

Versuch m-39-a Apache Konfiguration

Steve Moser, Markus Ganzenmüller

21. November 2002

Inhaltsverzeichnis

1	Allgemein	2
1.1	Einleitung	3
1.2	Aufgabenstellung	3
1.3	Client/Server Prinzip	4
1.4	Arten von Servern	4
1.5	Kommunikationsprotokolle	5
2	Apache Konfiguration	7
2.1	Installation des Apache	8
2.2	Konfigurationsdateien	9
2.3	httpd.conf	9
2.4	Alternativkonfiguration	20
2.5	Die rc.config	20

Kapitel 1

Allgemein

1.1 Einleitung

Jeder Computerbenutzer der im Internet surft, hatte, meist ohne es zu wissen, schon einmal mit einem Apache Webserver zu tun, welcher die benötigten Daten an seinen Browser übermittelt hat. Kaum jemand weiß jedoch, was eigentlich ein Webserver ist und was er tut, geschweige denn, wie man selbst einen Webserver einrichtet und konfiguriert. In den folgenden Kapiteln sollen zunächst kurz einige grundlegende Begriffe des Internets erklärt werden, so z.B. das Client/Server-Prinzip und die verschiedenen Arten von Servern, die es gibt. Dies ist nötig, um einen Überblick über die Arbeitsweise eines Servers zu bekommen und die danach folgenden Kapitel bezüglich der Einrichtung und Konfiguration eines Apache-Webserver besser zu verstehen. Die Verwendung von Unix/Linux Servern, insbesondere von Apache-Servern hat sich in den letzten Jahren vervielfacht, und das mit Recht. Gute Erreichbarkeit, Stabilität, geringer Wartungsaufwand und Schnelligkeit sind nur einige der Gründe, warum sich sowohl Firmen, als auch Privatanwender für die Verwendung von Unix/Linux Servern entscheiden. Tests haben ergeben, dass diese z.B. Windows-NT Servern in punkto Erreichbarkeit um den Faktor fünf überlegen sind. Der Name "Apache" leitet sich aus der Tatsache her, daß das Apache-Projekt von einer Gruppe Programmieren gebildet wurde, die Patches zu den NCSA-Quellen geschrieben hatten. Nach deren Zusammenschluß ergab sich A PAtCHy Server.

1.2 Aufgabenstellung

Thema: Installation und Konfiguration
Rechner: G215 Platz A3, Minos
Betriebssystem: Linux

Aufgabenstellung:

Konfigurieren sie eine neue Instanz des Apache-Servers ein (Port 8080).

Richten sie PHP-Support durch Apache ein.

Kolloquium:

Welche Rolle spielt die S.U.S.E Konfiguration (`/etc/rc.config`) bei der Konfiguration von Apache?

1.3 Client/Server Prinzip

Das Internet kann man als ein riesiges Netzwerk bezeichnen, welches aus vielen Clients und Servern besteht. Ein Web-Server ist ein Rechner, oder auch nur ein Programm welches dabei als Knotenpunkt fungiert, und verschiedene Teilbereiche des Internets miteinander verbindet, Informationen oder Anfragen nach bestimmten Webseiten beantwortet, oder an einen andern Server weiterleitet. Wurden die benötigten Daten auf einem Server gefunden, werden diese an den Client-Rechner weitergeleitet, welcher sie dann in lesbarer Weise auf dem Bildschirm darstellt. Server sind die Anbieter von Informationen, Ressourcen, Dienstleistungen und Daten, welche die Clients nutzen. Übersetzt heißt "to serve" nichts anderes als "dienen, jemanden versorgen". Als Client wird jeder Rechner bezeichnet, der die Dienste eines Servers in Anspruch nimmt. Wie im nächsten Abschnitt ersichtlich wird, sind die Aufgabengebiete von Servern sehr weit gefächert.

1.4 Arten von Servern

Es gibt eine Menge von verschiedenen Arten von Servern, welche sich durch ihre Funktion unterscheiden.

File-Server:

File-Server stellen ihren Client-Rechnern Dateien und viel Platz auf ihrem System zur Verfügung. Zusätzlich übernehmen sie oft Datensicherungsfunktionen. Einsatzgebiete dafür sind vor allem größere Unternehmen, die ein Intranet betreiben, und dessen Daten im ganzen Unternehmen (oder auch außerhalb) erreichbar sein sollen.

