6 Informationen per Web-Server verteilen

Immer mehr Einrichtungen nutzen Web-Server, um Informationen im Intranet, Extranet und Internet bereitzustellen. Dies beeinflusst die gesamte Kommunikationskultur erheblich.

Wichtige technische Grundlagen des World Wide Webs und von Intranets sind:

- HyperText Markup Language (*HTML*), die Sprache der Web-Seiten.
- HyperText Transfer Protocol (*HTTP*), das die Seitenanforderungen und Übertragungen regelt,
- Uniform Resource Locator (*URL*), die eindeutige Adresse für eine Information im Internet und
- inzwischen immer mehr die Extended Markup Language (*XML*) für universelle, Web-basierte Kommunikation.

Alle marktführenden Linux-Distributionen enthalten einen Web-Server, meist den *Apache*. Da SuSE ihn bei der Standardinstallation nicht automatisch installiert, holen Sie das am besten schnell nach.

Apache hat nichts mit dem gleichnamigen Indianerstamm gemein, sondern verballhornt die Wortkombination *a patchy server*. Die Wurzeln des Apache liegen in Anpassungen (*Patches*) des NCSA Web-Servers. Inzwischen entwickelt eine Gruppe von etwa 20 Programmierern, die *Apache HTTP Server Group*, Apache eigenständig für Linux und Windows NT weiter.

Dieses Kapitel beschreibt die Grundlagen, um Apache sinnvoll im lokalen Netz einzusetzen.

Der Web-Server Apache erlaubt es, Seiten nur für geschlossene Benutzergruppen zugänglich zu machen. Nur wer über einen geeigneten Benutzernamen und das zugehörige Passwort verfügt, kann dann auf die geschützten Seiten zugreifen.

Da Browser Benutzernamen und Passwort normalerweise unverschlüsselt an den Web-Server übertragen, was ein unnötiges Sicherheitsrisiko darstellt, sollten Sie die Datenübertragung verschlüsseln.

Lesen Sie in diesem Kapitel, wie

- Web-Server arbeiten (6.2),
- man Apache installiert und einrichtet (6.3),

- man das Einrichten und Pflegen von Web-Inhalten organisatorisch löst (6.4),
- man eine Zugriffssteuerung für geschlossene Nutzergruppen einrichtet (6.5),
- virtuelle Server zu verstehen sind (6.6),
- man gesicherte Zugriffe mit Secure Sockets Layer (*SSL*) einrichtet (6.7),
- man Web-Server-Zugriffe protokolliert (6.8),
- Sie die Protokolldatei des Web-Servers grafisch aufbereiten (6.9) und
- man eine eigene Suchmaschine einrichten (6.10) kann.

6.1 Wann benötigen Sie einen eigenen Web-Server?

Einen eigenen Web-Server benötigen Sie eigentlich immer. Statt Informationen auf einem schwarzen Brett in der Kantine auszuhängen oder Kunden per Mailing zu informieren, kann man besser Seiten für den lokalen Web-Server erstellen und dort aktuelle Ankündigungen und Termine hinterlegen. Wichtig ist, die Inhalte regelmäßig zu pflegen und zu aktualisieren.

Hierzu verwendet man am besten Content Management-Systeme. Freie Content Management-Systeme sind u. a. *Midgard* (`http://www.midgard-project.org`) und `plone (http://www.plone.org)`. Beim Entwickeln von Web-Auftritten können jedem leicht Fehler unterlaufen. Peinlicherweise sind diese bei über das Internet zugänglichen Seiten weltweit sichtbar. Den eigenen Auftritt sollte man daher zuerst im lokalen Netz entwickeln und testen, um sich Blamagen zu ersparen.

6.2 So arbeiten Web-Server

Beim HyperText Transfer Protocol (*http*) sendet der Client, der Web-Browser, eine Anfrage nach einem Dokument an den Server, den http-Dämon. Dieser liefert dem Client den MIME-Typ der angeforderten Datei sowie die Datei selbst. Aus dem MIME-Typ schließt der Client, was er mit den empfangenen Daten anfangen soll.

Die häufigsten MIME-Typen zeigt er so an:

- `text/html` als HTML-Dokument,
- `text/plain` als normalen ASCII-Text und
- `image/gif` als GIF-Grafik.

Daneben gibt es noch viele weitere Typen. Auf dem Linux-Server enthält die Datei `/etc/apache2/mime.types` über 100 Einträge der Form:

```
text/css                        css
text/html                       html htm
text/plain                      asc txt c cc h hh cpp hpp
```

Der Web-Server übermittelt Dateien mit der Endung `.html` oder `.htm` als Typ `text/html`. Zeigt der Browser HTML-Dateien im Quellcode an, deutet dies auf ein Problem mit der Datei `/etc/apache2/mime.types`.

Für jede laufende Verbindung ist ein httpd-Prozess zuständig. Der WWW-Server startet bei Bedarf Kopien seiner selbst, die dann die zusätzlichen Verbindungen bedienen, und beendet diese dann wieder. Wie viele derartige Prozesse laufen dürfen, lässt sich über die Konfigurationsdatei einstellen.

Trotz vieler Prozesse verschwendet Apache dank Linux (oder des jeweils verwendeten Systems) keinen Speicherplatz für Prozesse, weil alle Kopien des WWW-Servers den Speicher gemeinsam nutzen.

6.3 Web-Server Apache installieren und einrichten

Bisher gibt es zwei Versionen des Apache, die bisherige Version und den vollkommen neu entwickelten Apache2. Glücklicherweise haben sich hauptsächlich die internen Funktionen des Apache verändert, weniger seine Nutzung und Konfiguration. Diese fünfte Auflage unseres Buchs beschreibt die neue Version den Apache2: Wenn hier im Buch von Apache geschrieben wird, dann ist immer Apache2 gemeint.

SuSE legt den Apache in die Selektion `Einfacher Webserver mit Apache2` ins Paket `apache2` bzw. auf dem FTP-Server oder der *CD3* in die entsprechende rpm-Datei. Da SuSE den Web-Server in der Standardinstallation nicht installiert, sollten Sie dies gemäß der Anleitung im Abschnitt 2.5 dieses Buchs nachholen.

Nach der Installation starten Sie den Web-Server mit dem Befehl

```
rcapache2 start
```

Überzeugen Sie sich, dass der Web-Server lauffähig ist, indem Sie von einem Client aus seine URL, hier im Beispiel `http://192.168.1.2` aufrufen. Der Browser müsste folgende recht leere Startseite anzeigen:

Abbildung 6.1: Standardstartseite im Browser

Um die Beispielseiten der bisherigen Versionen zu nutzen, müssen Sie auch das Paket *apache2-example-pages* installieren, das Sie in der Selektion *Einfacher Webserver mit Apache2* oder direkt auf der *CD4* finden.

Folgende Dateien sind für die Konfiguration des Web-Servers Apache wichtig:

Datei	*Bedeutung*
`/usr/sbin/httpd2`	Das Binärprogramm des Apache.
`/etc/apache2/`	Verzeichnis für die Konfigurationsdateien.
`/etc/apache2/httpd.conf`	Hauptkonfigurationsdatei.
`/etc/apache2/mime.types`	Datei mit den bekannten Dateitypen.
`/srv/www/`	Wurzelverzeichnis des Web-Servers.
`/srv/www/htdocs/`	Verzeichnis für normale Web-Dokumente.
`/srv/www/cgi-bin/`	Verzeichnis für ausführbare Programme (*CGI*).
`.htaccess`	Konfigurationsdatei im jeweiligen Web-Verzeichnis.

Tabelle 6.1: Dateien und ihre Bedeutung für die Konfiguration des Apache

Zum Einrichten des Apache dient die Hauptkonfigurationsdatei `httpd.conf`, die weitere Konfigurationsdateien einbindet.

`/etc/apache2/httpd.conf` (Dateianfang)

```
#
# /etc/apache2/httpd.conf
#
# This is the main Apache server configuration file. It contains
# the configuration directives that give the server its
```

```
# instructions. See <URL:http://httpd.apache.org/docs-2.0/> for
# detailed information aboutthe directives.

# Based upon the default apache configuration file that ships
# with apache, which is based upon the NCSA server configuration
# files originally by Rob McCool. This file was knocked
# together by Peter Poeml <poeml+apache@suse.de>.

# If possible, avoid changes to this file. It does mainly
# contain Include statements and global settings that can/should
# be overridden in the configuration of your virtual hosts.


# Overview of include files, chronologically:
#
# httpd.conf
#  |
#  |-- uid.conf  . . . . . . . . . . . . . UserID/GroupID
#                                          to run under
#  |-- server-tuning.conf  . . . . . . . . sizing of the server
#                                          (how many processes
#                                          to start, ...)

#  |-- sysconfig.d/loadmodule.conf . . . [*] load these modules
#  |-- listen.conf . . . . . . . . . . . . IP adresses /
#                                          ports to listen on
#  |-- mod_log_config.conf . . . . . . . . define logging formats
#  |-- sysconfig.d/global.conf . . . . . [*] server-wide
#                                          general settings
#  |-- mod_status.conf . . . . . . . . . . restrict access to
#                                          mod_status
#                                          (server monitor
ing)
#  |-- mod_info.conf . . . . . . . . . . . restrict access to
#                                          mod_info
#  |-- mod_usertrack.conf  . . . . . . . . defaults for cookie-
#                                          based user tracking
#  |-- mod_autoindex-defaults.conf . . . . defaults for
#                                          displaying of server-
#                                          generated directory
#                                          listings
#  |-- mod_mime-defaults.conf  . . . . . . defaults for mod_mime
#                                          configuration
#  |-- errors.conf . . . . . . . . . . . . customize error
#                                          responses
#  |-- ssl-global.conf . . . . . . . . . . SSL conf that applies
#                                          to default server _and
#                                          all_ virtual hosts
```

```
#  |
#  |-- default-server.conf . . . . . .  set up the default
#                                        server that replies
#                                        to non-virtual-host
#                                        requests

...
```

SuSE hat gegenüber den alten Versionen das Wurzelverzeichnis des Web-Servers verschoben. Statt unterhalb von `/usr/local/httpd/` finden Sie die Web-Dokumente jetzt unterhalb von `/srv/www/`.

