Administration eines Apache Servers

Inahlt

1. Einführung 3
2. Installation 4
3. Konfiguration 7
3.1. Einstellungen in httpd.conf 7
3.2. Einstellungen in access.conf 9
3.3. Einstellungen in srm.conf 11
3.4. Einstellungen in mime.types 13
4. HTTP-Daemon 16
4.1. Funktionsprinzip 16
4.2. Starten 16
4.3. Beenden 17
5. Wichtige Funktionalitäten 18
5.1. Common Gateway Interface (CGI) 18
5.2. Server-Side Include (SSI) 18
5.3. Content Negotiation 19
Type-Maps 19
Multiviews 20
6. Überwachung 21
6.1. Sicherheit 21
6.2. Logfile Analyse 21
6.3. Serverinformationen 22
7. Zusammenfassung 24

1. Einführung

Der Apache Web-Server basiert auf dem NCSA Web-Server, Version 1.3 und wurde im April 1995 als erste Beta Version veröffentlicht. Innerhalb kürzester Zeit hat er sich zur Nummer 1 unter den im Internet eingesetzten Servern entwickelt. Seit einiger Zeit ist sein Marktanteil - vor allem wegen des kostenlos mit Windows NT ausgelieferten Internet Information Servers von Microsoft - etwas rückläufig. Dennoch ist er noch führend und hat momentan laut Netcraft Survey einen Marktanteil von 44%. Die aktuelle Version ist 1.2.4, es ist jedoch auch schon die Beta Version 1.3b3 erhältlich.

Was den Apache Server auszeichnet, ist die Verfügbarkeit auf fast allen Plattformen (Version 1.3b3 ist auch für Windows NT erhältlich), die hohe Performance bei geringen Hardware Anforderungen (laut c't 10/96 reicht schon ein Intel 486 mit 16MB Speicher für 10 - 100 Clients) und seine Flexibilität. Dieser letzte Punkt wird vor allem durch das modulare Konzept des Apache erreicht. Neben den diversen Modulen die bereits in der Apache Distribution enthalten sind, kann man aus dem Internet noch viele weitere Module für die unterschiedlichsten Anwendungen herunterladen.

Diese Seminarabeit soll Installation, Konfiguration und Administration des Apache Servers beschreiben. Anschließend soll der Server zum Echtbetrieb ins Internet gestellt werden.

2. Installation

Auf dem Webserver wurde von mir S.U.S.E Linux 5.0 mit der Kernelversion 2.0.30 installiert. Bei dem Rechner handelt es sich um einen Intel Pentium 133 mit 32 MB Hauptspeicher und einer 1GB Festplatte. Da sich bei der verwendeten Linux Distribution der Apache Server im Lieferumfang befindet, habe ich ihn mitinstallieren lassen. Leider wird die schon veraltete Version 1.2.0 mitgeliefert, so daß ich das Programm neu übersetzen lassen mußte.

Den Sourcecode habe ich von ftp://ftp.apache.org/dist als gezipptes tar-Archiv (apache_1_2_4.tar.gz) geladen.

Die Installation lief in folgenden Schritten ab:

1. Entpacken des Apache-Archivs:
   gzip -d apache_1_2_4.tar.gz
   tar -xvf apache_1_2_4.tar
2. Bearbeiten der Conifuration Datei
   Eine Modifikation der Flags war nicht notwendig. Von den mitgelieferten Modulen (weitere gibt es unter ftp://ftp.apache.org/apache/dist/contrib/modules) gab ich die mit x gekennzeichneten zur Installation an:

Modul Name	Funktion	installiert
mod_access	Kontrolle der Zugriffe auf Basis von Hostnamen und/oder IP Adressen	x
mod_actions	Zuordnung von CGI-Skripten zu MIME-Typen	x
mod_alias	Anlegen von virtuellen Verzeichnisnamen und Redirect auf andere URLs	x
mod_asis	Auslieferung von Dokumenten ohne automatisch generiertem Header	
mod_auth	Konfiguration von passwortgeschützten Verzeichnissen und Directories	x
mod_auth_anon	Anonymous Zugang mit E-Mail Adresse als Passwort	
mod_auth_db	Authentifikation der Benutzer mittels Berkley DB-Dateien	
mod_auth_dbm	Authentifikation der Benutzer mittels DBM-Dateien	
mod_auth_msql	Authentifikation der Benutzer mittels mSQL Datenbank	
mod_browser	Browserspezifische Konfigurationen	x
mod_cgi	Ermlglicht das Ausführen von CGI Skripten	x
mod_digest	Authentifikation der Benutzer mittels MD5 Digest Authentication	
mod_cern_meta	Ermöglicht Verwendung von CERN Meta-Files	
mod_dir	Kann Verzeichnisindexe erzeugen, wenn keine Index Datei vorhanden ist	x
mod_dld	Dynamisches Binden von Modulen	
mod_env	Übergabe von Umgebungsvariablen an CGI-Skripte und SSI-Dokumente, setzen beliebiger Variablen in der	x

	Server-Konfiguration	
mod_example	Demonstration und Test der API	
mod_expires	Hinzufügen eines Verfalldatums an den Header	
mod_headers	Modifizieren von HTTP-Headern	
mod_imap	Ermöglicht Server-Side Image-Maps	
mod_include	Zur Benutzung von Server-Side Includes (SSI)	x
mod_info	Informationen über Serverinstallation	x
mod_log_agent	Protokollieren der Informationen über Typ und Version der verwendeten Browser	
mod_log_config	Führen von ein oder mehreren Logfiles	x
mod_log_referer	Informationen über Seiten, die auf die eigene URL verweisen	
mod_mime	Generieren der MIME-Header abhängig von der Datei-Endung	x
mod_negotiatian	Funktionen, um Type-Maps und MultiViews zu handhaben	x
mod_proxy	Funktionen eines Proxy-Servers	
mod_rewrite	Ersetzt und erweitert das mao_alias	
mod_status	Informationen über aktuellen Serverstatus	x
mod_userdir	Einbinden von User Directories	x
mod_usertrack	Kontrolle des Userverhaltens mittels Cookies	