Datenbank-Server:

Auf einem Server dieser Art läuft eine Datenbankprogramm (z.B. von Oracle). Der Server hat die Aufgabe, Daten in diese Datenbank einzufügen, zu löschen, Änderungen vorzunehmen, Daten zu sortieren, oder nach Daten zu suchen, um diese an das anfragende System weiterzuleiten.

Application-Server:

Ein Application-Server stellt dem Client Anwendungsprogramme zur Verfügung. Der Kernpunkt dabei ist, dass das Anwendungsprogramm dabei nicht auf dem Client-Rechner läuft, sondern auf dem Server selbst, welcher nur ein grafisches Frontend an den Client schickt, über das dieser das Programm bedient. Der Vorteil dabei ist, dass alle Anwendungsprogramme von einer zentralen Stelle aus verwaltet werden können. Zudem ergeben sich Einsparungen bei der Hardware der Clients, weil diese unter anderem keine Festplatte benötigen, da sie ja keine Daten speichern, sondern nur der Server.

Compute-Server:

Ein Compute-Server funktioniert nach einem ähnlichen Prinzip wie die drei vorangegangenen Serverarten, nur das den Clients diesmal keine Anwendungen oder Speicherplatz zur Verfügung gestellt wird, sondern Rechenzeit. Dabei sendet ein Client für die Verarbeitung vorbereitete Daten an den Compute-Server, welcher diese dann verarbeitet und die Ergebnisse an den Client zurückschickt. Verwendung finden solche Strukturen vor allem in wissenschaftlichen Instituten, die aufwändige Berechnungen durchführen, für die ein normaler PC Wochen oder Monate brauchen würde. Die Rechenleistung eines Compute-Servers ist um ein vielfaches höher als die eines normalen Computers.

(Streaming) Media-Server:

Streaming/Media-Server stellen Multimediadaten, z.B. Video- oder Audio-Streams, in Echtzeit zur Verfügung.

Internet/Web-Server:

Internet- bzw. Web-Server stellen Internet- und Intranet-Dienste zur Verfügung. Typische Dienste dabei umfassen das World Wide Web, Domain-Name-Services, EMail, FTP, usw...

1.5 Kommunikationsprotokolle

Mittlerweile ist klar, welche vielfältigen Aufgabengebiete Server haben. Es stellt sich jedoch die Frage, auf welche Art und Weise Server und Client eigentlich miteinander kommunizieren. Dazu gibt es verschiedene Protokolle, die definieren, nach welchen Regeln die Kommunikation stattfindet. Die nun folgende Aufzählung umfasst nur einen geringen Teil der grundlegenden Protokollarten in offenen Netzen.

IP

Abkürzung für "INTERNET Protocol" . Das IP gehört zur TCP/IP Protokollfamilie, einem anerkannten Industriestandard zur Kommunikation zwischen offenen Systemen. Das Übertragungsprotokoll definiert die Regeln und Vereinbarungen, die den Informationsfluß in einem Kommunikationssystem steuern. Hauptaufgabe des IP ist die netzübergreifende Adressierung. Das Protokoll arbeitet nicht leitungs-, sondern paketorientiert: Sogenannte Datagramme suchen sich über die jeweils verfügbaren Verbindungen ihren Weg zum Empfänger.

TCP

Abkürzung für "Transmission Control Protocol over Internet Protocol" . Standard-Protokoll im Internet / Intranet sowie WAN und Unix-Netzen. Das Internet Protocol (IP) dient der Fragmentierung und Adressierung von Daten und übermittelt diese vom Sender zum

Empfänger, sichert die Übertragung allerdings nicht ab. Das Transmission Control Protocol (TCP) baut darauf auf, sorgt für die Einsortierung der Pakete in der richtigen Reihenfolge beim Empfänger und bietet die Sichererstellung der Kommunikation durch Bestätigung des Paket-Empfangs. Es korrigiert Übertragungsfehler automatisch.

Das PPP ("Point to Point Protocol") ist ein moderner Zusatz zu TCP/IP, der auch Netzwerk-Verbindungen über Modems erlaubt.