Tipp: Bei großen Systemen sollten Sie für die Web-Dokumente eine eigene Partition benutzen.

Die etwa 200 Zeilen lange Konfigurationsdatei `/etc/apache2/httpd.conf` hat SuSE recht gut kommentiert, wobei die Übersichtlichkeit durch die Aufteilung in viele Einzeldateien etwas gelitten hat.

Für die Konfiguration des Apache2 finden Sie in *YaST* unter *Netzwerkdienste* den Punkt *HTTP-Server*.

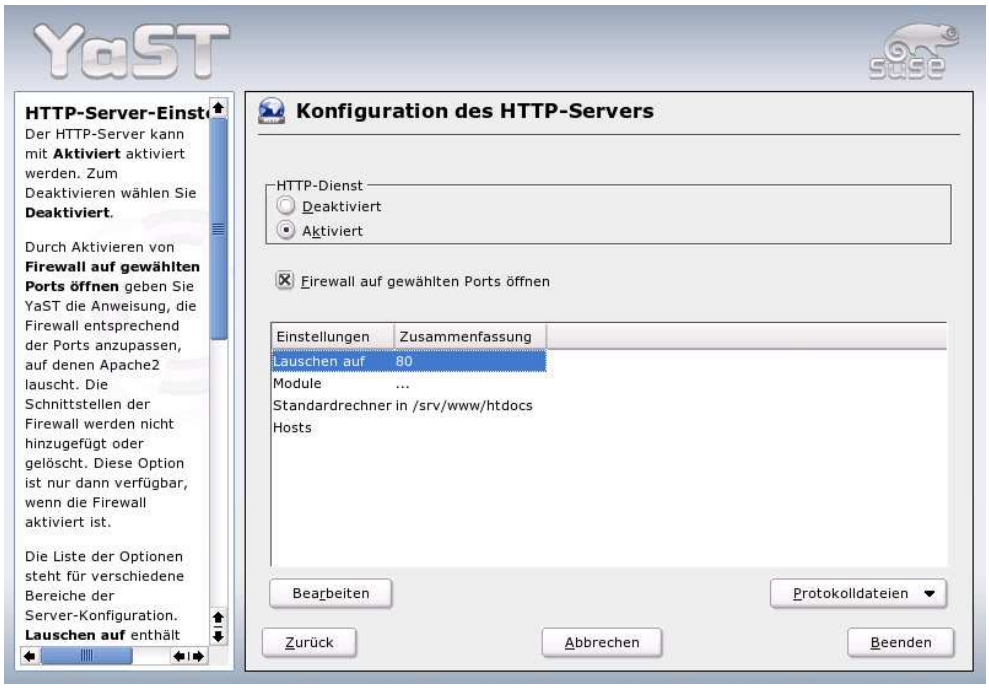

Abbildung 6.2: YaST: HTTP-Server-Konfiguration

Der Apache ist aber bereits ohne Änderungen an der Konfigurationsdatei voll funktionsfähig! Der folgende Text erläutert wichtige Abschnitte der Konfigurationsdatei, die für eine normale Nutzung bzw. das grundlegende Verständnis wichtig sind.

Tipp: Bearbeiten Sie die Konfigurationsdatei möglichst nie direkt, sondern nur mit YaST. Individuelle Veränderungen tragen Sie in zusätzliche Konfigurationsdateien ein und binden sie mit YaST per Include-Anweisung ein.

Ein wichtiger Teil der Konfiguration beschäftigt sich mit den ladbaren Modulen. Dies sind Programmteile, die der Apache bei Bedarf nachladen kann. Diese Module, die auch von Programmierern außerhalb des Apache-Teams stammen können, müssen sich an die Spezifikationen der *Apache HTTP Server Group* halten. Diese Offenheit und Erweiterbarkeit hat den enormen Erfolg des Apache Web-Servers mitbegründet.

Einen Eindruck von der Vielzahl der Erweiterungsmöglichkeiten bietet der folgende Ausschnitt aus einer der eingebundenen Konfigurationsdateien.

`/etc/apache2/sysconfig.d/loadmodule.conf`

```
#
# Files in this directory are created at apache start time by
# /usr/sbin/rcapache2. Do not edit them!
#

# as listed in APACHE_MODULES (/etc/sysconfig/apache2)

LoadModule access_module      /usr/lib/apache2-prefork/
                              ↵ mod_access.so
LoadModule actions_module     /usr/lib/apache2-prefork/
                              ↵ mod_actions.so
LoadModule alias_module       /usr/lib/apache2-prefork/
                              ↵ mod_alias.so
LoadModule auth_module        /usr/lib/apache2-prefork/
                              ↵ mod_auth.so
LoadModule auth_dbm_module    /usr/lib/apache2-prefork/
                              ↵ mod_auth_dbm.so
LoadModule autoindex_module   /usr/lib/apache2-prefork/
                              ↵ mod_autoindex.so
LoadModule cgi_module         /usr/lib/apache2-prefork/
                              ↵ mod_cgi.so
LoadModule dir_module         /usr/lib/apache2-prefork/
                              ↵ mod_dir.so
LoadModule env_module         /usr/lib/apache2-prefork/
                              ↵ mod_env.so
LoadModule expires_module     /usr/lib/apache2-prefork/
                              ↵ mod_expires.so
```

```
LoadModule include_module     /usr/lib/apache2-prefork/
                             ↵ mod_include.so
LoadModule log_config_module  /usr/lib/apache2-prefork/
                             ↵ mod_log_config.so
LoadModule mime_module        /usr/lib/apache2-prefork/
                             ↵ mod_mime.so
LoadModule negotiation_module /usr/lib/apache2-prefork/
                             ↵ mod_negotiation.so
LoadModule setenvif_module    /usr/lib/apache2-prefork/
                             ↵ mod_setenvif.so
LoadModule ssl_module         /usr/lib/apache2-prefork/
                             ↵ mod_ssl.so
LoadModule suexec_module      /usr/lib/apache2-prefork/
                             ↵ mod_suexec.so
LoadModule userdir_module     /usr/lib/apache2-prefork/
                             ↵ mod_userdir.so
#
```

Diese Konfigurationsdatei beschäftigt sich mit dem Laden der Module, hierbei muss man dem Apache den Dateinamen des Moduls angeben.

YaST bzw. SuSEconfig verwalten diese Konfigurationsdatei abhängig von den installierten und aktivierten Modulen.

Um Module zu aktivieren oder zu deaktivieren, klicken Sie in der HTTP-Konfiguration von YaST auf *Module* • *Bearbeiten*.

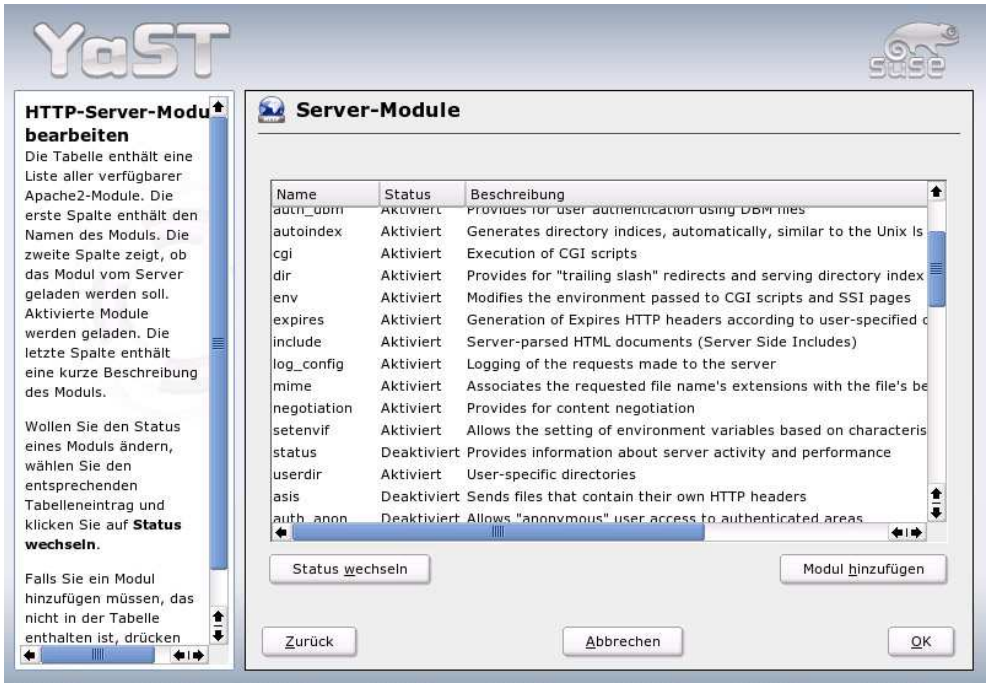

Abbildung 6.3: YaST: HTTP-Server-Module

Jede Änderung in diesem Menü modifiziert die `loadmodule.conf`.

In der Datei `/etc/apache2/uid.conf` legen Sie den Benutzernamen und die Gruppe für den Apache fest.

```
/etc/apache2/uid.conf
User wwwrun
Group www
```

Zum Schutz des Linux-Servers, auf dem der Web-Server läuft, verwendet der Web-Server den Benutzernamen `wwwrun` und die Gruppe `nogroup`, die beide mit wenigen Rechten verbunden sind. Dies verhindet z. B., dass der Web-Server auf fremde Dateien zugreifen kann.

Diese Einstellung sollten Sie nicht ändern. Andere Konfigurationen können Sie wieder bequem mit YaST verändern.

