzeichnisse der Konfigurations- und Logdateien, festgelegt. In unserer Beispielkonfiguration sieht das wie folgt aus:

[3] Eigentlich laufen nur die vom ursprünglichen `httpd` gestarteten Tochterprozesse unter dieser Benutzer- und Gruppenkennung.

[4] Eine andere gängige Praxis ist es, den standardmäßig eingerichteten User `nobody.nogroup` (UID: -1, GID: -1) zu verwenden oder aber für `wwwrun.www` die gleichen IDs zu vergeben wie für `nobody.nogroup`.

[5] Man kann auch die gesamte Konfiguration in einer Datei zusammenfassen, vgl. [2]; wir halten uns aber an die Vorgaben der Apache–Gruppe, die drei Dateien vorsehen.

```
# /usr/local/etc/httpd/conf/httpd.conf
ServerType standalone
Port 80
User wwwrun
Group www
ServerAdmin www-admin@bubble.bobble.org
ServerRoot /usr/local/etc/httpd
ErrorLog logs/error_log
TransferLog logs/access_log
PidFile logs/httpd.pid
ServerName www.bubble.bobble.org
MaxRequestsPerChild 30
```

Nun die Erläuterung der verwendeten Anweisungen:

ServerType

Syntax:	`ServerType` *`Typ`*
Voreinstellung:	`ServerType standalone`
Kontext:	`Serverkonfiguration`

Mittels der `ServerType`-Anweisung kann man festlegen, ob der Server als Hintergrundprozeß laufen soll (*Typ* =`standalone`) oder erst auf Anfrage eines Clients via `inetd` gestartet wird (*Typ* =`inetd`). Im Normalfall sollte aus Performancegründen ein als Hintergrundprozeß laufender Webserver bevorzugt und nur bei knappen Ressourcen auf den Start durch den `inetd` umgestellt werden, zumal laut [2] die Unterstützung dieser Betriebsart auslaufen wird.

Port

Syntax:	`Port` *`Portnummer`*
Voreinstellung:	`Port 80`
Kontext:	`Serverkonfiguration`

Gibt die Portnummer an, auf welcher der Server auf die Anfragen der Clients wartet. Dies ist im Normalfall natürlich Port 80, der Standardport für das HTTP–Protokoll.

User

Syntax:	`User` *`Benutzername`* oder `User #`*`Benutzer-ID`*
Voreinstellung:	`User #-1`
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Group

Syntax:	`Group` *`Gruppenname`* oder `Group #`*`Gruppen-ID`*
Voreinstellung:	`Group #-1`
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Legt Benutzer und Gruppe fest, unter denen der Webserver Anfragen von Clients beantwortet. Dazu muß allerdings der ursprünglich gestartete `httpd` mit `root`-Rechten gestartet werden. Zur Sicherheitsrelevanz dieser Einstellungen: vgl. Abschnitt 2.3 und [2, 3].

ServerAdmin

Syntax:	`ServerAdmin` *`e-Mail-Adresse`*
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Gibt die e–Mail–Adresse an, die der Server in Fehlermeldungen einfügt, die er an die Clients zurücksendet.

ServerRoot

Syntax:	`ServerRoot` *`Verzeichnisname`*
Voreinstellung:	`ServerRoot /usr/local/etc/httpd`
Kontext:	`Serverkonfiguration`

Gibt das Wurzelverzeichnis für die Konfigurations- und Logdateien des Webservers an, auf das sich dann relative Pfadnamen solcher Dateien beziehen. Die Kommandozeilenoption `-d` des `httpd` wird durch diese Anweisung überschrieben, kann aber natürlich verwendet werden, um das Verzeichnis der Konfigurationsdateien und insbesondere der Datei `httpd.conf` beim Start des Servers anzugeben.

ErrorLog

Syntax:	`ErrorLog` *`Dateiname`*
Voreinstellung:	`ErrorLog logs/error_log`
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Gibt den Namen der Datei an, die zum Protokollieren der Fehlermeldungen des Servers verwendet wird. Wenn der Dateiname nicht mit einem `/` beginnt, wird er als relativ zu `ServerRoot` interpretiert.