3. Aufruf des Shell Skripts Conifgure, das aus den Angaben in der Configuration Datei das Makefile und die Datei modules.c erzeugt
4. Aufrufe von make. Danach ist im Verzeichnis src die ausführbare Datei httpd erzeugt.
5. Anlegenen der Verzeichnisstruktur für den Server und Kopieren der benötigten Dateien. Das Default-Verzeichnis ist /usr/local/etc/httpd.
   Verwendet man wie ich S.u.S.E Linux, muß man sich entscheiden, ob man sich an den Apache oder den S.u.S.E Standard halten will. Ich habe mich für einen leicht modifizierten S.u.S.E Standard entschieden und die Dateien in folgender Verzeichnisstruktur angelegt:

   /usr/local/http/htdocs (Dokument-Root)
   /usr/local/http/cgi-bin (Skriptverzeichnis)
   /usr/local/http/icons (Icons für FancyIndexing)
   /etc/httpd (enthält die Konfigurationsdateien httpd.conf, srm.conf, access.conf und mime.types)
   /var/log (für die Logdateien httpd.access, httpd.error und httpd.script)
   /var/run (enthält die Datei httpd.pid mit der Prozeß ID)
   /var/lock (für das Lock File)

   Leider hat sich im Nachhinein herausgestellt, daß dies keine gute Entscheidung war. Denn der Aufwand, CGI-Scripts an diese Struktur anzupassen ist enorm und zudem fehlerträchtig. Wegen dieser Tatsache würde ich dazu raten, doch als Basisverzeichnis /usr/local/etc/httpd zu verwenden und darunter alle anderen Verzeichnisse anzulegen

6. Aufnahme in das Boot Skript des Rechners
   Bei Linux wird dies z.B. durch ein Shell-Skript in /etc/init.d sowie Links in /etc/init.d/rc2 und /etc/init.d/rc3 realisiert.
   Dabei ist darauf zu achten, daß beim Start die Option -f /etc/httpd/httpd.conf mit angegeben wird, da httpd diese Konfigurationsdatei standardmäßig unter /usr/local/etc/httpd/conf sucht.
7. Setzen der Dateiberechtigungen
   Dies geschieht bei der Installation mit S.U.S.E automatisch. Dabei wird jede Datei unterhalb von /usr/local/httpd dem Benutzer wwwrun und der Gruppe nogroup zugewiesen..

3. Konfiguration

Die Konfiguration gestaltet sich beim Apache recht übersichtlich. Es stehen vier Dateien zur Verfügung, durch die das Serververhalten gesteuert werden kann:

httpd.conf	Die eigentliche Server-Konfiguration
access.conf	Zugriffs-Konfiguration
srm.conf	Resource-Konfiguration
mime.types	Zuweisung von Datei-Endungen zu MIME-Typen. Diese braucht normalerweise nur geändert werden, wenn ein neuer Typ hinzugefügt werden soll.

Wichtig ist, daß nach Veränderungen an diesen Dateien der Web-Server neu gestartet werden muß.

Zusätzlich habe ich noch eine Paßwortdatei angelegt. Diese wird benötigt, um ausgewählten Benutzern Zugriff auf bestimmte Dateien und Verzeichnisse zu gewähren. Erzeugt wird die Datei von dem Programm htpasswd, das sich im support Verzeichnis des Apache befindet. Zusätzlich könnte man auch noch eine Gruppendatei anlegen, durch die Benutzer zu Einheiten zusammengefaßt werden können. Da bisher allerdings noch keine Notwendigkeit dazu besteht (es existiert nur ein ausgezeichneter User "webmaster"), habe ich darauf verzichtet.

3.1. Einstellungen in httpd.conf

Abhängig von den eingebundenen Modulen stehen eine Vielzahl von Einträgen zur Verfügung. Da ein Aufzählen aller dieser Möglichkeiten ein 200 Seiten Buch füllen würde, stelle ich nur die von mir verwendeten Parameter vor.

ServerType standalone Listen 80	Es wird festgelegt, daß der Server beim Starten selbst den mit Listen definierten Port (Standard ist 80) öffnet und auf Anfragen wartet. Falls mehrere Interfaces bzw. IP Adressen für den Rechner vorhanden sind, kann mit Listen festgelegt werden, welche Ports für welche Adresse abgehört werden sollen
User wwwrun Group nogroup ServerAdmin piet.guertler@bkr-regensburg.de	Nach dem öffnen des Ports (als root), wechselt Apache zu der als User und Group angegebenen UID bzw. GID. Die Zuweisung kann entweder durch angeben des Namens oder durch # gefolgt von der ID erfolgen Bei Fehlermeldungen wird diese Adresse dem Client zurückgeliefert.
ServerName www.bkr-regensburg.de	Wird bei Redirects benötigt (wenn man beispielsweise durch Eingabe von http://www.bkr-regensburg.de/inhalt auf das Verzeichnis inhalt zugreifen will, erzeugt der Apache einen Redirect auf http://www.bkr-regensburg.de/inhalt/ . Ohne ServerName würde der richtige Hostname des Rechners zurückgeliefert
ServerRoot /usr/local/httpd	Diese Einstellung sorgt dafür, daß Apache ausgehend von diesem Verzeichnis die