Für wichtige Grundeinstellungen klicken Sie in der HTTP-Konfiguration auf den Eintrag *Standardrechner* in `/srv/www/htdocs` und dann auf *Bearbeiten*.

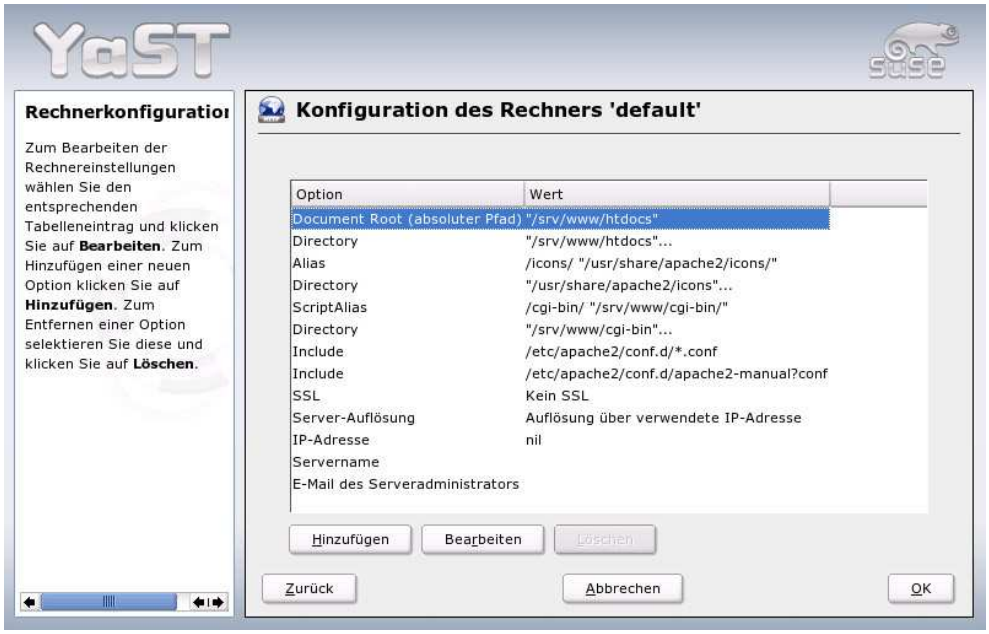

Abbildung 6.4: YaST: HTTP-Standardrechner

Hier können Sie viele Parameter einstellen. Einen Teil davon werden Sie beim Durcharbeiten dieses Buches kennen lernen.

Im ersten Schritt sollten Sie hier einen Rechnernamen und die Mail-Adresse des Web-Masters angeben.

YaST überträgt die Einstellungen dann in die Datei `/etc/apache2/default-server.conf`, die recht umfangreich ist.

`/etc/apache2/default-server.conf` (Dateiende)

```
…
# Include all *.conf files from /etc/apache2/conf.d/.
#
# This is mostly meant as a place for other RPM packages to drop
# in their configuration snippet.
#
# You can comment this out here if you want those bits include
# only in acertain virtual host, but not here.
#
Include /etc/apache2/conf.d/*.conf

# The manual... if it is installed ('?' means it won't complain)
Include /etc/apache2/conf.d/apache2-manual?conf
ServerName boss.lokales-netz.de
ServerAdmin webmaster@lokales-netz.de
# YaST auto define section
<IfDefine SSL>
 SSLEngine off
</IfDefine>
```

Die hier vorgeschlagene Einstellung für den *ServerAdmin* ist sehr allgemein, die Mail an diese Adresse wird aber sicher zugestellt. Passen Sie den Namen des Netzwerks an, und tragen Sie ggf. Ihre persönliche Mail-Adresse ein. Da der Apache diese Adresse bei Fehlermeldungen ausgibt, sollte sie einen Bezug zum lokalen System besitzen. Üblich ist eine Angabe wie `webmaster@boss.lokales-netz.de`.

Sie können diese Angabe auch ändern, wenn Sie im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache2* unter *APACHE_SERVERADMIN* den gewünschten Wert angeben.

Im Abschnitt *Virtuelle Server* (6.6) lesen Sie hier, dass der Apache mit mehreren Adressen gleichzeitig arbeiten kann. Daher können Sie ihm angeben, mit welchem Namen er sich gegenüber dem Klienten melden soll. Auch hier haben Sie in YaST bereits einen Vorschlag für *ServerName* eingetragen.

Gibt man keinen Namen an, benutzt Apache den lokalen Rechnernamen, wenn der Server Fehlermeldungen an den Browser übermittelt, hier im Beispiel also `boss.lokales-netz.de`. Wollte man lieber `www.lokales-netz.de` übermitteln, so könnte man das hier direkt ändern. Man darf aber nur Namen benutzen, die der Server auch korrekt auflösen kann. Hinweise zur Namensauflösung finden Sie im

Kapitel 13 über den Domain Name-Server. Solange auf Ihrem Linux-Server noch kein Name-Server läuft, sollten Sie hier zunächst die Vorgabe belassen.

Auch diese Angabe können Sie jederzeit über den *Editor für /etc/sysconfig-Daten* ändern, wenn Sie unter *Network* • *WWW* • *Apache2* für *APACHE_SERVERNAME* den gewünschten Wert eintragen.

Sie müssen dem Apache auch mitteilen, wo er seine Web-Seiten findet. Dies und weitere grundlegende Parameter stellen Sie in der bereits erwähnten Datei `default-server.conf` ein.

`/etc/apache2/default-server.conf` (Dateianfang)

```
#
# Global configuration that will be applicable for all virtual
# hosts, unless deleted here, or overriden elswhere.
#

DocumentRoot "/srv/www/htdocs"
```

Gegenüber älteren Versionen von SuSE Linux ist dies eine Veränderung; dort lagen die Web-Seiten unterhalb von `/usr/local/httpd/htdocs`.

Normalerweise muss man diese Einstellung nicht ändern. Im angegebenen Verzeichnis befinden sich die Seiten, die der Web-Server anbieten kann.

Für jedes über das Web zugängliche Verzeichnis kann man Parameter einstellen. Diese vererbt Apache an Unterverzeichnisse, sofern es für diese Unterverzeichnisse nicht neue Angaben gibt.

Die strengsten Vorgaben stehen dabei in der `httpd.conf`, sie lassen erst einmal keinerlei Zugriff zu.

```
/etc/apache2/httpd.conf
# forbid access to the entire filesystem by default
<Directory />
    Options None
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

Da dies das höchste Verzeichnis ist, beschränkt man hier massiv Rechte. Die Einschränkungen kann man in den einzelnen Unterverzeichnissen wieder aufheben. Die Konfiguration gibt keinerlei Optionen frei. Die Zeile `AllowOverride None` bewirkt, dass Benutzer die Einstellungen nicht durch Angaben in einer Datei `.htaccess` im jeweiligen Verzeichnis ändern dürfen. In einer derartigen Datei könnte man alle Optionen für Verzeichnisse überschreiben, wenn `Allow Override All` dies erlauben würde.

Einen Teil dieser Einschränkungen überschreiben Sie für das `htdocs`-Verzeichnis dann wieder in der `default-server.conf`.

```
#
# Configure the DocumentRoot
#
<Directory "/srv/www/htdocs">
  # Possible values for the Options directive are
  # "None", "All", or any combination of:
  #   Indexes Includes FollowSymLinks
  #   SymLinksifOwnerMatch ExecCGI MultiViews
  #
  # Note that "MultiViews" must be named
  # *explicitly* --- "Options All"
  # doesn't give it to you.
  #
  # The Options directive is both complicated and important.
  # Please see
  # http://httpd.apache.org/docs-2.0/mod/core.html#options
  # for more information.
Options None
  # AllowOverride controls what directives may be placed
  # in .htaccess files.
  # It can be "All", "None", or any combination of
  # the keywords:
  #   Options FileInfo AuthConfig Limit
AllowOverride None
  # Controls who can get stuff from this server.
Order allow,deny
 Allow from all
</Directory>
```

Entscheidend ist hier die Einstellung `Allow from all`, sonst wäre kein Zugriff auf den Web-Server möglich.

Welche Rechner auf das Verzeichnis zugreifen dürfen, legt man durch die Reihenfolge von Regeln und Einzel-Regeln fest:

```
Order allow,deny
Allow from all
```

Zuerst bestimmt eine Regel die Reihenfolge des Erlaubens und Ablehnens. Hier im ersten Beispiel haben Regeln der Art `allow` Vorrang vor Regeln der Art `deny`. Als einzige Regel folgt dann eine `allow`-Regel, die den Zugriff für alle Rechner freigibt. Wollte man nur den Rechnern der eigenen Domäne einen Zugriff erlauben, so wäre das wie hier im zweiten Beispiel möglich mit

```
Order deny,allow
Deny from all
Allow from lokales-netz.de
```

Die restlichen Einstellungen sind weiterhin sehr restriktiv. Die in vorherigen Versionen übliche Option `Options Indexes -FollowSymLinks +Includes +MultiViews` bewirkt, dass Apache für Ordner ohne Standard-Datei (z. B. `index.htm` s. u.) ein Inhaltsverzeichnis erzeugt. Symbolische Links sind immer noch verboten, erlaubt sind aber die Server Side Includes (*SSI*), spezielle Programmbefehle, die man in HTML-Seiten integrieren kann.

Sie können URLs verkürzen, wenn Sie Standards für die Namen der Startseite vorgeben. Üblich sind hier u. a. die Angaben `index.html` und `welcome.html`. Um hier etwas flexibler zu werden, können Sie eine Zeile in der Konfiguration noch erweitern. In der `httpd.conf` steht:

```
# List of resources to look for when the client requests a
# directory
DirectoryIndex index.html index.html.var
```

Dies bewirkt, dass man, wenn die Option *Indexes* gesetzt ist, bei Startseiten den Dateinamen weglassen darf. Die Eingabe der URL `http://192.168.1.2/` ist dann gleichbedeutend mit `http://192.168.1.2/index.html`. Um auch Startdateien wie `welcome.htm` zu berücksichtigen, müssen Sie diese Zeile erweitern. Legen Sie eine Datei `/etc/apache2/linuxbuch.conf` mit folgendem Inhalt an:

```
DirectoryIndex index.html index.htm welcome.html welcome.htm
```

Die Reihenfolge dieser Aufzählung entscheidet über den Vorrang. Wenn sowohl eine Datei `index.html` als auch eine Datei `welcome.htm` existieren, dann überträgt Apache die Datei `index.html`.

Abbildung 6.5: Eigene Konfigurationsdatei einbinden

Zum Aktivieren dieser Änderung müssen Sie anschließend im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache2* für die Variable *APACHE_CONF_INCLUDE_FILES* den Wert `/etc/apache2/linuxbuch.conf` angeben.

Nach einem Neustart des Web-Servers mit

```
rcapache2  restart
```

sind diese Änderungen wirksam.

In der Standardinstallation des Apache funktioniert der Seitenaufruf `http://192.168.1.2/icons/`. Unterhalb von `/srv/www/htdocs` gibt es aber kein Verzeichnis `icons`.

Einstellungen für so genannte virtuelle Namen in der Datei `/etc/apache2/default-server.conf` sorgen dafür, dass der Link trotzdem funktioniert:

```
# Aliases: aliases can be added as needed (with no limit).
# The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the
# server will # require it to be present in the URL.
# So "/icons" isn't aliased in this
# example, only "/icons/".
# If the fakename is slash-terminated, then the
# realname must also be slash terminated, and if the
# fakename omits the
# trailing slash, the realname must also omit it.
#
# We include the /icons/ alias for FancyIndexed directory
# listings.  If you
# do not use FancyIndexing, you may comment this out.
#
Alias /icons/ "/usr/share/apache2/icons/"

<Directory "/usr/share/apache2/icons">
 Options Indexes MultiViews
 AllowOverride None
 Order allow,deny
 Allow from all
</Directory>
```

Der Apache ordnet virtuellen Namen, hier `/icons/`, reale Dateien bzw. Verzeichnisse zu, hier `/usr/share/apache2/icons/`. Der virtuelle Name heißt `Alias`. Der Aufruf von `http://192.168.1.2/icons/` greift also nicht auf `/srv/www/htdocs/icons/` zu, sondern auf `/usr/share/apache2/icons/`. Wie Sie diese praktische Einrichtung selbst nutzen können, lesen Sie im Abschnitt 6.4.

Ausführbare Programme (z. B. cgi-Skripten) sammelt man üblicherweise in dem speziellen Verzeichnis `/cgi-bin/`. Zur Verbesserung der Systemsicherheit legt man dieses Verzeichnis nicht unterhalb von `htdocs` an. Benutzern, die nur Web-Seiten erstellen dürfen, kann man beispielsweise per FTP oder Samba einen Zugriff auf das `htdocs`-Verzeichnis erlauben, ohne dass sie Programme im `cgi-bin`-Verzeichnis ablegen können.

Für Verzeichnisse mit ausführbaren Programmen gibt es einen speziellen Skript-Alias-Befehl, der ebenfalls in der Datei `/etc/apache2/default-server.conf` zu finden ist.

```
# ScriptAlias: This controls which directories contain server
# scripts.
# ScriptAliases are essentially the same as Aliases, except
# that documents in the realname directory are treated as
# applications and run by the server when requested rather
# than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias
# directives as to Alias.
#
ScriptAlias /cgi-bin/ "/srv/www/cgi-bin/"

# "/srv/www/cgi-bin" should be changed to whatever your
# ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/srv/www/cgi-bin">
 AllowOverride None
 Options +ExecCGI -Includes
 Order allow,deny
 Allow from all
</Directory>
```

Jedes ausführbare Programm in diesem Verzeichnis ist ein Sicherheitsrisiko. Sie sollten die Zugriffsberechtigung für das `cgi-bin`-Verzeichnis daher nur sehr zurückhaltend vergeben.

6.4 Web-Dokumente ordnen und aufspielen

Die Vorgehensweise für das Ordnen und Aufspielen von Web-Dokumenten hängt sehr von den individuellen Arbeits- und Organisationsformen ab. Beim Verwalten von Web-Sites kann man in der Praxis drei Systeme beobachten:

- Zentralisiert,
- hierarchisch und
- chaotisch.

Bei einer zentralisierten Web-Verwaltung hat im Extremfall nur ein einziger Mitarbeiter, der Web-Administrator, Schreibzugriff auf die Seiten. Alle anderen Mitarbeiter müssen ihm Seiten zukommen lassen, er überprüft sie und bindet sie in das Gesamtangebot ein. Hier genügt es, wenn der Web-Administrator das Verzeichnis `/srv/www/htdocs` per FTP (s. u.) bzw. Samba (s. u.) erreichen kann. Beim FTP-Zugriff gestattet man diesem Web-Administrator entweder einen Zugriff auf das gesamte System, oder man verlegt sein Home-Verzeichnis nach `/srv/www/htdocs`, wobei er dann dort natürlich keine privaten Dateien ablegen sollte.

Bei einem hierarchischen System verwaltet ein Web-Administrator die Startseite, alle weiteren Rubriken betreuen jeweils andere Mitarbeiter, die Inhalte bestimmter Verzeichnisses selbst verantworten, z. B. die Benutzerin *Meyer* das Verzeichnis `speiseplan`. Der Web-Administrator muss dann nur die Verweise auf die Startseiten dieser Verzeichnisse anlegen.

Für die Zugriffe auf diese individuellen Verzeichnisse benutzt man das Alias-System des Apache. Hierzu legen Benutzer ein Verzeichnis `html` in ihr Home-Verzeichnis. Der Administrator setzt ein Alias auf dieses Verzeichnis, hier im Beispiel in der Datei `/etc/apache2/linuxbuch.conf`:

```
Alias  /speiseplan/  /home/meyer/html/
```

Der Zugriff auf die URL `http://192.168.1.2/speiseplan/` landet dann im Home-Verzeichnis der Benutzerin *Meyer*. Auf dieses Verzeichnis hat sie bei den hier im Buch beschriebenen Installationen von FTP und Samba vollen Zugriff.

Am aufwändigsten ist die chaotische Verwaltung zu regeln, bei der alle Benutzer vollen Zugriff auf alle Dokumente des Web-Servers haben. Dazu muss das gesamte `htdocs` Verzeichnis per FTP oder Samba erreichbar sein.

Für Samba ist eine spezielle Freigabe `www` auf dieses Verzeichnis die einfachste Lösung. Beim FTP-Zugriff verzichtet man entweder auf die sicherere *Changed-Root-Umgebung* (siehe FTP, Kapitel 7), oder man legt das `htdocs`-Verzeichnis einfach unterhalb von `/home` an, indem man den Eintrag `DocumentRoot` in der Apache-Konfigurationsdatei verschiebt:

```
DocumentRoot "/home/wwwhome/htdocs"
```

Dies ist ein auf vielen Web-Servern übliches Verfahren. Man muss bei der Veränderung etwas aufpassen, da man alle Pfade in der `Apache-Konfiguration` anpassen muss, die bisher mit `/srv/www/htdocs` anfingen.

6.5 Zugriffssteuerung für geschlossene Nutzergruppen

Auf vielen Web-Servern (nicht nur auf unanständigen) gibt es Bereiche, die man nur betreten kann, wenn man über einen dafür gültigen Benutzernamen und ein Passwort verfügt.

Wenn man z. B. unterhalb der URL `http://192.168.1.2/protokolle/` vertrauliche Protokolle ablegen will, muss man dem Apache mitteilen, dass er die Berechtigung für Zugriffe auf dieses Verzeichnis überprüfen soll.

Dazu muss man in der Datei `/etc/apache2/linuxbuch.conf` eine weitere Directory-Direktive einfügen:

```
<Directory /srv/www/htdocs/protokolle>
 authName Geheim-Protokolle
 authType Basic
 authuserFile /etc/apache2/protokolle.pwd
 require valid-user
</Directory>
```

Die erste Zeile legt den Text fest, den Apache den Benutzern im Eingabefenster für das Passwort anzeigt. Die zweite Zeile bestimmt die Art der Autorisierung. Üblich ist hier der Typ `Basic`, da nicht alle Browser den Typ `Digest` unterstützen, der die Benutzerdaten verschlüsselt zwischen Client und Server überträgt. Die dritte Zeile legt fest, wo die Datei mit den Benutzernamen und Passwörtern liegt und die letzten Zeile regelt, dass alle Benutzer, die sich anmelden können, einen Zugriff bekommen. Die möglichen Einstellungen hier sind `user`, `group` und `valid-user`. Würde man hier im Beispiel angeben:

```
require  user  meyer
```

so bekämen andere Benutzer keinen Zugriff, auch wenn sich ihr Benutzername und Passwort in der angegebenen Passwortdatei wiederfindet. Neben dem `authuserFile` könnte man auch noch ein `authgroupFile` angeben, um gruppenbezogene Zugriffe zu erlauben.

Tipp: Die Benutzer, Gruppen und Passwörter haben nichts mit denen des Linux-Systems zu tun. Die Apache-Benutzernamen sollten von Linux-Benutzernamen abweichen, da Benutzernamen unverschlüsselt über das Netz gehen, wenn man nicht mit gesicherten http-Verbindungen arbeitet.

Bevor Sie die neue Konfiguration testen können, müssen Sie noch die in der Konfiguration angegebene Passwortdatei erzeugen und mindestens einen Benutzer einrichten.

Das Programm `/usr/bin/htpasswd2` erzeugt und verändert die Passwortdatei. Eine neue Passwortdatei mit einer Benutzerin `meyer` erzeugt man mit