TransferLog

Syntax:	`TransferLog` *`Dateiname`* oder `Transferlog |`*`Programm`*
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Diese Anweisung ist nur verfügbar, wenn das Modul `mod_log_config` einkompiliert ist, was ab Apache 1.2 standardmäßig der Fall sein sollte. Dann kann als Option entweder eine Datei angegeben werden, in der die Zugriffe auf den Webserver protokolliert werden (zur Syntax des Dateinamen: vgl. `ErrorLog`) oder ein Pipe–Zeichen | gefolgt von einem Programmnamen. In letzterem Fall wird dem angegebenen Programm auf der Standardeingabe die Logging–Information zur Verfügung gestellt. Dabei sollte man allerdings beachten, daß dieses Programm mit den Rechten desjenigen Benutzers läuft, der den `httpd` gestartet hat, also im allgemeinen `root`.

PidFile

Syntax:	`PidFile` *`Dateiname`*
Voreinstellung:	`PidFile logs/httpd.pid`
Kontext:	`Serverkonfiguration`

Gibt die Datei an, in der die Prozeß–ID des laufenden `httpd` gespeichert wird. Diese Option ist nur im `standalone`-Modus des Servers wirksam. Bezüglich der Angabe des Dateinamens gilt natürlich wieder die unter der `ErrorLog`-Anweisung beschriebene Regel.

ServerName

Syntax:	`ServerName` *`FQDN`*
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Hiermit kann der Name des Webservers als *fully qualified domain name* angegeben werden, falls die standardmäßig vorgenommene Auflösung des Namens aus der IP–Adresse des Servers nicht das gewünschte Resultat liefert. Dies kann z.B. dann der Fall sein, wenn zwei Namen für diese IP–Adresse vergeben sind und der gewünschte nicht der kanonische ist (z.B. `www.bubble.bobble.org` statt `server.bubble.bobble.org`). Benötigt wird dieser Name allerdings nur, falls man „umgeleitete URLs“ verwendet, auf die hier nicht näher eingegangen werden kann.

MaxRequestsPerChild

Syntax:	`MaxRequestsPerChild` *Zahl*
Voreinstellung:	`MaxRequestsPerChild 0`
Kontext:	`Serverkonfiguration`

Als Beispiel für die vielen Anweisungen, die dem Feintuning des Servers dienen, sei hier `MaxRequestsPerChild` angegeben, da eine Veränderung der Standardeinstellung hier besonders sinnvoll erscheint. Setzt man nämlich `MaxRequestsPerChild` auf einen Wert größer Null, so wird ein vom `httpd` gestarteter Tochterprozeß nach der angegebenen Anzahl von Anfragen beendet und neu gestartet. Dies hat vor allem den Vorteil, daß durch Speicherlecks – solche können durch Fehler im Apache selbst oder in den von ihm verwendeten Bibliotheken durchaus entstehen – eventuell verloren gegangener Speicher wieder freigegeben wird.

2.4.2 Die Datei `srm.conf`

In dieser Datei werden u.a. das Wurzelverzeichnis der HTML–Dokumente, die automatische Generierung von Indexseiten durch den Server u.v.m. festgelegt. Für unserer Beispielkonfiguration sieht das auszugsweise folgendermaßen aus:

```
# /usr/local/etc/httpd/conf/srm.conf
DocumentRoot /var/httpd/htdocs
UserDir public_html

DirectoryIndex index.html
IndexOptions FancyIndexing
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
#...
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/folder.gif ^^DIRECTORY^^
#...
DefaultIcon /icons/unknown.gif
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
ReadmeName README
HeaderName HEADER

Alias /icons/ /var/httpd/icons/
ScriptAlias /cgi-bin/ /var/httpd/cgi-bin/
```

Die wichtigsten der hier auftretenden Anweisungen sind:

DocumentRoot

Syntax:	`DocumentRoot` *Verzeichnisname*
Voreinstellung:	`DocumentRoot /usr/local/etc/httpd/htdocs`
Kontext:	`Serverkonfiguration, virtuelle Hosts`

`Verzeichnisname` gibt das Wurzelverzeichnis der vom Server bereitgestellten Dateien, d.h. der HTML–Dokumente und der CGI–Skripten, an. Wird eine URL von einem Client nachgefragt, so wird der Pfadname aus der URL an diesen Verzeichnisnamen angehängt, sofern kein Kommando wie `Alias` (s.u.) zutrifft. Wegen eines Fehlers im Modul `mod_dir` sollte der angegebene Verzeichnisname *keinen* abschließenden Schrägstrich / besitzen.