HTTP

Abkürzung für "Hypertext Transfer Protocol". Kommunikationsprotokoll zwischen Web-Server und Web-Browser zur Übertragung von HTML-Daten. Das HTTP-Protokoll stellt die oberste von mehreren Protokoll-"Schichten" zur Verwaltung im Internet dar:

- Das sogenannte Internet Protocol (IP) stellt die Grundlage dar, die das Internet definiert.
- Das sogenannte Transport Control Protocol (TCP) fungiert als Zwischenschicht und richtet die Verbindungswege zur Datenübertragung ein.
- Das HTTP-Protokoll teilt in der obersten Schicht die Daten in einzelne Pakete auf und legt fest, wie diese verschickt werden sollen.

Sendet ein Browser per http-Protokoll eine Anfrage nach einem Dokument an einen Server, liefert dieser ihm einerseits das Dokument selbst, zusätzlich jedoch auch den sogenannten MIME-Type, was nichts anderes heißt, als dass dem Browser mitgeteilt wird, wie er die bekommenen Daten interpretieren soll, also um was für eine Art von Daten es sich handelt. MIME bedeutet "Multi Purpose Mail Extensions".

Im nächsten Kapitel soll nun die Konfiguration eines Apache-Webservers unter SuSe Linux genauer erläutert werden. Dazu sollen folgende Fragen beantwortet werden:

- Wo bekomme ich die Installationspakete und wie werden sie installiert ?
- Welche Konfigurationsdateien beeinflussen das Verhalten des Apache ?
- Wie, bzw. wo konfiguriert man einen Apache-Server ?
- Welche Hilfsmittel stehen zur Konfiguration zur Verfügung
- Welche Funktionen bietet er ?

Kapitel 2

Apache Konfiguration

2.1 Installation des Apache

Wenn man sich für die Installation eines Apache-Web-Servers entschlossen hat, stellt sich zunächst die Frage, woher man die benötigten Software-Pakete bekommt, die dafür notwendig sind. Dafür gibt es zwei Möglichkeiten. In unserem Fall haben wir auf die Distributions-CDs von SuSe zurückgegriffen, weil die dort vorhandene Apache-Version für unsere Testzwecke vollkommend ausreichend war. Allgemein ist es jedoch empfehlenswert, auf den Webseiten von <http://www.apache.org> die neueste Version herunterzuladen und diese zu installieren, um sicherzugehen, dass sich keine unnötigen Bugs in der Software befinden, die eigentlich schon längst behoben wurden. Vor allem im Hinblick auf eine Nutzung in Unternehmen ist dies sinnvoll, aber auch Privatanwender sollten keine unnötigen Risiken eingehen.

Hier ein Überblick für Ungeduldige :

Download : `lynx http://www.apache.org/dist/httpd/httpd-2_0_NN.tar.gz`

Auspacken :

`gzip -d httpd-2_0_0 NN.tar.gz`

`tar xvf httpd-2_0_NN.tar`

Konfigurieren : `./configure --prefix=PREFIX`

Kompilieren : `make`

Installieren : `make install`

Anpassen : `vi PREFIX/conf/httpd.conf`

Testen : `PREFIX/bin/apachectl start`

NN muss durch die Nummer der Unterversion ersetzt werden, und PREFIX durch den Verzeichnispfad, in dem der Server installiert werden soll. Wenn PREFIX nicht angegeben ist, wird die Voreinstellung `/usr/local/apache2` verwendet.

Für eine genauere Beschreibung der verschiedenen Installationsversionen ist ein Besuch auf der oben angegebenen Webseite, also <http://www.apache.org>, sehr hilfreich. Die gesamte Dokumentation des Apache ist ausführlich und umfangreich.

2.2 Konfigurationsdateien

Verlief die Installation erfolgreich, ist es die nächste Aufgabe, die Installation an die eigenen Bedürfnisse anzupassen. Dafür sollte man sich die entsprechende Zeit nehmen, da eine fehlerhafte Konfiguration zu erheblichen Sicherheitsmängeln führen kann.