```
/usr/bin/htpasswd2 -c /etc/apache2/protokolle.pwd meyer
```

Hier muss man dann zweimal ihr Passwort angeben. Der Schalter `-c` (für *create*) erzeugt die Datei beim ersten Aufruf und muss bei weiteren Eingaben entfallen, da Sie sonst die vorhandene Datei überschreiben würden.

Nach einem Neustart des Apache mit

```
rcapache2 restart
```

können Sie einen ersten Zugriff auf den Ordner ausprobieren, indem Sie die URL `http://192.168.1.2/protokolle/` in einen Browser eingeben. In einem Fenster sehen Sie dann einen Dialog zur Eingabe von Benutzernamen und Passwort.

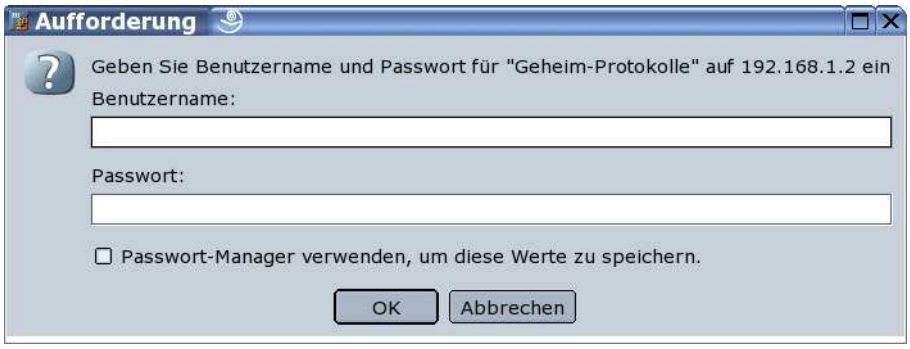

Abbildung 6.6: Authentifizierung

Das genaue Aussehen dieses Fensters hängt vom Client-Betriebssystem, dem Browser und dessen Konfiguration ab.

Nach erfolgreichem Aufruf müssten Sie nun das bisher leere Inhaltsverzeichnis des Ordners sehen. Bei einer Fehlermeldung finden Sie die Fehlerursache auf dem Linux-Server in der Datei `/var/log/apache2/error_log`.

Einträge in der Passwortdatei löscht man mit einem Texteditor, nicht mit `htpasswd2`, da dieses keine derartige Funktion kennt.

Die Zeile für die Benutzerin `meyer`, die Sie soeben eingerichtet haben, sieht in der Datei folgendermaßen aus:

```
/etc/apache2/protokolle.pwd
meyer:TY3OT2DpYoAVA
```

In der ersten Spalte steht vor dem Doppelpunkt der Benutzername, danach folgt das verschlüsselte Passwort. Löschen Sie diese Zeile, so nehmen Sie der Benutzerin die Zugriffsrechte auf den Ordner wieder weg.

Zum Anlegen der Gruppendateien benötigt man ebenfalls einen Texteditor.

```
/etc/apache2/protokolle.grp
autoren: adams, tikart, meyer
koerner: roggen, gerste, hirse
```

Links vom Doppelpunkt steht der Name der Gruppe, rechts davon die Mitgliederliste.

Mit der Gruppenzugehörigkeit und der Möglichkeit, unabhängige Passwort- und Gruppendateien für jedes Verzeichnis anzulegen, kann man die Zugriffsrechte sehr genau regeln.

Wenn aber sehr viele Benutzer auf diese Art den Zugriff auf die gesicherten Web-Seiten bekommen sollen, wie es z. B. bei den *Linuxbu.ch/Tools* (siehe Kapitel 3) der Fall ist, dann ist der Aufwand hoch.

In der aktuellen Version gibt es dazu ein Paket mit einem Modul zur Authentifikation gegenüber einer MySQL-Datenbank, das Modul `mod_auth_mysql`, das Sie bei Bedarf im Paket `apache2-mod_auth_mysql` finden.

Nach allen Änderungen an der Konfigurationsdatei müssen Sie den Apache unbedingt mittels

```
rcapache2 restart
```

neu starten.

Wenn Sie in der Konfigurationsdatei statt der Angabe

```
require valid-user
```

den Text

```
require group autoren
```

einsetzen, dann bekommen auf diese Art nur die Mitglieder der vorher definierten Gruppe *autoren* Zugriff auf die geschützten Seiten.

6.6 Virtuelle Server

Internet-Provider bieten Homepages für viele Kunden auf demselben Web-Server an. All diese Web-Sites bedient derselbe Web-Server, der nicht nur auf seine IP-Adresse sondern auch auf viele verschiedene Web-Adressen reagieren muss. Für jede Web-Adresse benutzt der virtuelle Server ein anderes Home-Verzeichnis.

Der Apache bietet dieses Feature unter der Bezeichnung `VirtualHosts`, *virtuelle Server*, an.

Bevor Sie virtuelle Server konfigurieren, müssen Sie einen Name-Server installiert haben (siehe Kapitel 13).

Mehrere virtuelle Web-Server auf demselben System können auch im lokalen Netz sinnvoll sein, um inhaltliche Bereiche klar voneinander zu trennen.

Betreiben Sie neben dem normalen Web-Server `http://www.lokales-netz.de` einen Server `http://www2.lokales-netz.de`, so können Sie diesen so konfigurieren, dass er das Unterverzeichnis `Protokolle` aus dem vorangegangenen Beispiel als Home-Verzeichnis anzeigt. Dazu müssen Sie die Konfiguration erweitern, entweder mit YaST oder besser direkt mit einem Texteditor.

Wenn Sie mit YaST arbeiten wollen, dann starten Sie unter *Netzwerkdienste* den Punkt *HTTP-Server*, gehen dann auf den Eintrag *Hosts* und klicken dann auf *Bearbeiten*.

Sie gelangen in das Menü *Konfigurierte Rechner*, in dem Sie nur den Standard-Rechner als Eintrag vorfinden. Wenn Sie hier nun auf *Hinzufügen* klicken, können Sie in einer neu geöffneten Maske die gewünschten Daten erfassen.

Abbildung 6.7: YaST: Neue Rechner-Informationen

Wenn Sie jetzt zweimal auf *OK* und dann auf *Beenden* klicken, übernimmt YaST Ihre Eingaben in die Konfiguration. Sie finden die Angaben in der Datei `/etc/apache2/vhosts.d/yast2_vhosts.conf`.

```
<VirtualHost 192.168.1.2>
# YaST auto define section
 <IfDefine SSL>
```

```
  SSLEngine off
 </IfDefine>
 DocumentRoot /srv/www/htdocs/protokolle/
 ServerName www2.lokales-netz.de
 ServerAdmin webmaster@lokales-netz.de
</VirtualHost>
```

Bei sehr speziellen Konfigurationswünschen für Ihre virtuellen Hosts wird die Nutzung von YaST recht umständlich. In diesem Fall können Sie mit dem Texteditor Ihrer Wahl selbst eine Konfigurationsdatei erstellen und im Verzeichnis `/etc/apache2/vhosts.d/` ablegen. Apache bindet beim Start alle Dateien in diesem Verzeichnis, deren Datei-Namen auf `.conf` enden, automatisch mit ein. Sie finden in diesem Verzeichnis auch eine sehr umfangreiche Beispieldatei `vhost.template`. Entweder erweitern Sie diese Datei entsprechend, oder Sie halten sich an das folgende Beispiel der Autoren. Benennen Sie die Datei so, dass ihr Name auf `.conf` endet.

`/etc/apache2/vhosts.d/vhost.conf` (Beispiel)

```
NameVirtualHost *:80

<VirtualHost _default_:80>
</VirtualHost>

<VirtualHost *>
 ServerName  www.lokales-netz.de
</VirtualHost>

<VirtualHost *>
  Servername www2.lokales-netz.de
  DocumentRoot /srv/www/htdocs/protokolle
</VirtualHost>
```

Die Einstellungen sind deutliche übersichtlicher als in der Beispieldatei von SuSE.

Beim Arbeiten mit virtuellen Hosts möchte der Apache die zugehörige IP wissen, da es auch möglich wäre, dass die Hosts auf verschiedene Adressen reagieren.

```
NameVirtualHost 192.168.1.2
```

Benutzer mit dynamischen IP-Adressen konnten bei den früheren Apache-Versionen keine virtuellen Server einrichten, da sie keine feste IP für die Konfigurationsdatei angeben konnten. Bei der aktuellen Apache-Version können Sie statt der IP immer auch das Jokerzeichen * angeben, das dann für alle IP-Adressen steht. Zusätzlich müssen Sie dem Apache auch den Port mit angeben.

```
NameVirtualHost  *:80
```

Damit können Sie auch bei dynamischen IP-Adressen virtuelle Server einrichten.

Den neuen virtuellen Server mit dem Wurzelverzeichnis `/srv/www/htdocs/protokolle` definieren Sie mit der VirtualHost-Direktive des Apache.

```
<VirtualHost  *>
  ServerName  www.lokales-netz.de
</VirtualHost>

<VirtualHost  *>
  ServerName  www2.lokales-netz.de
  DocumentRoot  /srv/www/htdocs/protokolle
</VirtualHost>
```

Den bisherigen Standard-Server sollte man hier noch einmal definieren. Auch dieser ist jetzt nur noch ein virtueller Host. Zusätzlich muss man auch Anfragen regeln, die nicht über `www` oder `www2` auf den Server zukommen, sondern z. B. direkt über die IP-Adresse; auch hierfür muss ein virtueller Host definiert sein. Alle denkbaren Möglichkeiten deckt eine `default`-Definition für den WWW-Port 80 ab:

```
<VirtualHost _default_:80>
</VirtualHost>
```

Alle Konfigurationseinstellungen, die in der VirtualHost-Directive fehlen, übernimmt der Apache aus der Grundkonfiguration, die Sie ja schon vorher erstellt haben.

Über virtuelle Hosts kann man das eigene Web-Angebot benutzerspezifisch strukturieren oder die Angebote mehrerer Firmen bzw. Abteilungen auf einem einzigen Server hosten. Je nachdem, welchen Web-Server Besucher ansprechen, bietet der Apache dann verschiedene Zugänge an.

Damit der Apache die Veränderungen der Konfigurationsdatei übernimmt, müssen Sie ihn neu starten.