Hinweis: CGI–Skripten sollten nicht im für die HTML–Dokumente verwendeten Bereich des Verzeichnisbaumes gespeichert werden, um ein Lesen des Programmkodes durch Unbefugte zu verhindern. So verringert man nämlich die Gefahr, daß potentielle Eindringlinge Fehler in den CGI–Skripten ausmachen. Deshalb sollte man mittels der `ScriptAlias`-Anweisung (s.u.) ein separates Wurzelverzeichnis für CGI–Skripten angeben.

UserDir

Syntax:	`UserDir` *`Verzeichnisname`*
Voreinstellung:	`UserDir public_html`
Kontext:	`Serverkonfiguration, virtuelle Hosts`

Mit dieser Anweisung des Moduls `mod_userdir` kann man den Verzeichnisnamen angeben, der bei einem Zugriff auf persönliche Dokumente eines Benutzers mit der Tilde–Syntax verwendet wird. Die Angabe `disabled` schaltet einen solchen Zugriff ab. Das voreinstellte Verhalten bewirkt beispielsweise, daß ein Zugriff auf die URL `http://www.bubble.bobble.org/~user/` in `/home/user/public_html/index.html` übersetzt wird. Der User ist selbst verantwortlich, daß dieses Verzeichnis entsprechend angelegt wird. Die Verwendung von absoluten Pfadangaben oder dem Wildcard `*` in *`Verzeichnisname`* ermöglicht außerdem die Ersetzung von `/home` durch einen anderen Pfad [3].
Hinweis: Mit dieser Anweisung sollte man sehr vorsichtig umgehen, d.h. man sollte `UserDir` auf `disabled` setzen, sofern man nicht wirklich genau weiß, was man tut. Eine Gefahr liegt beispielsweise darin, daß Benutzer in ihrem persönlichem HTML–Verzeichnis einen symbolischen Link auf ein Verzeichnis anlegen, das außerhalb des Bereiches für HTML–Dokumente liegt (z.B. das Wurzelverzeichnis `/` des Rechners). In diesem Fall wäre es einem Client natürlich möglich, auf praktisch alle Dateien des Rechners zuzugreifen, es sei denn, man schützt diese vor unberechtigten Zugriffen, wie wir es in unserer `access.conf`–Datei getan haben (s.u.). Deshalb an dieser Stelle nochmals ein Hinweis auf [3], insbesondere auf die dortige Seite mit Sicherheitshinweisen.

Die folgenden Anweisungen beschäftigen sich alle mit der Darstellung des Verzeichnisinhaltes, die notwendig wird, falls dem Server vom Client eine mit einem Schrägstrich `/` endende URL übergeben wird. Benötigt wird dazu das standardmäßig einkompilierte Modul `mod_dir`. Wegen der Vielzahl der `AddIcon`–Anweisungen empfiehlt sich als Vorlage für die Einrichtung hier auf jeden Fall die mitgelieferte Datei `srm.conf-dist`.

DirectoryIndex

Syntax:	`DirectoryIndex` *`Lokale URL Lokale URL...`*
Voreinstellung:	`DirectoryIndex index.html`
Kontext:	`Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Diese Anweisung des Moduls `mod_dir` gibt eine Liste der lokalen URLs der Dokumente bzw. Skripten an, die der Server übergeben bzw. ausführen soll, falls vom Client eine mit einem Schrägstrich `/` endende URL angefragt wird. Falls mehrere URLs angegeben werden, verwendet der Server die erste existierende. Sollte keine der angegebenen URLs vorhanden sein, wird ein formatierte Ausgabe des Verzeichnisinhaltes generiert, sofern die Option `Indexes` des `Options`-Kommandos gesetzt wurde (s.u.). Hierzu muß natürlich das entsprechende Verzeichnis lesbar, nicht nur ausführbar sein.