Apache (in unserem Fall Version 1.3) wird in folgenden Verzeichnissen installiert:

```
/etc/httpd      -> Konfigurationsdatei(en)
/usr/sbin/httpd -> Serverprogramm
/var/log/httpd/ -> Protokolldateien
/var/run/httpd.pid -> PID des Servers
/usr/local/httpd -> Server-Root
/etc/init.d/apache -> Start/Stop-Skript
```

Die folgenden Dateien müssen auf einem SuSe-System angepasst werden:

```
/etc/httpd/httpd.conf
/etc/rc.config
/etc/httpd/suse_include.conf
/etc/httpd/suse_public_html.conf
/etc/httpd/suse_loadmodule.conf
/etc/httpd/suse_addmodule.conf
```

2.3 httpd.conf

Die Datei *httpd.conf* ist die wichtigste Konfigurationsdatei zur Steuerung des Apache-Servers, bzw. dem *httpd*-Dämon, dem eigentlichen Serverprogramm. Eine komplette Beschreibung wäre zu umfangreich, deshalb sollen nur die wichtigsten Auszüge daraus erklärt werden.

```
1 #####
2 # 1. Abschnitt: Globale Umgebung
3
4 # Die Anweisungen in diesem Abschnitt bestimmen grundsätz-
5 # lich die Arbeitsweise des Apache-Servers.
6
```

```

7 # "ServerType": Der Server-Typ ist entweder "inetd" oder
8 # "standalone". "inetd" wird allerdings nur von UNIX-
9 # Plattformen unterstützt/verwendet.
10
11 ServerType inetd
12
13 # "ServerRoot" bezeichnet die Spitze des Verzeichnisbaums,
14 # unter dem die Server-Konfiguration, Fehleranzeigen und
15 # Protokolle zu finden sind.
16 # Am Ende dieser Zeile darf KEIN slash stehen!!!
17
18 ServerRoot "/usr/local/httpd"
19
20 # "PidFile": das ist die Datei, in der der Server jedesmal
21 # dann seine "process identification number" niederlegen
22 # sollte, wenn er gestartet wird.
23
24 PidFile /var/run/httpd.pid
25
26 # In der Standard-Konfiguration wird der Server die Dateien
27 # httpd.conf (diese Datei), srm.conf, und access.conf in
28 # dieser Reihenfolge abarbeiten. Die beiden letztgenannten
29 # Dateien werden jetzt leer belassen, da empfohlen wird,
30 # zur besseren Übersicht sämtliche Anweisungen in einer
31 # einzigen Konfigurationsdatei zusammenzufassen.
32 # Die auskommentierten Werte unten sind jeweils die
33 # vorgegebenen Standardwerte. Man kann festlegen, dass der
34 # Server diese Dateien ignoriert, indem man den Anweisungen
35 # den Parameter "/dev/null" mit-gibt.
36
37 #ResourceConfig conf/srm.conf
38 #AccessConfig conf/access.conf
39
40
41 # "Timeout" legt die Sekundenanzahl fest, ehe ein Timeout
42 # (eine laufzeitbedingte Unterbrechung) gesendet wird.
43
44 Timeout 300
45
46
47 # "KeepAlive": legt fest, ob persistente Verbindungen