	Verzeichnisse für Dokumente, Logfiles und Konfigurationsdateien sucht. Da unter Linux diese Verzeichnisse nicht alle unter /usr/local/httpd liegen, müssen sie gegebenenfalls durch vollständige Dateipfade angegebene werden.
DocumentRoot /usr/local/httpd/htdocs AccessConfig /etc/httpd/access.conf	Hier werden die HTML Seiten abgelegt (eigentlich hätte hier htdocs als Angabe gereicht. Zur Übersichtlichkeit habe ich hier jedoch den vollständigen Pfad angegeben) Datei zur Definition der Funktionen und Zugriffsrechte bestimmter Verzeichnisse
ResourceConfig /etc/httpd/srm.conf	Diese Datei enthält Angaben zur Formatierung und dem Aussehen der automatisch erzeugten Verzeichnisindizes (FancyIndexing) sowie Angaben zu den einzelnen Dateitypen.
TypesConfig /etc/httpd/mime.types PidFile /var/run/httpd.pid	Enthält die Zuweisungen Dateiendung zu Mime-Typ Datei, in der sich die Prozeß ID des Httpd-Daemons befindet
LockFile /var/locks/httpd.lock	Spezifizert das Lock File
ErrorLog /var/log/httpd.error	Datei zur Aufnahme von Fehlermeldungen und sonstigen Info Meldungen (mod_log_config)
ScriptLog /var/log/httpd.script CustomLog /var/log/httpd.access "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""	Datei zum Debuggen von CGI-Skripten (mod_cgi) Erzeugt die Datei httpd.access im Combined Log Format (Hostname, Remote Logname, User ID, Zeit, erste Zeile der Client-Anfrage, HTTP-Statuscode, Größe der vom Server ausgelieferten Datei (ohne Header), Wert des Referer Headers, Wert des User-Agent Headers)
Timeout 300	Zeitlimit für Client-Anfragen in Sekunden
StartServers 5 MaxClients 150	Es werden automatisch 5 Kindprozesse des Apache gestartet Maximum von gleichzeitig laufenden Server-Prozessen
MaxRequestsPerChild 30	Nach 30 Anfragen wird der Kindprozeß beendet und gegebenenfalls ein neuer gestartet.
MinSpareServers 5 MaxSpareServers 10	Es existieren immer mindestens 5 und höchstens 10 leerlaufende Prozesse
KeepAlive On	Unterstützung der HTTP / 1.1 Persistent Connections. Dadurch können über eine TCP-Verbindung mehrere Anfragen an den Server geschickt werden.
MaxKeepAliveRequests 10	Es werden 10 aufeinanderfolgende Anfragen innerhalb einer Keep-Alive-Verbindung erlaubt

KeepAliveTimeout 15	Der Server wartet maximal 15 Sekunden auf weitere Anfragen des Clients bei einer Keep-Alive-Verbindung
IdentityCheck Off	Apache soll keinen IDENT-Lookup ausführen
HostnameLookups Off	Es sollen keine DNS-Lookups ausgeführt werden
ContentDigest On BrowserMatch ^Mozilla/2 nokeepalive BrowserMatch Java/1.0 force-response-1.0 BrowserMatch JDK/1.0 force-response-1.0 BrowserMatch "RealPlayer 4.0" force-response-1.0	Es wird ein Content-MD5 Header erzeugt und an den Client zurückgeschickt Hiermit wird Apache angewiesen, auf bestimmte Browser unterschiedlich zu reagieren

3.2. Einstellungen in access.conf

<Directory /> Options None AllowOverride None Order deny,allow Deny from all </Directory>	Im root-Verzeichnis werden keine Konfigurationsanweisungen in der Datei .htaccess beachtet und es wird kein Zugriff auf irgendwelche Unterverzeichnisse gestattet
<Files ~ "(\.htaccess|/core|~|#|\.bak|\.BAK)$"> order deny,allow deny from all </Files>	Es wird kein Zugriff auf Dateien gestattet die .htaccess heißen, sich in einem Verzeichnis core befinden oder ~ bzw. # enthalten oder mit .bak bzw. .BAK enden
<Location /> Otions SymLinksIfOwnerMatch IncludesNOEXEC order allow,deny allow from all AuthType Basic AuthUserFile /etc/httpd/passwd </Location> <Directory /usr/local/httpd/cgi-bin> Options ExecCGI Order allow,deny Allow from all </Directory>	Für /usr/local/httpd/htdocs und alle Unterverzeichnisse wirde festgelegt, daß symbolische Links verfolgt werden dürfen, wenn sie dem Benutzer (wwwrun) gehören und daß Include Anweisungen ausgeführt werden dürfen (außer exec). Da die Digest Authentikation von den meisten Browsern noch nicht unterstützt wird, wird Basis Authentikation verwendet. Im Verzeichnis cgi-bin können CGI-Skripts ausgeführt werden

<Location /intern/server-status> SetHandler server-status AuthName server-status require user webmaster </Location>	Es wird zugelassen, daß der Benutzer webmaster sich den aktuellen Server Status anzeigen lassen kann
<Location /intern/server-info> SetHandler server-info AuthName server-info require user webmaster </Location> <Location /intern/analyse> AuthName analyse require user webmaster </Location>	Es wird zugelassen, daß der Benutzer webmaster sich die aktuellen Server Infos anzeigen lassen kann In diesem Verzeichnis liegt die Startseite des Logfile-Analyseprogramms analog.
<Location /guertler/test-cgi> Options ExecCGI order allow,deny allow from all </Location> <Location /guertler/test-typemap> AddHandler type-map var </Location> <Location /guertler/test-multiviews> Options +MultiViews </Location> <Location /guertler/test-info> SetHandler server-info </Location> <Location /guertler/test-status> SetHandler server-status </Location> <Location /guertler/test-index> Options +Indexes </Location>	Dies sind meine Testverzeichnisse, die zur Demonstration der verschiedenen Möglichkeiten des Apache gebraucht werden.