IndexOptions

Syntax:	`IndexOptions` *`Option Option...`*
Kontext:	`Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Diese Anweisung kontrolliert das Verhalten des Verzeichnisindexmoduls `mod_dir`. In der Beispielkonfiguration ist mittels der Option `FancyIndexing` eine mit Icons verzierte Darstellung des Verzeichnisinhaltes eingeschaltet worden. Auch hierzu muß natürlich das Verzeichnis lesbar sein. Weitere Optionen kann man [2, 3] entnehmen.

AddIcon

Syntax: `AddIcon` *`Icon Name Name...`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

AddIconByEncoding

Syntax: `AddIconByEncoding` *`Icon MIME-Encoding MIME-Encoding...`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

AddIconByType

Syntax: `AddIconByEncoding` *`Icon MIME-Type MIME-Type...`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

DefaultIcon

Syntax: `DefaultIcon` *`URL`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Mit diesen Anweisungen lassen sich die Icons angeben, die das Modul `mod_dir` bei eingeschaltetem `FancyIndexing` für die verschiedenen Dateitypen verwendet werden. Dabei spielt die Reihenfolge der Angaben durchaus eine Rolle; das erste passende Icon wird verwendet. *`Icon`* steht dabei entweder für eine URL oder für `(`*`Text,URL`*`)`, wobei *`Text`* einen Text angibt, der bei nicht graphikfähigen Browser an der Stelle des Icons angezeigt wird. Beginnt die URL nicht mit einem Schrägstrich `/`, so bezieht sie sich auf das mit `DocumentRoot` gesetzte Verzeichnis, ansonsten auf das Verzeichnis, von dem das Inhaltsverzeichnis erstellt werden soll. In unserem Fall wird allerdings später die `Alias`-Anweisung dazu verwendet, um eine andere Lage des Icon–Verzeichnis `icons` anzugeben.
`Name` steht dabei entweder für `^^DIRECTORY^^`, `^^BLANKICON^^`, eine Dateiendung, einen Ausdruck mit Wildcards oder einen (unvollständigen) Dateinamen. Die Optionen *`MIME-Encoding`* bzw. *`MIME-Type`* sind wohl selbsterklärend für denjenigen, der sich mit dem MIME–Konzept auskennt. Benötigt wird bei ihrer Verwendung allerdings eine `mime.types`-Datei.
Via `DefaultIcon` läßt sich eine URL auf ein Icon angeben, das für alle Dateien verwendet werden, die von den `AddIcon`-Anweisungen nicht erfaßt wurden.

IndexIgnore

Syntax: `IndexIgnore` *`Datei Datei...`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Gibt eine Liste von Dateien an, die nicht in der generierten Verzeichnisliste auftauchen sollen, wobei die Wildcards `*` und `?` im bekannten Shellformat verwendet werden können. Mehrere `IndexIgnore`-Anweisungen erweitern diese Liste, anstatt sie zu ersetzen. Standardmäßig enthält die Liste den Eintrag „`.`“.

HeaderName

Syntax: `HeaderName` *`Dateiname`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

ReadmeName

Syntax: `ReadmeName` *`Dateiname`*
Kontext: `Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Mit `HeaderName` und `ReadmeName` kann man der generierten Verzeichnisliste Kopf- bzw. Fußzeilen hinzufügen. *`Dateiname`* gibt dabei den Namen einer Datei relativ zu dem Verzeichnis an, von dem die Liste erstellt wird. Zunächst versucht der Server dann *`Dateiname`*`.html` zu öffnen und als HTML–Kode einzubinden. Existiert eine solche Datei nicht, fügt er einfach *`Dateiname`* als Textdatei ein.

Die beiden letzten Anweisungen dieses Abschnittes ermöglichen das Speichern von Dokumenten und Skripten außerhalb des mit `DocumentRoot` angegebenen Pfades. Dies kann z.B. zur leichteren Verwaltung des Webservers und zur sicheren Speicherung von CGI–Skripten verwendet werden.