```
rcapache2 restart
```

6.7 Gesicherte Zugriffe mit Secure Sockets Layer (SSL)

Beim bisher besprochenen Zugriffsschutz mit Benutzernamen und Passwort schickt der Browser die Daten unverschlüsselt über das Netz.

Vertrauliche Informationen sollte man besser verschlüsselt übertragen. Das von Netscape entwickelte System basiert auf dem SSL-Protokoll, das auch für andere Dienste, z. B. FTP verwendbar ist.

Das zum Nutzen dieses Protokolls benötigte Apache-Modul `mod_ssl` aktiviert die SuSE-Installation nicht per Voreinstellung.

Sie müssen SSL im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache2* aktivieren, indem Sie bei der Variablen *APACHE_SERVER_FLAGS* den Wert `-D SSL` setzen und dann den Apache neu starten.

Zwei Konfigurationsschritte bleiben noch:

- Man muss die Apache-Konfiguration so erweitern, dass Apache auf dem Port 443 gesicherte Verbindungen aufbaut, und
- ein Zertifikat erzeugen, mit dem sich der Linux-Server gegenüber dem Browser ausweist.

Da SuSE schon ziemlich viel vorbereitet hat, muss man die Einstellungen nur an die eigenen Bedingungen anpassen und dann aktivieren.

Da es sich hier wieder um einen virtuellen Server handelt, erfolgen die Einstellungen erneut im Verzeichnis `/etc/apache2/vhosts.d/`. Sie finden hier eine Musterdatei `vhost-ssl.template`, die Sie einfach kopieren können.

```
cp vhost-ssl.template  vhost-ssl.conf
```

Durch den neuen Namen sind die Einstellungen in der Datei bei einem Neustart des Apache sofort aktiv. Sie können die Datei jederzeit an Ihre eigenen Bedürfnisse anpassen.

```
/etc/apache2/vhosts.d/vhost-ssl.cof (Auszug)
##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

        #  General setup for the virtual host
        DocumentRoot "/srv/www/htdocs"
        #ServerName www.example.com:443
        #ServerAdmin webmaster@example.com
        ErrorLog /var/log/apache2/error_log
        TransferLog /var/log/apache2/access_log

        #   SSL Engine Switch:
        #   Enable/Disable SSL for this virtual host.
        SSLEngine on
```

Diesen Abschnitt wertet Apache nur dann aus, wenn er mit dem Parameter zum Einbinden des SSL-Modules startet, was Sie im Schritt davor mittels YaST geregelt haben.

Da Ihnen SuSE mit der Vorlagendatei schon einen großen Teil der Konfigurationsarbeit abnimmt, müssen Sie nur noch virtuelle Server für den Apache definieren.

Sie definieren für SSL-Verbindungen einen eigenen Server (*Virtual Host*). Die Einstellung 443 für den Standard-Port für https sollte man nicht verändern.

```
##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>
```

Sie sollten für diesen Server einen eigenen Verzeichnisbaum aufbauen, üblich ist `/srv/www/ssldocs`. Die Vorlage von SuSE legt den Server auch in den Verzeichnisbaum `htdocs`. Es ist jedoch riskant, wenn gesicherter und ungesicherter Server im selben Verzeichnis liegen. Ändern Sie die Vorgaben, damit das gesicherte Verzeichnis nicht über den normalen Server erreichbar ist.

Die restlichen Einstellungen überschreiben die Grundeinstellungen für diesen Server. Die Log-Dateien können mit denen für den normalen Server identisch sein; darin besteht kein Sicherheitsrisiko.

```
#  General setup for the virtual host
DocumentRoot "/srv/www/ssldocs"
#ServerName www.example.com:443
#ServerAdmin webmaster@example.com
ErrorLog /var/log/apache2/error_log
TransferLog /var/log/apache2/access_log
```

Die Einstellung für die SSLEngine ist wichtig. Nur wenn `SSLEngine` auf `on` steht, aktiviert der Apache wirklich SSL.

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on
```

Nun folgen bis zum Dateiende noch ein paar Einstellungen und Pfade für SSL, die man nicht ändern muss:

```
#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:
  ↵ +SSLv2:+EXP:+eNULL
...
```

SSL übertragt dann Login und Daten verschlüsselt. Der Browser stellt mit dem Schlüssel sicher, dass er mit dem echten Server verbunden ist und nicht etwa mit einem Rechner, der sich nur für den echten Server ausgibt. Dazu muss man auf dem

Server ein Schlüsselzertifikat erzeugen und es von einer anerkannten Zertifizierungsstelle (*Certification Authority*, CA) signieren lassen.

Browser erkennen einige bekannte Zertifizierungsstellen automatisch an.

Deutsche Zertifizierungsstellen für SSL sind immer noch im Aufbau, zu den bereits aktiven Organisationen gehört der DFN-Verein, dessen SSL-Informationen Sie unter `http://www.pca.dfn.de/dfnpca/certify/ssl/` finden.

Die Zertifizierungsstellen sind kommerzielle Einrichtungen und verlangen für ihre Dienste in der Regel eine Gebühr.

Für viele Anwendungszwecke reicht auch eine kostenlose Lösung für eine Testinstallation. Benutzen Sie für Tests als Zertifizierungsinstanz die fiktive Firma *Snake Oil;* die notwendigen Daten dieser Firma gehören zum SSL-Modul. Ein Nachteil besteht darin, dass Browser die Zertifikate dieser Firma nicht automatisch anerkennen.

Zum Erzeugen der Zertifikate wechseln Sie in das Verzeichnis `/usr/share/doc/packages/apache2` und starten das Programm

```
./certificate.sh
```

das dann die notwendigen Angaben erfragt. Eigene Eingaben sind hier fett hervorgehoben.

```
SSL Certificate Generation Utility (mkcert.sh)
Copyright (c) 1998 Ralf S. Engelschall, All Rights Reserved.

Generating test certificate signed by Snake Oil CA [TEST]
WARNING: Do not use this for real-life/production systems
_____________________________________________________________________

STEP 0: Decide the signature algorithm used for certificate
The generated X.509 CA certificate can contain either
RSA or DSA based ingredients. Select the one you want to use.
Signature Algorithm ((R)SA or (D)SA) [R]:R
_____________________________________________________________________
STEP 1: Generating RSA private key (1024 bit) [server.key]
488077 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
 ......................+++++
........+++++
e is 65537 (0x10001)
_____________________________________________________________________

STEP 2: Generating X.509 certificate signing request
[server.csr]
Using configuration from .mkcert.cfg
```

```
You are about to be asked to enter information that
will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Country Name             (2 letter code) [XY]:DE
2. State or Province Name   (full name)     [Snake
 ↵ Desert]:Germany
3. Locality Name            (eg, city)      [Snake
 ↵ Town]:Hamburg
4. Organization Name        (eg, company)   [Snake Oil,
 ↵ Ltd]:lokales-netz
5. Organizational Unit Name (eg, section)   [Webserver
 ↵ Team]:Webteam
6. Common Name              (eg, FQDN)
 ↵ [www.snakeoil.dom]:www.lokales-netz.de
7. Email Address  (eg, name@FQDN)
 ↵ [www@snakeoil.dom]:root@lokales-netz.de
_____________________________________________________________________

STEP 3: Generating X.509 certificate signed by Snake Oil CA
      ↵ [server.crt]
Certificate Version (1 or 3) [3]:3
Signature ok
subject=/C=DE/ST=Germany/L=Hamburg/O=lokales-
  ↵ netz/OU=Webteam/CN=www.lokales-netz/Email=root@lokales-
  ↵ netz.de
Getting CA Private Key
Verify: matching certificate & key modulus
read RSA private key
Verify: matching certificate signature
/etc/httpd/ssl.crt/server.crt: OK
_____________________________________________________________________

STEP 4: Enrypting RSA private key with a  pass phrase for
security [server.key]
The contents of the server.key file (the generated private key)
has to be kept secret. So we strongly recommend you to encrypt
the server.key file with a Triple-DES cipher and a Pass Phrase.
Encrypt the private key now? [Y/n]: n


Warning, you're using an unencrypted RSA private key.
Please notice this fact and do this on your own risk.
_____________________________________________________________________
RESULT: Server Certification Files
```

```
o  conf/ssl.key/server.key
   The PEM-encoded RSA private key file which you configure
   with the 'SSLCertificateKeyFile' directive (automatically
   done when you install via APACI). KEEP THIS FILE PRIVATE!

o  conf/ssl.crt/server.crt
   The PEM-encoded X.509 certificate file which you configure
   with the 'SSLCertificateFile' directive (automatically done
   when you install via APACI).

o  conf/ssl.csr/server.csr
   The PEM-encoded X.509 certificate signing request file
   which you can send to an official Certificate Authority (CA)
   in order to request a real server certificate (signed by
   this CA instead of our demonstration-only Snake Oil CA) which
   later can replace the conf/ssl.crt/server.crt file.

WARNING: Do not use this for real-life/production systems
```

Dies erzeugt ein Server-Zertifikat, das die fiktive *Snake Oil CA* zertifiziert. Nach einem Neustart von Apache können Sie den Zugriff testen.

Bei einem Aufruf von `https://192.168.1.2` fragt Mozilla, ob Sie das unbekannte Zertifikat annehmen wollen. Wenn Sie einige Male auf *weiter* geklickt haben, können Sie die Startseite des SSL-Servers sehen.

Abbildung 6.8: Sichere Verbindung im Mozilla

Das hervorgehobene Schloss in der rechten unteren Ecke signalisiert dann eine gesicherte Verbindung.