```

```
48 # (mehr als eine Anfrage pro Verbindung) zulässig sind. Wird
49 # mit "Off" deaktiviert.
50
51 KeepAlive On
52
53
54 # "MaxKeepAliveRequests": die Höchstzahl der während einer
55 # persistenten Verbindung zulässigen Anfragen. Wird hier 0
56 # angegeben, ist unbegrenzter Zugriff möglich. Empfohlen wird
57 # ein hoher Wert, um eine hohe Performance zu erhalten.
58
59 MaxKeepAliveRequests 200
60
61
62 # "KeepAliveTimeout": Zeitspanne (Zahl an Sekunden), um auf die
63 # nächste Abfrage desselben Clients mit derselben Verbindung zu
64 # warten.
65
66 KeepAliveTimeout 15
67
68
69 # Größenanpassung des Server-Pools: ähnlich wie man selbst
70 # abschätzen wird , wieviele Server-Prozesse man braucht,
71 # passt sich Apache dynamisch der Belastung an, die er
72 # erkennt - das bedeutet, er versucht, genügend Server-
73 # Prozesse zur Bewältigung der aktuellen Serverlast
74 # bereitzustellen und zusätzlich noch ein paar Ersatz-Server,
75 # damit kurzzeitige Spitzenbelastungen (z.B. multiple simultane
76 # Requests von einem einzelnen Netscape-Browser) bewältigt
77 # werden können.
78 #
79 # Das verläuft so, dass periodisch abgefragt wird, wieviele
80 # Server gerade auf eine Rückmeldung warten. Wenn das weniger
81 # sind als in "MinSpareServers" vorgegeben, wird ein neuer
82 # erstellt. Sind das mehr als in "MaxSpareServers" vorgegeben,
83 # werden einige beendet (sie "sterben"). Die Standardvorgaben
84 # sind vermutlich für die meisten Seiten so in Ordnung.
85
86 MinSpareServers 1
87 MaxSpareServers 1
88
```

```
89
90 # "StartServers": Anzahl der Server, die zu Beginn gestartet
91 # werden. Das sollte eine vernünftige Vorgabe sein.
92
93 StartServers 5
94
95
96 # "MaxClients": Höchstzahl der gleichzeitig laufenden
97 # Server. Damit wird auch die Zahl der Clients eingegrenzt,
98 # die simultan mit dem Server verbunden werden können. Wenn
99 # diese Zahl erreicht ist, werden weitere Clients
100 # ausgeschlossen; also sollte die Vorgabe kein zu kleiner
101 # Wert sein. Diese Anweisung soll vor allem verhindern, dass
102 # ein "durchdrehender" Serverprozess das gesamte System
103 # mitnimmt, wenn er sich beendet.
104
105 MaxClients 150
106
107
108 # "MaxRequestsPerChild": Höchstzahl an Anfragen, die jeder
109 # "Kind-Prozeß" (der Webserver startet bei Anfragen solche
110 # "Kind-Prozesse", in denen die Threads ablaufen können)
111 # behandeln darf, ehe er beendet wird. Das Beenden von
112 # "Kind-Prozessen" ist wichtig, um Probleme zu vermeiden,
113 # falls Apache (und möglicherweise die von ihm verwendeten
114 # Bibliotheken) den Speicher oder andere Ressourcen
115 # aufgebraucht hat. Auf den meisten Plattformen wird das
116 # zwar kaum einmal vorkommen, aber ein paar, wie zum Beispiel
117 # Solaris, verfügen über beachtliche "Leaks" in ihren
118 # Bibliotheken. Für solche Plattformen sollte man hier
119 # 10 000 oder einen ähnlichen Wert festlegen. Wenn man 0 (Null)
120 # vorgibt, ist die Anzahl unbegrenzt.
121 #
122 # Achtung: mit Ausnahme der Initial-Anfrage gilt dieser Wert
123 # NICHT für "Keepalive"-Anfragen. Wenn ein "Kind-Prozeß"
124 # beispielsweise mit einer Initial-Anfrage startet, der
125 # weitere zehn Anfragen folgen, so würde hier lediglich 1
126 # gezählt werden.
127
128 MaxRequestsPerChild 0
129
```

```

130
131 # "Listen": Diese Anweisung gestattet es, Apache mit spezifi-
132 # schen IP-Adressen und/oder Ports zu verbinden, zusätzlich
133 # zu den Standardvorgaben.
134
135 Listen 3000
136 Listen 192.168.0.1:80
137
138
139 # "BindAddress": mit dieser Option können virtuelle Hosts
140 # unterstützt werden. Die Anweisung wird genutzt, um dem
141 # Server mitzuteilen, welche IP er wählen soll. Sie kann
142 # entweder einen Asterisk (*) oder einen vollständigen
143 # Internetnamen enthalten.
144
145 BindAddress *
146
147 # Die folgenden Module werden geladen, wenn das zugehörige
148 # Paket auf dem System installiert ist. Die entsprechenden
149 # Variablen werden in /sbin/init.d/apache definiert und
150 # kontrollieren, welche Module dynamisch geladen werden.
151
152 <IfDefine PHP>
153     LoadModule php3module          /usr/lib/apache/libphp3.so
154 </IfDefine>
155 <IfDefine PHP4>
156     LoadModule php4module          /usr/lib/apache/libphp4.so
157 </IfDefine>
158
159                               usw...
160
161 <IfDefine MODULES>
162 LoadModule allowdevmodule    /usr/lib/apache/modallowdev.so
163 LoadModule cookieauthmodule /usr/lib/apache/modauthcookie.so
164 </IfDefine>
165
166 #####
167 #
168 # 2. Abschnitt: Generelle Server-Konfiguration
169 # -----
170 #