Alias

Syntax: `Alias` *`URL-Pfad Verzeichnisname`*
Kontext: `Serverkonfiguration, virtuelle Hosts`

ScriptAlias

Syntax: `ScriptAlias` *`URL-Pfad Verzeichnisname`*
Kontext: `Serverkonfiguration, virtuelle Hosts`

Die Kommandos `Alias` und `ScriptAlias` des Moduls `mod_alias` führen dazu, daß URLs, deren Pfad mit *`URL-Pfad`* beginnt, nicht relativ zu `DocumentRoot`, sondern relativ zum mit *`Verzeichnisname`* angegebenen Verzeichnis gesucht werden. Die `Alias`-Anweisung in unserer Beispielkonfiguration führt daher dazu, daß die Icons nicht in `/var/httpd/htdocs/icons/`, sondern in `/var/httpd/icons/` gesucht werden.
`ScriptAlias` legt zusätzlich fest, daß das angegebene Verzeichnis CGI–Skripten enthält (vgl. `Options`-Anweisung). Sie sollte immer verwendet werden, wenn man CGI–Skripte benötigt, da nur auf diese Weise sichergestellt werden kann, daß keine CGI–Skripten unterhalb von `DocumentRoot`, d.h. im Bereich für HTML–Dokumente, gespeichert werden müssen.

2.4.3 Die Datei `access.conf`

Hier wird der Zugriff auf die Verzeichnisse, in denen die HTML–Dokumente bzw. die CGI–Skripten abgelegt sind, festgelegt. In unserem Fall sieht das so aus:

```
# /usr/local/etc/httpd/conf/access.conf
<Directory />
  order deny,allow
  deny from all
</Directory>

<Directory /var/httpd/htdocs>
  Options Indexes FollowSymLinksIfOwnerMatch
  AllowOverride None
  order allow,deny
  allow from all
</Directory>

<Directory /var/httpd/cgi-bin>
  Options None
  AllowOverride None
</Directory>
```

Die auftretenden Anweisungen werden im folgenden kurz erläutert:

<Directory>

Syntax: `<Directory` *`Verzeichnisname`*`>...</Directory>`
Kontext: `Serverkonfiguration, virtuelle Hosts`

`<Directory>` und `</Directory>` schließen einen Anweisungsblock ein, der sich auf das angegebene Verzeichnis und dessen Unterverzeichnis bezieht. Dabei können die von der Shell gewohnten Wildcards `*` und `?` benutzt oder durch Voranstellen einer Tilde `~` reguläre Ausdrücke angegeben werden. Innerhalb des `Directory`-Anweisungsblocks lassen sich nur die innerhalb eines Verzeichniskontextes erlaubten Anweisungen verwenden.

Options

Syntax:	`Options` *`[+|-]Option [+|-]Option...`*
Voreinstellung:	`Options All`
Kontext:	`Serverkonfiguration, virtuelle Hosts, Verzeichnis, .htaccess`

Hiermit kann man angeben, welche Serverfunktionen (im entsprechenden Verzeichnissen) aktiviert werden. Folgende Optionen sind verfügbar:

`All`: Entspricht `ExecCGI`, `FollowSymLinks`, `Includes` und `Indexes`.

`ExecCGI`: Erlaubt die Ausführung von CGI–Skripten.

`FollowSymLinks`: Der Server folgt symbolischen Links.

`Includes`: Erlaubt „server–side includes".

`IncludesNoExec`: „server–side includes" sind erlaubt, aber `#exec` und `#include` in CGI–Skripten sind ausgeschaltet.

`Indexes`: Falls eine URL angefragt wird, die auf ein Verzeichnis zeigt, das keine `index.html`-Datei (oder entsprechendes) enthält, wird eine formatierte Ausgabe des Verzeichnisses zurückgeliefert, falls diese Option gesetzt ist.

`MultiViews`: Schaltet die Unterstützung von Multiviews ein, d.h. Client und Server können das Rückgabeformat und die Sprache von Dokumenten untereinander aushandeln.

`SymLinksIfOwnerMatch`: Symbolische Links werden nur dann verfolgt, wenn sie auf Dateien oder in Verzeichnisse führen, die dem gleichen Benutzer gehören wie der Link.

Wird eine Option ohne vorangestelltes `+` oder `-` angegeben, werden alle Optionen ausgeschaltet, d.h. auch die Standardoption `All`.
Hinweis: CGI–Skripten sollten nur in Verzeichnissen abgelegt werden, die außerhalb des eigentlichen Bereiches für HTML–Dokumente liegen. Dies wird im allgemeinen durch Verwenden der Anweisung `ScriptAlias` realisiert. Diese schaltet ihrerseits wiederum die Erlaubnis zum Ausführen von CGI–Skripten ein, weshalb man diese Erlaubnis durchaus im Rahmen der `Options`-Anweisung verwehren sollte (vgl. Beispielkonfiguration).