Beim Internet Explorer muss man nur dreimal auf *Ja* klicken, um das neue Zertifikat anzunehmen und die Startseite angezeigt zu bekommen.

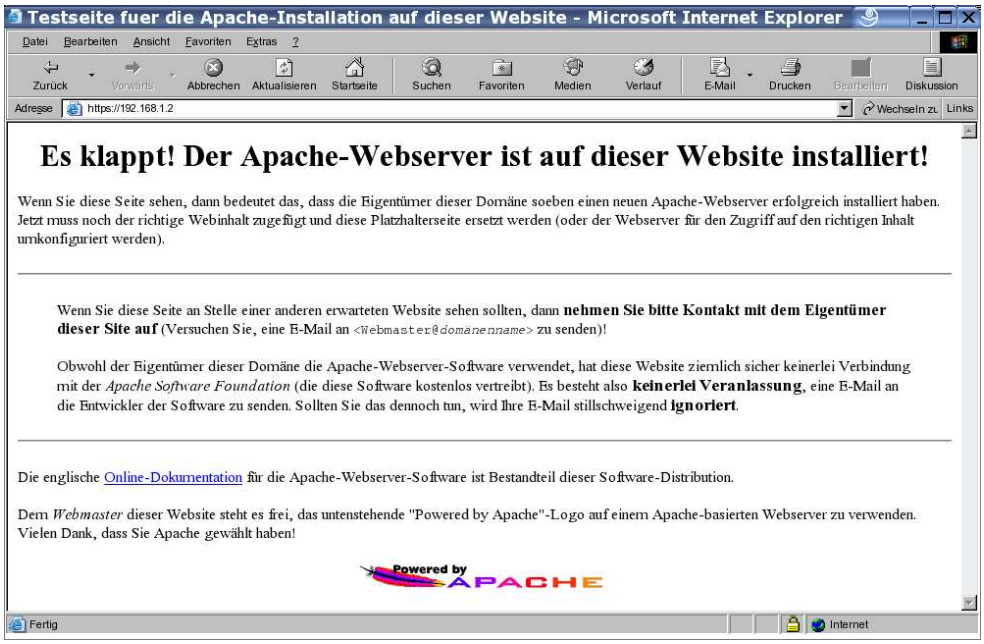

Abbildung 6.9: Sichere Verbindung im Internet Explorer

Abbildung 6.10: Das Zertifikat

Das selbst erstellte Zertifikat ist nur für ein Jahr gültig. Wer mehr über das Zertifikat erfahren möchte, sollte seinen Browser neu starten. Hier im Beispiel ist das Zertifikat vom Mozilla Browser bisher nur für die aktuelle Sitzung angenommen. Auf der zweiten Seite, beim Akzeptieren des Zertifikates, gibt es einen Knopf *Zertifikat untersuchen*. Klickt man diesen an, kann man Details des Zertifikates sehen.

Auch im Internet Explorer kann man Details über das Zertifikat ansehen, bevor man es annimmt.

Abbildung 6.11: Das Zertifikat im Internet Explorer

Auf der dritten Dialogseite kann man wählen, wie lange der Browser das Zertifikat akzeptieren soll. In der Voreinstellung ist das Zertifikat nur für die aktuelle Sitzung gültig. Wenn man mit dem erzeugten Zertifikat zufrieden ist, kann man es ruhig auch unbefristet annehmen. Dann erscheint der Dialog erst nach einem Jahr wieder, wenn Sie das Zertifikat erneuert haben.

Dieser Teil des Kapitels konnte nur die allerwichtigsten technischen Fragen zu Zertifikaten streifen und Ihnen helfen, ein funktionsfähiges Testsystem einzurichten. Für ein reales System benötigen Sie wie oben bereits erwähnt eine offizielle Zertifizierung.

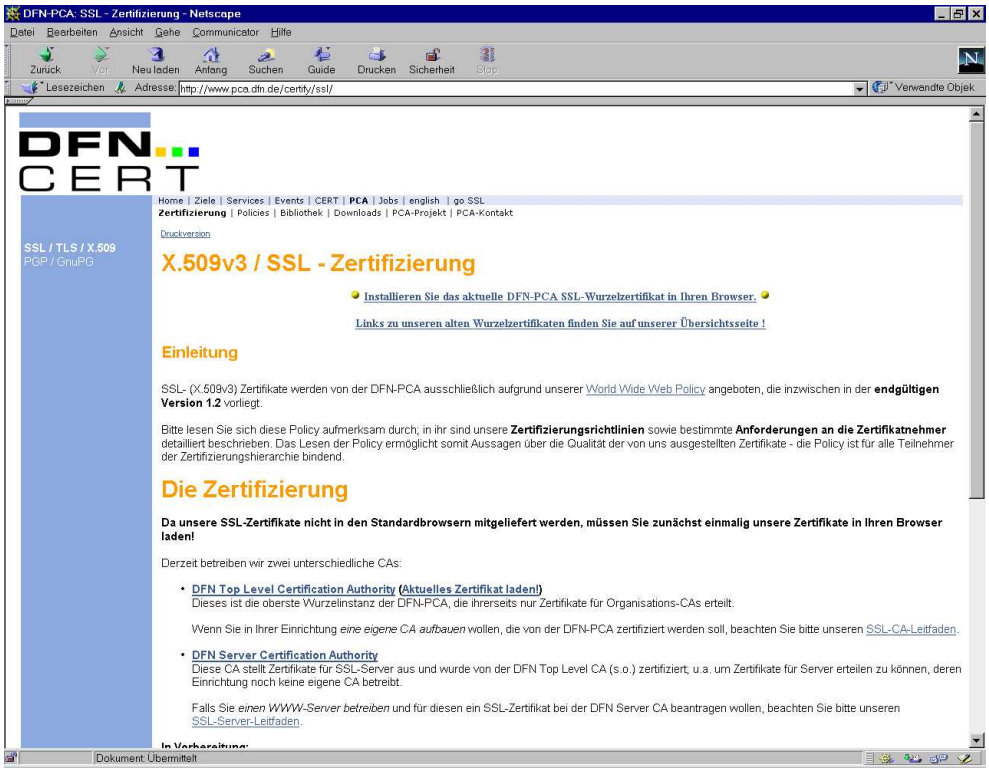

Abbildung 6.12: SSL-Zertifizierung beim DFN

6.8 Zugriffe protokollieren und auswerten

Betreiber von Web-Sites möchten gern wissen, ob ihr Web-Server anständig funktioniert und was die Besucher auf der Web-Site treiben.

Apache protokolliert alle Zugriffe in der Datei `/var/log/apache2/access_log`. Geben Sie im Browser die URL `http://192.168.1.2` ein, trägt Apache Folgendes in die Log-Datei ein:

```
192.168.1.56 - - [23/Nov/2004:14:01:39 +0100]
  ↵ "GET / HTTP/1.1" 200 1456
192.168.1.56 - - [23/Nov/2004:14:01:40 +0100]
  ↵ "GET /apache_pb.gif HTTP/1.1" 200 2326
192.168.1.56 - - [23/Nov/2004:14:26:52 +0100]
  ↵ "GET / HTTP/1.1" 200 1456
192.168.1.56 - - [23/Nov/2004:14:26:52 +0100]
  ↵ "GET /apache_pb.gif HTTP/1.1" 200 2326
192.168.1.56 - - [23/Nov/2004:14:28:55 +0100]
  ↵ "GET / HTTP/1.1" 200 2202
192.168.1.56 - - [23/Nov/2004:14:28:55 +0100]
  ↵ "GET /apache_pb.gif HTTP/1.1" 200 2326?
```

Die erste Zeile dieser Einträge ist folgendermaßen zu lesen:

Eintrag	*Bedeutung*
`192.168.1.56`	IP-Nummer des Client-Rechners, hier ein Rechner aus dem lokalen Netz.
`23/Nov/2004:14:01:39 +0100`	Datum und Uhrzeit. Da im November in Deutschland die Sommerzeit nicht gilt, weicht die Zeit um +1 Stunde von der Standardzeit (*GMT*) ab.
`"GET/HTTP/1.0"`	Die Datei `/srv/www/htdocs/index.html` oder falls nicht vorhanden das Inhaltsverzeichnis wird mit dem Protokoll `HTTP 1.0` übertragen.
`200`	Die Datei wurde erfolgreich übertragen.
`1456`	Größe der übertragenen Datei in Bytes.

Tabelle 6.2: Erklärung der Einträge in der Datei `/var/log/apache2/access_log`

Bei einer fehlerhaften Anfrage wie `http://192.168.1.2/nichtda.htm` schreibt der Apache folgende Meldung in die `access_log`:

```
192.168.1.56 - - [23/Nov/2004:14:40:22 +0100] "GET /
  ↵ nichtda.htm HTTP/1.1" 404 1042 "-" "Mozilla/
  ↵ 5.0 (X11; U; Linux i686; de-AT; rv:1.6) Gecko/20040114"
```

Statt des Codes `200` für eine erfolgreiche Datenübertragung taucht hier `404` für `File does not exist` auf.

Der Inhalt der Log-Datei ist sehr aussagekräftig und gut für statistische Auswertungen nutzbar.

Fehler protokolliert der Apache zusätzlich in der Datei `/var/log/apache2/error_log`. Nach der fehlerhaften Anfrage hat sie folgenden Inhalt:

```
[Tue Nov 23 14:00:52 2004] [notice] suEXEC mechanism enabled
(wrapper: /usr/sbin/suexec2)
[Tue Nov 23 14:00:56 2004] [notice] Apache/2.0.50
(Linux/SUSE) configured -- resuming normal operations
[Tue Nov 23 14:40:22 2004] [error] [client 192.168.1.1]
File does not exist: /srv/www/htdocs/nichtda.htm
```

Die ersten Zeilen hat der Apache beim Start erstellt. Hier können Sie u. a. erkennen, dass er ohne Fehler starten konnte.