3.3. Einstellungen in srm.conf

Alias /icons /usr/local/httpd/icons Alias /cgi-bin /usr/local/httpd/cgi-bin Alias /images /usr/local/httpd/images	Aliasname für bestimmte Directories. Diese werden anschließend behandelt, wie wenn sie sich unter htdocs befinden würden
UserDir public_html	Benutzer können in ihren home Verzeichnissen einen Ordner public_html anlegen und so Seiten veröffentlichen
DirectoryIndex index.html index.htm	Wird nur ein Directory angegeben, wird automatisch nach einer Datei index.htm bzw. index.html gesucht und angezeigt
IndexOptions FancyIndexing	Wird keine Index Datei gefunden, wird der Inhalt des Verzeichnisses als Verzeichnisbaum angezeigt
IndexIgnore .??* *~ *# *.bak *.BAK HEADER.* README.* RCS core	Diese Dateien werden nicht durch FancyIndexing angezeigt
ReadmeName README	Die Datei README wird nach dem Verzeichnisbaum angezeigt (also vor </body>
HeaderName HEADER	Die Date HEADER wird direkt nach dem <body> Tag in den Verzeichnisindex eingefügt
AccessFileName .htaccess	Jedes Verzeichnis kann eine Datei .htaccess enthalten, mit der (falls erlaubt) der Inhalt von access.conf überschrieben werden kann
DefaultType text/plain	Dateien ohne Endung werden wie Text-Files behandelt
AddType text/html shtml	Dateien mit der Endung shtml wird der Mime Typ text/html zugeordnet
AddHandler server-parsed shtml AddHandler cgi-script cgi AddHandler type-map var	Weist Dateien mit bestimmten Endungen Handler zu
AddIcon (SRC,/icons/c.gif) .c .h AddIcon(SCR,/icons/p.gif) .pl .py AddIcon (FOR,/icons/f.gif) .for AddIcon (UUE,/icons/uuencoded.gif) .uu AddIcon (SCR,/icons/script.gif) .conf .config .skel .cf .mc .in .ksh Makefile AddIcon (COR,/icons/bomb.gif) core AddIcon (PAT,/icons/patch.gif) .patch AddIcon (TXT,/icons/hand.right.gif) README AddIcon (CGI,/icons/link.gif) .cgi AddIcon (WRL,/icons/world1.gif) wrl.gz wrl.Z AddIcon (POS,/icons/ps.gif) ps.gz ps.Z AddIcon (DVI,/icons/dvi.gif) dvi.gz dvi.Z AddIcon /icons/back.gif .. AddIcon /icons/folder.gif ^^DIRECTORY^^ AddIcon /icons/blank.gif ^^BLANKICON^^	Für das Fancy Indexing werden gewissen Endungen Icons zugewiesen
AddEncoding x-compress Z	

AddEncoding x-gzip gz tgz AddIconByEncoding (CMP,/icons/compressed.gif) x-gzip x-compress
AddType x-world/x-vrml vrm
AddIconByType (BIN,/icons/binary.gif) application/octet-stream AddIconByType (HQX,/icons/binhex.gif) application/mac-binhex40 AddIconByType (TAR,/icons/tar.gif) application/x-gtar application/x-tar AddIconByType (CMP,/icons/compressed.gif) application/zip application/x-stuffit AddIconByType (POS,/icons/ps.gif) application/postscript AddIconByType (PDF,/icons/pdf.gif) application/pdf AddIconByType (HTM,/icons/layout.gif) text/html AddIconByType (DVI,/icons/dvi.gif) application/x-dvi AddIconByType (SCR,/icons/script.gif) application/x-csh application/x-tcl application/x-shar application/x-sh AddIconByType (TEX,/icons/tex.gif) application/x-latex application/x-tex application/x-texinfo
AddIconByType (PPT,/icons/sphere2.gif) application/powerpoint AddIconByType (DOC,/icons/quill.gif) application/msword AddIconByType (RTF,/icons/text.gif) application/rtf AddIconByType (XDR,/icons/portal.gif) application/x-director AddIconByType (TRF,/icons/text.gif) application/x-troff application/x-troff-man application/x-troff-me application/x-troff-ms
AddIconByType (WAL,/icons/script.gif)

```
application/x-wais-source

AddIconByType (ICE,/icons/sound1.gif) x-
conference/*

AddIconByType (WRL,/icons/world2.gif) x-
world/*

AddIconByType (TXT,/icons/text.gif) text/*

AddIconByType (IMG,/icons/image2.gif)
image/*

AddIconByType (SND,/icons/sound2.gif)
audio/*

AddIconByType (VID,/icons/movie.gif)
video/*
```

```
DefaultIcon /icons/unknown.gif
```

```
AddDescription "Hypertext document" .html
.htm
AddDescription "Textfile" .txt
AddDescription "GIF image" .gif
AddDescription "JPEG image" .jpeg .jpg
AddDescription "PNG image" .png
AddDescription "Tape archive" .tar
AddDescription "ZIP archive" .zip
```

```
AddLanguage de .de # German
AddLanguage en .en # English
```

Angaben für ContentNegotiatian

```
LanguagePriority de en
```

Falls Dokumente in zwei Sprachversionen vorhanden sind, wird das deutsche Dokument an den Client gesandt