```

```

171 # Die Anweisungen in diesem Abschnitt legen die generell vom
172 # Server benötigten Werte fest, mit denen auf alle Anfragen
173 # reagiert wird, die nicht von virtuellen Hosts abgearbeitet
174 # werden.
175 # Alle diese Anweisungen können auch innerhalb von
176 # <VirtualHost>-Containern stehen, womit dann die hier
177 # vorgenommenen Standard-Einstellungen für den jeweiligen
178 # virtuellen Host berschrieben werden.
179 #
180 # Falls die Festlegung zu "ServerType" (weiter oben im
181 # Abschnitt "Globale Umgebung") "inetd" lautet, haben die
182 # nächsten paar Anweisungen hier keinerlei Auswirkungen, da
183 # ihre Wirksamkeit bzw. Nichtwirksamket mit der Konfiguration
184 # für inetd definiert wird. Man kann in diesem
185 # Fall die folgenden Schritte bis zu den Festlegungen für
186 # "ServerAdmin" übergehen.
187 #
188 #-----
189 #
190 # "Port": der port, den der Server benutzt. Manche Firewalls
191 # müssen korrekt konfiguriert sein, wenn Apache auf einen
192 # bestimmten Port zugreifen soll.
193 #
194 # Für niedrigere ports als 1023 muss man Apache zunächst als
195 # "root" starten.
196
197 Port 80
198
199
200 # SSL-Unterstützung (Secure Socket Layer)
201 #
202 # Wenn auch für SSL gesorgt werden soll, muss der Standard-
203 # HTTP-Port ebenso genutzt werden können wie der Standard-HTTPS
204 # -Port.
205 #
206 <IfDefine SSL>
207     Listen 80
208     Listen 443
209 </IfDefine>
210
211

```

```

212 # "ServerAdmin" gibt die Adresse an, an die eventuelle Probleme
213 # mit dem Server gemeldet werden können. Diese Adresse erscheint
214 # auf allen Dokumenten, die vom Server generiert werden, zum
215 # Beispiel Fehlermeldungen.
216
217 ServerAdmin root@localhost
218
219
220
221 # "DocumentRoot": das Verzeichnis, von dem aus die zur
222 # Publikation vorgesehenen Dokumente erreicht werden können.
223 # Im Standardfall werden sämtliche Server-Anfragen von diesem
224 # Verzeichnis aus beantwortet, und symbolische links sowie Aliase
225 # können auf andere Verzeichnisse verweisen.
226
227 DocumentRoot "/usr/local/httpd/htdocs"
228
229
230 # Jedes Verzeichnis, auf das Apache zugreifen kann, kann unter
231 # Berücksichtigung dessen, welche Dienste und Features hier
232 # (und in Unterverzeichnissen) erlaubt und/oder nicht zugelassen
233 # sind, konfiguriert werden.
234 #
235 # Zuerst wird ein "Standard" so konfiguriert, dass sich ein
236 # sehr restriktiver Satz an Berechtigungen ergibt.
237
238 <Directory />
239     AuthUserFile /etc/httpd/passwd
240     AuthGroupFile /etc/httpd/group
241     Options -FollowSymLinks +Multiviews
242     AllowOverride None
243 </Directory>
244
245         usw...
246
247 # Zugriffsberechtigungen für UserDir-Verzeichnisse. Das
248 # Folgende ist ein Beispiel, mit dem diese Verzeichnisse nur
249 # gelesen werden dürfen (read-only).
250
251 <Directory /home/*/public-html>
252     AllowOverride FileInfo AuthConfig Limit