AllowOverride

Syntax:	`AllowOverride` *`Option Option ...`*
Voreinstellung:	`AllowOverride All`
Kontext:	`Verzeichnis`

Diese Anweisung gibt an, welche Einstellungen eine `.htaccess`-Datei noch verändern kann. In unserem Fall haben wir hier `None` angegeben, was die Verwendung dieser Datei gewissermaßen ausschaltet. Näheres in [2, 3].

order

Syntax:	`order` *`Ordnung`*
Voreinstellung:	`order deny,allow`
Kontext:	`Verzeichnis, .htaccess`

Diese Option benötigt das Modul `mod_access` und gibt an, in welcher Reihenfolge `allow`- und `deny`-Anweisungen ausgewertet werden.

`Ordering` kann folgende Werte annehmen:

`deny,allow`: Wenn ein Client durch eine `deny`-Anweisung ausgeschlossen wird, wird er abgewiesen, sofern ihm nicht eine `allow`-Anweisung doch den Zugriff erlaubt. Falls keine der beiden Anweisungen auf ihn zutrifft, wird ihm der Zugriff erlaubt.

`allow,deny`: Wenn eine `allow`-Anweisung den Zugriff eines Clients erlaubt, erhält er diesen auch, falls keine `deny`-Anweisung auf ihn zutrifft. Trifft keine der beiden Anweisungen auf ihn zu, wird der Zugriff verweigert.

`mutual-failure`: Hier wird der Zugriff eines Clients nur erlaubt, wenn eine `allow`–Anweisung für und keine `deny`–Anweisung gegen ihn spricht.

allow

Syntax: `allow from` *`Host Host...`*
Kontext: `Verzeichnis, .htaccess`

deny

Syntax: `deny from` *`Host Host...`*
Kontext: `Verzeichnis, .htaccess`

Diese Anweisungen stehen wiederum nur bei einkompiliertem `mod_access`-Modul zur Verfügung. Sie legen zusammen mit der `order`-Anweisung fest, welchen Clients der Zugriff auf die entsprechenden Verzeichnisse gestattet wird. Die Angabe `all` für *`Host`* paßt dabei auf alle Clients. Ansonsten kann man (Teile von) Hostnamen oder IP–Adressen angeben und so den Zugriff steuern.

Der Vollständigkeit halber sei noch hinzugefügt, daß man natürlich auch den Zugriff auf Benutzerbasis steuern kann. Dies verlangt allerdings etwas mehr Aufwand und kann daher im Rahmen dieses Vortrages nicht behandelt werden.

3 Start des Apache

Nachdem man die Konfiguration des Apache erledigt hat, kann man ihn als `root` einfach mit

```
httpd
```

starten, wobei vorausgesetzt wurde, daß sich das Verzeichnis `/usr/local/bin` im Pfad des Systemverwalters befindet. In unserem Fall ist es nicht nötig, den Namen der Konfigurationsdatei anzugeben, da wir ja die Standardeinstellung `/usr/local/etc/httpd/conf/httpd.conf` verwendet haben. Im allgemeinen kann man mit der Kommandozeilenoption `-f` den Namen dieser Datei angeben.

Hat man nun noch einige HTML–Seiten, beispielsweise die mitgelieferte Dokumentation zum Apache [3], in `/var/httpd/htdocs` abgelegt, kann man mit einem beliebigen Browser unter der URL `http://localhost/` den Erfolg der ganzen Bemühungen bewundern. Falls Probleme auftauchen, kann ein Blick in die Fehlerprotokolldatei `/usr/local/etc/httpd/logs/error_log` behilflich sein.

Literatur & Links

[1] `http://www.apache.org/`

[2] Laurie, Ben; Laurie, Peter, *Apache: The Definitive Guide* Cambridge: O'Reily & Associates, 1997

[3] *Apache User's Guide* im Unterverzeichnis `htdocs/manual` der Apache–Quellen