3.4. Einstellungen in mime.types

Zuordnung von Datei Endungen zu Mime Typen

```
application/activemessage
application/andrew-inset
application/applefile
application/atomicmail
application/dca-rft
application/dec-dx
application/mac-binhex40 hqx
application/mac-compactpro cpt
application/macwriteii
application/msword doc
application/news-message-id
application/news-transmission
application/octet-stream bin dms lha lzh
exe class
application/oda oda
application/pdf pdf

application/postscript ai eps
application/powerpoint ppt
```



```
application/remote-printing
application/rtf rtf
application/slate
application/wita
application/wordperfect5.1
application/x-bcpio bcpio
application/x-cdlink vcd
application/x-compress
application/x-cpio cpio
application/x-csh csh
application/x-director dcr dir dxr
application/x-dvi dvi
application/x-gtar gtar
```

```
application/x-gzip
application/x-hdf hdf
application/x-koan skp skd skt skm
application/x-latex latex
application/x-mif mif
application/x-netcdf nc cdf
application/x-sh sh
application/x-shar shar
application/x-stuffit sit
application/x-sv4cpio sv4cpio
application/x-sv4crc sv4crc
application/x-tar tar
application/x-tcl tcl
application/x-tex tex
application/x-texinfo texinfo texi

application/x-troff t tr roff
application/x-troff-man man
application/x-troff-me me
application/x-troff-ms ms
application/x-ustar ustar
application/x-wais-source src
application/zip zip
```

```
audio/basic au snd
audio/midi mid midi kar
audio/mpeg mpga mp2
audio/x-aiff aif aiff aifc
audio/x-pn-realaudio ram
audio/x-pn-realaudio-plugin rpm
audio/x-realaudio ra
audio/x-wav wav
```

```
chemical/x-pdb pdb xyz
```

```
image/gif gif
image/ief ief
image/jpeg jpeg jpg jpe
image/png png
image/tiff tiff tif
image/x-cmu-raster ras
image/x-portable-anymap pnm
image/x-portable-bitmap pbm
image/x-portable-graymap pgm
image/x-portable-pixmap ppm
image/x-rgb rgb
```

image/x-xbitmap xbm image/x-xpixmap xpm image/x-xwindowdump xwd
message/external-body message/news message/partial message/rfc822
multipart/alternative multipart/appledouble multipart/digest multipart/mixed multipart/parallel
text/html html htm text/plain txt text/richtext rtx text/tab-separated-values tsv text/x-setext etx text/x-sgml sgml sgm
video/mpeg mpeg mpg mpe video/quicktime qt mov video/x-msvideo avi video/x-sgi-movie movie
x-conference/x-cooltalk ice
x-world/x-vrml wrl vrml

4. HTTP-Daemon

4.1. Funktionsprinzip

Die aktuellen Versionen des Apache arbeiten nach dem folgenden Prinzip: Es wird ein httpd-Hauptprozeß als root gestartet, der sich an alle ihm durch Port oder Listen zugewiesene Ports (normalerweise 80 oder 443 bei SSL) bindet. Anschließend erzeugt dieser Hauptprozeß Kindprozesse unter der eingestellten UID und GID, die die Client-Anfragen behandeln. Mit der MinSpareServer und MaxSpareServers kann man bestimmen, wieviele unbeschäftigte Kindprozesse in Reserve gehalten werden (pre-forking). Mit der Anweisung MaxClients kann man festlegen, wieviele Kindprozesse maximal gleichzeitig gestartet werden können. Per Default liegt das Maximum der gleichzeitigen Kindprozesse bei 256. Will man diesen Wert erhöhen, kann man dies beim Compilieren über die Compiler-Option HARD_SERVER_LIMIT einstellen.

Vorteil dieser Methode ist die schnelle Reaktion des Servers auf Clientanfragen. Nachteil ist der große Speicherplatzbedarf. Hat man das Maximum an gleichzeitigen Kindprozessen zu hoch eingestellt, so daß der Hauptspeicher nicht ausreicht um alle im Speicher zu halten, muß geswappt werden, was natürlich die Performance drastisch verringert.

In Zukunft (vermutlich ab Version 2.0) wird der Apache nicht mehr pre-forking verwenden sondern Multithreading. Einen Vorgeschmackt darauf bietet die Version 1.3b für Windows NT, die bereits heute nach diesem Prinzip arbeitet. Dies wird auch mehr Sicherheit bieten, denn es können nach wie vor mehrere Prozesse gestartet werden. Fällt ein Prozeß aus, kann der nächste dafür einspringen. Momentan wirkt es noch fatal, wenn der Hauptprozeß ausfällt.

4.2. Starten

Es ist immer wieder nötig, den httpd-Daemon neu zu starten, z.B. wenn man Veränderungen an der Konfigurationsdatei httpd.conf 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. Mit einer Ausnahme: wenn Änderungen an einem Logfile-Format vorgenommen wurden ist; denn solange noch aktive Verbindungen bestehen, benutzen diese das alte Logfile.

Was mir anfangs häufig passierte, waren fehlerhafte Einträge in der Datei httpd.conf. In diesem Fall gibt der Server eine Fehlermeldung in der Shell aus und wird nicht gestartet.

Etwas nervig ist, daß beim Starten immer die Option -f /etc/httpd/httpd.conf angegeben werden muß. Wenn ich den Apache das nächste Mal übersetze, werde ich deshalb entweder dieses Verzeichnis mit den Optionen HTTPD_ROOT und SERVER_CONFIG_FILE hardcodieren oder doch den Apache Vorschlag für die Verzeichnisstruktur übernehmen.