```

```

253     Options MultiViews Indexes SymLinksIfOwnerMatch
254             IncludesNoExec
255     <Limit GET POST OPTIONS PROPFIND>
256         Order allow,deny
257         Allow from all
258     </Limit>
259     <LimitExcept GET POST OPTIONS PROPFIND>
260         Order deny,allow
261         Deny from all
262     </LimitExcept>
263 </Directory>
264
265             usw...
266
267 # "AccessFileName": Name der Datei, die in jedem Verzeichnis
268 # Informationen über Zugriffsberechtigungen zur Verfügung
269 # stellen kann.
270
271 AccessFileName .htaccess
272
273
274 # Die folgenden Zeilen schützen ".htaccess"-Dateien davor,
275 # dass Clients deren Inhalt auslesen können. Aus
276 # Sicherheitsgründen ist der Zugriff normalerweise nicht
277 # gestattet, da ".htaccess" Informationen zur Autorisierung
278 # enthält. Analog sollten Dateinamen wie ".htpasswd" für
279 # eine Datei mit dem aktuellen Passwort verwendet werden,
280 # die dann ebenfalls vor Zugriffen geschützt ist.
281
282 <Files ~ "\.ht">
283     Order allow,deny
284     Deny from all
285 </Files>
286
287             usw...
288
289 # "TypesConfig" zeigt an, ob/wo die Datei mime.types
290 # (oder ihr äquivalent) gefunden werden kann.
291
292 <IfModule mod-mime.c>
293     TypesConfig /etc/httpd/mime.types

```

```

294 </IfModule>
295
296
297 # "DefaultType" ist der Standard-MIME-Typ, den der Server
298 # einem Dokument zuweist, falls es nicht von einer Extension
299 # her genauer bestimmt wird. Wenn Ihr Server überwiegend Text-
300 # oder HTML-Dokumente enthält ist "text/plain" ein guter
301 # Wert. Wenn die meisten Dateien Binärdateien sind
302 # (z. B. Applikationen oder Bildformate wie .BMP) sollten
303 # Sie "application/octet-stream" angeben, statt den
304 # Browsern zu erlauben, diese Binärdateien so darzustellen,
305 # als ob sie Text wären.
306
307 DefaultType text/plain
308
309         usw...
310
311 # "HostnameLookups": Zeichnet die Namen von Clients auf
312 # oder auch nur deren IP-Adresse; z.B. www.apache.org (on)
313 # oder 204.62.129.132 (off). Der Standard ist off, weil es
314 # insgesamt für das Netz besser ist, wenn die Leute dieses
315 # Feature bewusst nutzen. Es bedeutet nämlich, dass jede
316 # Anfrage eines Clients mindestens eine lookup-Rückfrage
317 # an den Nameserver weitergibt.
318
319 HostnameLookups Off
320
321         usw...
322
323 #####
324 #
325 # 3. Abschnitt: Virtuelle Hosts
326 # -----
327 #
328 # "VirtualHost": Wenn mit mehreren Domain- bzw. Host-Namen
329 # auf diesem Server gearbeitet werden soll, können virtuelle
330 # "Container" für jeden einzelnen Hostnamen festgelegt werden.
331 # Bitte unbedingt die Dokumentation auf
332 # <http://www.apache.org/docs/vhosts/> nachlesen bzw. das
333 # mit der Distribution ausgelieferte Manual. Um die
334 # Konfiguration der virtuellen Hosts zu überprüfen, steht der

```

```

335 # Befehlszeilen-Parameter "-S" zur Verfügung.
336 #
337 # Konfiguration für DNS-Namen-gesttzte virtuelle Hosts
338
339 NameVirtualHost *
340
341
342 # Beispiel für einen virtuellen Host:
343 #
344 # Fast jede Apache-Anweisung kann in einem "VirtualHost"-
345 # Container verwendet werden.
346 # Der erste Container in der (beliebig langen) Liste wird
347 # dabei immer genutzt, wenn eine Webadresse ohne bekannten
348 # Servernamen verlangt wird.
349
350 <VirtualHost *>
351     ServerAdmin webmaster@dummy-host.example.com
352     DocumentRoot /www/docs/dummy-host.example.com
353     ServerName dummy-host.example.com
354     ErrorLog logs/dummy-host.example.com-error_log
355     CustomLog logs/dummy-host.example.com-access_log common
356 </VirtualHost>
357 #####

```

Wie man sieht, gibt es alleine in der Datei *httpd.conf* enorm viele Konfigurationsparameter, um das Verhalten des Apache-Webservers zu bestimmen. Deshalb ist es, wie oben schon erwähnt, sehr ratsam, die Dokumentation genau zu lesen, um potentiell vorhandene Sicherheitsrisiken zu minimieren. Es ist immer wieder nötig, den httpd-Daemon neu zu starten, wenn man Veränderungen an den Konfigurationsdateien vorgenommen hat. Es existieren dazu zwei Möglichkeiten:

- Beim normalen Restart wird dem Hauptprozeß ein HUP- Signal geschickt (*kill -HUP 'cat /var/run/httpd.pid'*). Alle bestehenden Verbindungen werden dabei beendet und die Logfiles geschlossen.
- Seit der Version 1.2 unterstützt der Apache einen sogenannten Graceful Restart, bei dem bestehende Client-Verbindungen nicht getrennt werden (*kill -USR1 'cat /var/run/httpd.pid'*).