4.3. Beenden

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`

5. Wichtige Funktionalitäten

Je nach installierten Modulen, besitzt der Apache unterschiedliche Funktionen. Drei der meiner Meinung nach wichtigsten habe ich herausgenommen um sie etwas ausführlicher vorzustellen. Diese sind:

- Common Gateway Interface
- Server Side Includes und
- Content Negotiation

5.1. Common Gateway Interface (CGI)

Der Apache bietet verschiedene Möglichkeiten, CGI-Skripts auszuführen. Zum einen kann man mit der Anweisung Options ExecCGI für ganze Verzeichnisse festlegen, daß darin Skripts ausgeführt werden können (unabhängig von ihrer Endung). Andererseit kann man aber auch durch SetHandler cgi-script bestimmen, daß in bestimmten Verzeichnissen Programme mit einer vorher festgelegten Endung (z.B. .cgi) ausgeführt werden können.

Da ich Testrechner nicht nur einmal durch Fehler in Skripts zum Absturz gebracht habe, und Skripts auch eine große Sicherheitslücke sein können, habe ich beschlossen, sehr restriktiv mit Freigaben umzugehen. Es gibt nur zwei Verzeichnis, in denen Skripts erlaubt sind: /cgi-bin und /guertler/test-cgi. Außerdem verwende ich nur Skripts, deren Quellcode mir zugänglich ist und die ich vorher ausgiebig auf Testrechnern ausprobiert habe. Die Möglichkeit, durch den suEXEC-Wrapper Benutzer Skripts unter deren ID ausführen zu lassen nutze ich nicht.

Da ich überwiegend Perl Skripts verwende, werde ich bei der nächsten Installation des Apache das Modul mod_perl mitcompilieren, da die Ausführungsgeschwindigkeit dadurch, daß der Perl Interpreter ständig im Speicher gehalten wird um angeblich mehrere 100% erhöht werden kann.

Momentan habe ich auf dem installierten Server ein C-Skript zur Auswertung der Logfiles laufen (analog).

5.2. Server-Side Include (SSI)

Durch SSI können vom Apache Dokumente vor Auslieferung an den Client auf Include-Anweisungen hin untersucht werden. Sinnvoll ist es, diesen Dateien eine bestimmte Endung zu geben, wie z.B. .shtml, damit nicht automatisch jedes Dokument untersucht wird. Die Syntax für eine Include Anweisung sieht folgendermaßen aus:

<!--#*Element Attribut*="*Wert*" *Attribut*="*Wert*" . . . -->

Ein nützliches Beispiel hierzu ist das Einfügen von Dateiinformationen, wann also beispielsweise das Dokument das letzte Mal verändert wurde. Dies sieht dann so aus

<!--#flastmod file="Dateiname" -->

Es besteht auch die Möglichkeit, ganze Dateien einzufügen. Dies kann sehr nützliche sein, wenn beispielsweise auf einer Seite mit Neuigkeiten an bestimmten Stellen Textdateien eingefügt werden. Diese können dann von Leuten erstellt werden, die des HTML nicht mächtig sind. Außerdem vermeidet man dadurch Fehler in der Originaldatei.

```
<--#include file="daten.txt" -->
```

Eine weitere Möglichkeit ist das Einbinden von Informationen, die von einem Skript zurückgeliefert werden. Oft wird dies für Counter benutzt, die zeigen sollen wie oft auf die Seite schon zugegriffen worden ist.

```
<--#include virtual="/cgi-bin/counter.cgi" -->
```

Eine äußerst gefährliche Art der Includes wird mit dem exec-Element angeboten. Hiermit lassen sich beliebige Skripts oder Shell-Kommandos ausführen also beispielsweise auch:

```
<--#exec cmd=/bin/rm -rf /" -->
```

Daß in einem Verzeichnis Dokumente mit Include Anweisung auch tatsächlich untersucht werden, wird mit der Anweisung Options +Includes mitgeteilt. Allerdings, wie das letzte Beispiele zeigt, können dadurch Sicherheitslücken entstehen, die nicht verantwortbar sind. Deshalb verwende ich immer Options +IncludesNOEXEC und verhindere somit das Ausführen von Programmen.

5.3. Content Negotiation

Der Apache kann eine Antwort den Wünschen und Fähigkeiten des Clients anpassen, wenn verschiedene Versionen eines Dokuments oder einer Grafik vorhanden sind. Mitgeteilt wird dies vom Web-Client über vier verschiedene HTTP-Header: Accept (MIME-Typ), Accept-Charset (Zeichensatz), Accept-Encoding (Kodierung) und Accept-Language (Sprache).

Es existieren auf Apache Seite zwei Möglichkeiten der Content Negotiation: Type-Maps und Multiviews-Suche.

Type-Maps

Hat man durch die Anweisung AddHandler type-map var für eine bestimmte Location oder ein Directory Type-Maps aktiviert, muß man manuelle eine Type-Map Datei erstellen. Diese kann folgende Angaben enthalten:

- URI enthält Angaben zur Position der gewünschten Datei. Dies geschieht immer relativ zum aktuellen Verzeichnis. Also z.B. URI: test.htm oder URI: ../test.htm
- CONTENT-TYPE gibt den MIME-Typ und durch Semikolon getrennt eventuell noch einen Qualitätswert "qs" (quality score) an. Zusätzlich kann als weiterer Parameter noch der in der Datei verwendete Zeichensatz angegeben werden. Beispiel: Content-Type: text/html; qs=0.5; charset=iso-8859-5
- CONTENT-LENGTH gibt die Größe der zugehörigen Datei in Bytes an. Dieser Eintrag ist überflüssig, da Apache diesen Wert selbständig ermittelt und auch hierdurch falsch eingetragene Werte ignoriert.
- CONTENT-LANGUAGE macht die im Dokument verwendete Sprache kenntlich, z.B. Content-Language: de
- CONTENT-ENCODING zeigt an, mit welchem Kodierungsprogramm eine Datei verschlüsselt oder gepackt ist, z.B. Content-Encoding: gzip
- DESCRIPTION wird an den Client mit der Fehlermeldung geschickt, wenn keine passende Variante gefunden wurde und der Benutzer auswählen muß, welche Variante er nehmen will. Beispiel: Description: "Dies ist das Dokument in deutscher Sprache"

Ich habe folgende Beispiel Type-Map Datei im Verzeichnis guertler/test-typemap angelegt:

```
#
# type-map: /guertler/test-typemap/test.var
#

URI: test.de.htm
Content-Type: text/html; qs=0.6
Content-Language: de
Description: "Deutsche HTML-Version"

URI: test.en.txt
Content-Type: text/plain; qs=0.2
Content-Language: en
Description: "Englische HTML-Version"

URI: test.txt
Content-Type: text/plain; qs=0.4
Content-Language: de
Description: "Deutsche TXT-Version"
```

Je nachdem, welche Einstellungen der Client getroffen hat, wird ihm das Dokument übermittelt, das die größe Übereinstimmung aufweist. Bevorzugt er Dokumente in englischer Sprache, wird auf jeden Fall test.en.htm übermittelt. Bevorzugt er deutsche Dokumente, wird standardmäßig test.de.htm ausgeliefert, außer er bevorzugt reine Textdateien.

Diese Möglichkeit der Type-Maps ist sehr flexibel, allerdings auch extrem aufwendig. Leichter ist es mit

Multiviews

Hier übernimmt der Apache den Aufbau der Type-Maps zur Laufzeit. Auch diese Funktion muß durch Option +MultiViews aktiviert werden. Greift anschließend ein Client auf ein aktiviertes Verzeichnis zu also z.B. /guertler/test-multiviews/test.htm, sucht der Apache nach allen Dokumenten test.htm.* und erstellt basierend auf dem Suchergebnis eine Type-Map. Am interessantesten ist hier wahrscheinlich die Anwendung zum Ausliefern von Dokumenten in verschiedenen Spachversionen.

6. Überwachung

Läßt man das Pflegen der Inhalte außen vor, so bereitet die Überwachung des Servers den größten Aufwand. Zum einen muß sichergestellt sein, daß der Server immer einwandfrei läuft, zum anderen sollen eventuelle Eindringlinge möglichst frühzeitig entdeckt werden.

- Sicherheit
- Logfile Analyse
- Serverinformationen

6.1. Sicherheit

Der wichtigste Satz zum Thema Sicherheit lautet: Der Server kann nur so sicher sein wie das Betriebssystem unter dem er läuft. Sobald ein Hacker Zugriff zum Filesystem erlangt, kann natürlich auch der Apache Server ausgehebelt werden. Es sollte deshalb - selbstverständlich unter Abwägung von Aufwand und Schutzwürdigkeit der Daten - ein stabiles Serverbetriebssystem ausgewählt werden. Im Fall des von mir installierten Servers, bei dem keine sicherheitsrelevanten Daten verwaltet werden, habe ich mich für Linux entschieden. Würde aber auf dem selben Server noch eine Datenbank mit z.B. Patienten- oder Personaldaten liegen, würde ich ein C2 zertifiziertes Unix System wie SCO verwenden. Von Windows NT (abgesehen davon, daß der Apache hier bisher nur als Beta Version verfügbar ist) wurde mir von den befragten Internet Providern (ECRC, SpaceNet) wegen des unvergleichlich höheren Ressourcenverbrauchs und größerer Fehleranfälligkeit abgeraten.

Nach Auswahl des Betriebssystems sollten alle unnötige Dienste vom Rechner entfernt werden und gegebenefalls ein Router als Packetfilter vorgeschaltet werden. Anschließend kann man daran gehen, das Filesystem zu sichern. Die UID unter der Apache läuft darf keinesfalls Schreibrechte besitzen außer an Stellen, wo dies explizit gewünscht ist (beispielsweise zum erstellen von Lock-Files bei CGI-Skripts).

Als praktisch hat er sich herausgestellt, wenn man bei den Verzeichnissen htdocs und cgi-bin das s-Bit setzt, da somit alle Dateien und Verzeichnisse die hier angelegt werden automatisch der Gruppe nogroup zugewiesen werden. Somit ist sichergestellt, daß auch alle Dateien für den Web-Server lesbar sind. Als umask sollte man beim Anlegen von Dateien und Verzeichnissen 027 verwenden. Dateien werden dadurch mit dem Modus 640 und Verzeichnisse mit 750 angelegt. Die Dateien sollten keine Lese- oder Schreibrechte für other besitzen, damit sie gesichert sind, falls ein Benutzer über das Filesystem auf sie zugreifen will.

Das Verzeichnis /etc/http muß für den Apache lesbar sein, wenn man wie ich eine Paßwortdatei in diesem Verzeichnis verwendet oder das Modul mod_info benutzt.