Grundsätzlich ist diese Methode vorzuziehen.

Ausnahme: wenn Änderungen an einem Logfile- Format vorgenommen wurden sind. Solange noch aktive Verbindungen bestehen, benutzen diese das alte Logfile.

Beendet wird der Apache Server durch ein TERM-Signal an den Hauptprozeß. Daraufhin werden sofort alle Kindprozesse terminiert, die Logfiles geschlossen und anschließend der Hauptprozeß selbst beendet. Das Kommando dafür lautet: *kill `cat /var/run/httpd.pid`*

2.4 Alternativkonfiguration

Es gibt auch weitere Möglichkeiten einen Apache nach der Installation zu konfigurieren. Ein Beispiel dafür ist das grafische Tool "Comanche". Mit diesem Tool wird dem Anwender die Möglichkeit gegeben alle Dateien im Überblick zu haben die zu einer vollständigen Administration nötig sind. Voraussetzung ist natürlich das der Apache ordnungsgemäß installiert wurde. Es besteht desweiteren die Möglichkeit die html Dateien, die im DocumentRoot Verzeichnis liegen, nach zu bearbeiten. Die Dateien error.log und access.log sind auf einen Klick einsehbar, sowie alle Informationen über Module die geladen werden. Natürlich kann man den Server auch starten, aktualisieren oder stoppen.

2.5 Die rc.config

Eigentlich steht hier nichts weltbewegendes darin. Zum einen findet man eine Pfadvariable wie folgt:

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

und den Eintrag dass der Apache gestartet werden soll. Wobei anzumerken ist, dass durch dieses Argument nur festgelegt wird, dass der Apache gestartet werden soll. Dies hängt wiederum davon ab in welchem runlevel gestartet wird. Es gibt 5 startlevel.

- 0: System halt. Alles wird runtergefahren, Kernel wird angehalten, Rechner wird ausgeschaltet (falls man ein ATX Mainboard hat, dies eingestellt ist, und das BIOS dies unterstützt).
- 6: System reboot. Alles wird runtergefahren, System wird neu gestartet.
- S: Single User Mode. Ein Terminal, nur root darf rein, kein Netz.
- 1: Multiuser ohne Netzwerk. (meistens...)
Ab hier ist alles distributionsspezifisch (- siehe '/etc/inittab'!)
- 2: Multiuser mit Netzwerk.
- 3: Multiuser mit Netzwerk und grafischem Login.
- 4: Benutzerdefiniert.

5: Benutzerdefiniert

In unserem Beispiel wird Standardmässig Runlevel 5 gestartet. Im Verzeichnis `/etc/init.d/rc5.d/` sind alle Verknüpfungen eingefügt die beim Start des Runlevel 5 ausgeführt werden sollen. Verknüngen heisst: Linux legt ein Verzeichnis `rc5.d` an. Dort werden eine Sammlung von Links eingefügt, die wiederum auf die zu startenden Skripte verweisen. z.B. `"K14nfs"` verzweigt auf das `networkfilesystem` das wiederum die Einstellungen dessen festlegt.

Abschließend haben wir festgestellt, daß eine Menge Grundwissen dazu gehört, um einen Apache-Server richtig zu installieren und zu konfigurieren. Hat man sich jedoch einmal mit der Thematik auseinandergesetzt, bereitet der Rest keine Schwierigkeiten mehr. Zudem hat man durch das genaue Studium der Konfigurationsdateien genug Wissen erworben, um volle Kontrolle über das System zu erlangen. Auch zusätzliche Features, wie z.B. PHP-Unterstützung oder die Anbindung an eine MySQL-Datenbank, wie sie sich durch ergänzende Module einrichten lassen, stellen kein großes Problem mehr dar.