6.2. Logfile Analyse

Ich verwende 3 Logfiles: http.access, http.error und http.script. Wie der Namen schon andeuten werden in httpd.access alle Seitenzugriffe gespeichert. Das Format dieses Files wurde in httpd.conf festgelegt. (CustomLog /var/log/httpd.access "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"). Eine Zeile in diesem Logfile sieht nun beispielsweise so aus:

132.199.208.62 - - [17/Dec/1997:18:09:37 +0100] "GET /bilder/logobkr.gif HTTP/1.1" 200 4283 "http://www.bkr-regensburg.de/schalt.htm" "Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)"

132.199.208.62	Ist die IP Adresse des zugreifenden Clients. Hat man HostnameLookup aktiviert wird der DNS Name dieses Rechners zurückgeliefert (was aber die Performance beträchtlich verringert).
-	Ist der IdentityCheck aktiviert, erhält man hier das Ergebnis einer IDENT Anfrage.
-	Hätte eine Authentikation stattgefunden, würde hier die UID zurückgeliefert
[17/Dec/1997:18:09:37 +0100]	Datum und Uhrzeit des Zugriffs
"GET /bilder/logobkr.gif HTTP/1.1"	Erste Zeile der Client-Anfrage
200	HTTP-Statuscode (in diesem Fall "o.k.")
4283	Die Größe der vom Server ausgelieferten Datei in Bytes (ohne Header)
"http://www.bkr-regensburg.de/schalt.htm"	Seite von der aus die Anfrage gestartet wurde
"Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)"	Der verwendete Browser

Es ist sinnlos zu versuchen, diese Files "per Hand" auszuwerten. Dazu gibt es mehrere Hilfsprogramme, die teilweise auch als CGI-Skripts funktionieren. Ich habe beispielsweise das Programm analog auf dem Server installiert.

Das Logfile httpd.error enthält Meldungen über fehlgeschlagene Aktionen. Ein typischer Eintrag sieht so aus:

[Wed Dec 17 16:31:28 1997] access to /usr/local/httpd/htdocs/bilder/putz.jpg failed for 132.199.128.203, reason: File does not exist

Damit kann man schnell feststellen, ob sich auf dem Server tote Hyperlinks befinden. Außerdem wird jedes Hoch- und Herunterfahren des Servers in dieser Logdatei aufgezeichnet.

[Wed Dec 17 15:26:24 1997] httpd: caught SIGTERM, shutting down
[Wed Dec 17 15:27:17 1997] created shared memory segment #0
[Wed Dec 17 15:27:17 1997] Server configured -- resuming normal operations

Das dritte von mir verwendete Logfile heißt httpd.script und enthält Fehlermeldungen von CGI-Scripts. Dies kann äußerst nützlich sein, da ansonsten meist nur eine Meldung der Art "Es ist einschwerer Fehler aufgetreten" erscheint.

6.3. Serverinformationen

Durch das Modul mod_info hat man die Möglichkeit, sich aktuelle Server Informationen anzeigen zu lassen. Voraussetzung dafür ist, daß man in der Datei access.conf einem URL per SetHandler den handler server-info zuweist.

```
<Location /guertler/test-info>
SetHandler server-info
</Location>
```

Ruft man diese URL nun auf, wird dynamisch eine HTML-Seite generiert, die Informationen wie Server Settings, Server Version, API Version usw. anzeigt.

Ebenso verhält es sich mit dem Server Status. Dieser wird durch das Modul mod_status und dem Setzten des Handlers server-status in einer Location oder Directory Sektion der access.conf erzeugt.

```
<Location /guertler/test-status>
SetHandler server-status
</Location>
```

Zu testen sind beide von der test.htm Seite aus.

7. Zusammenfassung

Der Apache ist ein Server, der bis ins kleinste Detail den Benutzerwünsche angepaßt werden kann. Dies hat aber gleichzeitig zur Folge, daß man sich tief einarbeiten muß, will man eine optimale Konfiguration erreichen. Anfangs ist man erschlagen von den vielen Modulen und Parametern die in den verschiedensten Zusammenhängen eingesetzt werden können. Die Beispiel Konfigurationsdateien, die mit dem Apache ausgeliefert werden helfen zwar, den Rechner zum Laufen zu bringen. Aber die Praxis zeigt, daß diese Einstellungen bei weitem nicht ausreichen.

Ich komme nicht aus der Unix Welt und beschäftige mich auch erst seit dem Softwarepraktikum mit Linux. Dies hat sich als extrem nachteilig herausgestellt. Ich habe sehr viel Zeit damit zugebracht, herauszufinden, wie der Server "per Hand" installiert wird. Hätte ich nicht die von S.u.S.E mitgelieferte Apache Version installiert, wüsste ich vermutlich bis heute noch nicht, wie der Server beim Booten gestartet werden muß. Auch die Aufteilung in die diversen Verzeichnisse hat mich erheblichen Suchaufwand gekostet.

Einem Windows NT Benutzer wie mir fehlt das graphische Benutzerinterface. Und auch die Installation ist erheblich aufwendiger als die des Internet Information Servers (der Nummer 2 auf dem Web-Server Markt). In punkto Betriebssicherheit und Leistung bin ich aber mittlerweile der Meinung, daß der IIS dem Apache keinesfalls das Wasser reichen kann. Dies ist natürlich keine wissenschaftliche Aussage, sondern beruht auf meinem subjektiven Empfinden (als Intranetserver verwalte ich einen IIS 3.0 unter Windows NT auf einem Rechner der gleichen Bauart).

Ich habe jetzt einige Monate mit dem Apache gearbeitet und kann nicht behaupten, daß ich dadurch schon zum Profi geworden bin. Allerdings traue ich mir jetzt zu, diesen Server auch im Echtbetrieb zu administrieren.

Als Literatur habe ich fast ausschließlich das Buch "Apache Web-Server" von Lars Eilebrecht, erschienen 1997 bei Internat. Thomson Publ. verwendet, sowie die beim Apache mitgelieferten Online Dokumente.