In der letzten Zeile finden Sie die Fehlermeldung als Folge einer fehlerhaften Anforderung.

Tipp: Wenn Sie eigene CGI-Programme erstellen, können Sie dieser Datei die Fehlermeldungen Ihrer Programme finden.

6.9 Auswertung mit Webalizer

Wenn Ihnen die manuelle Auswertung der Log-Dateien nicht ausreicht, können Sie mit Analyse-Tools übersichtliche Statistiken erstellen.

Ein sehr weit verbreitetes Analyse-Tool ist das Programm *Webalizer*, das Sie im gleichnamigen Paket finden, aber nur auf der DVD, nicht auf den CDs.

Bei der Installation über YaST ist *Webalizer* nicht direkt über eine der Selektionen zu finden, benutzen Sie hier bitte die Suchfunktion.

Das Programm liefert eine Übersicht über die Nutzung des Web-Servers in den letzten 12 Monaten.

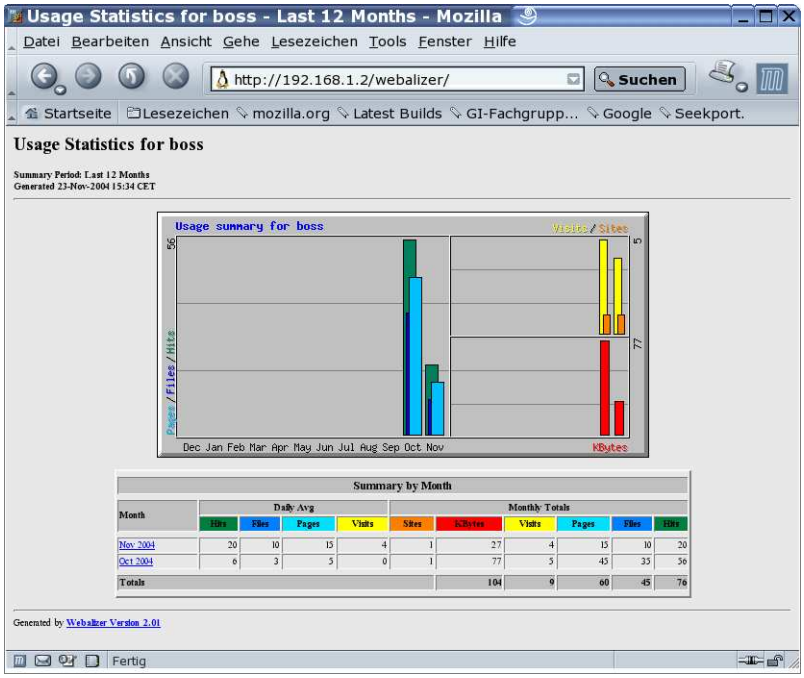

Abbildung 6.13: *Webalizer*: Monatsübersicht

Die Übersicht vergleicht die Monatsdaten. Die Summen und Durchschnittswerte beziehen sich auf einen einzelnen Tag.

6.9.1 Monatliche Auswertung

Klicken Sie in dieser Übersicht auf einen der Monate, so erhalten Sie eine viel umfangreichere Auswertung für den ausgewählten Monat. In dieser Auswertung finden Sie

- eine Zusammenfassung für den aktuellen Monat,
- die Zugriffsstatistik, aufgeschlüsselt nach den einzelnen Tagen des Monats,

- eine Statistik, aufgeschlüsselt nach Uhrzeiten,
- eine Auswertung der am häufigsten abgerufenen Seiten,
- eine Liste Ihrer häufigsten Einstiegsseiten,
- eine Liste der häufigsten Ausstiegsseiten,
- eine Zusammenstellung, welche Rechner Ihren Server aufgesucht haben,
- eine sehr interessante Liste der Adressen, von denen Ihre Besucher gekommen sind,
- vor, wenn Besucher über Suchmaschinen zu Ihnen gekommen sind, welche Suchbegriffe sie erfolgreich benutzt haben,
- vor, als welche Browser sich die Browser Ihrer Besucher ausgeben und
- vor, aus welchen Ländern die Besucher kommen.

Viele Informationen bereitet der *Webalizer* sowohl als Tabelle als auch als Grafik auf.

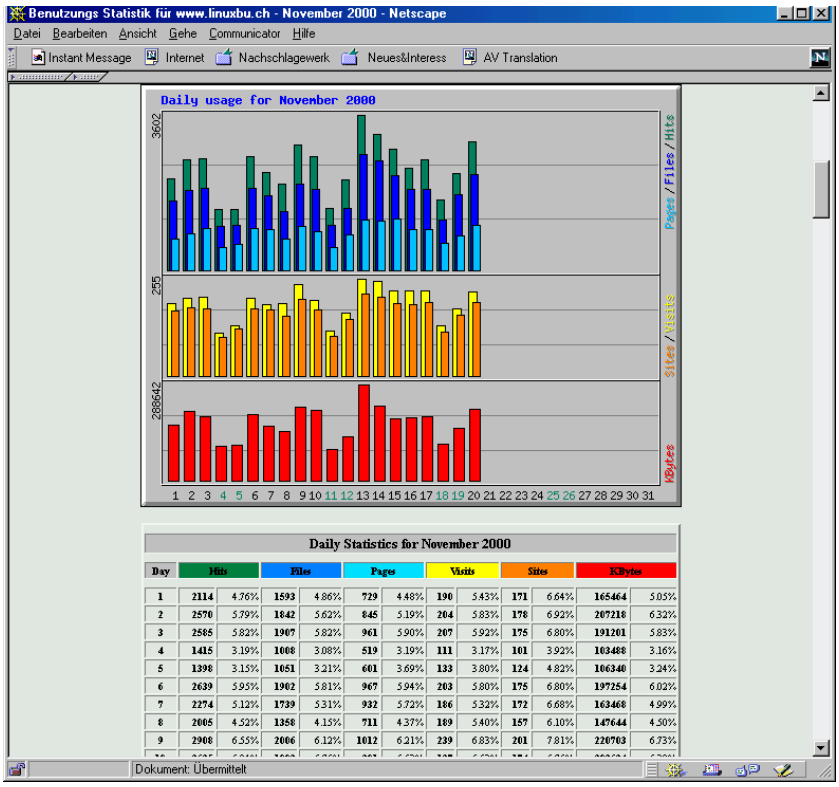

Abbildung 6.14: Tagesstatistik

Die Informationen aus den Auswertungen helfen, gezielt auf die Interessen und Gewohnheiten der Besucher der Web-Site einzugehen.

6.9.2 Konfiguration von Webalizer

Zum Konfigurieren von *Webalizer* müssen Sie nur die Datei `/etc/webalizer.conf` an Ihre Bedürfnisse anpassen.

`/etc/webalizer.conf` (Dateianfang)

```
#
# Sample Webalizer configuration file
# Copyright 1997-2000 by Bradford L. Barrett (brad@mrunix.net)
#
…
#
# LogFile defines the web server log file to use.  If not
# specified here or on on the command line, input will default
# to STDIN.  If the log filename ends in '.gz' (ie: a gzip
# compressed file), it will be decompressed on the fly as it
# is being read.

LogFile       /var/log/httpd/access_log

# LogType defines the log type being processed.  Normally, the
# Webalizer expects a CLF or Combined web server log as input.
# Using this option, # you can process ftp logs as well
# (xferlog as produced by wu-ftp and # others), or Squid
# native logs.  Values can be 'clf', 'ftp' or 'squid',
# with 'clf' the default.

#LogType        clf

# OutputDir is where you want to put the output files.  This
# should be a full path name, however relative ones might work
# as well. If no output directory is specified, the current
# directory will be used.

OutputDir      /var/lib/webalizer
```

Wichtig ist hier vor allem der Pfad zum Apache-Logfile:

```
LogFile        /var/log/apache2/access_log
```

Wenn Ihre Statistik allgemein zugänglich sein soll, ist es geschickter, statt `/var/lib/webalizer` das ebenfalls von der SuSE-Installation angelegte Verzeichnis `/srv/www/htdocs/webalizer` zu nutzen. Ändern Sie also an dieser Stelle die Konfigurationsdatei.

```
OutputDir      /srv/www/htdocs/webalizer
```

Der *Webalizer* ist dann ohne weitere Änderung sofort einsatzbereit. Starten Sie das Programm von der Konsole aus, indem Sie

```
webalizer
```

eingeben. Sobald das Programm die Reports erzeugt hat, können Sie in einem beliebigen Browser das Ergebnis unter der URL

```
http://192.168.1.2/webalizer/
```

aufrufen. Bei einem neu installierten System wird die Auswertung noch nicht sehr umfangreich sein, aber das kann sich ja im Laufe der Zeit ändern.

Die Konfigurationsdatei können Sie sehr leicht an Ihre Bedürfnisse anpassen, sie ist sehr gut und ausführlich dokumentiert.

6.9.3 Webalizer automatisieren

Da der *Webalizer* sehr schnell ist und Ihr System nicht unnötig belastet, können Sie ihn täglich starten. Dazu bietet sich ein Cronjob wie im folgenden Auszug aus der Crontab von `root` an:

```
PATH=/bin:/usr/bin:/usr/local/bin:/sbin:/root/bin:/root/sbin
mailto=root

50 23 * * * webalizer
```

Rufen Sie den *Webalizer* täglich kurz vor Mitternacht auf, da bei SuSE-Systemen Cron um Mitternacht einen Job startet, der die Länge von Log-Dateien überwacht und diese gegebenenfalls stutzt. Wenn Sie den *Webalizer* erst danach starten, fehlen Ihnen die Zugriffe zumindest des letzten Tages, was hässliche Lücken in der Statistik hinterlässt.

Damit der *Webalizer* seine Auswertungen speichert, sollten Sie unbedingt die `webalizer.conf` bearbeiten.

`/etc/webalizer.conf` (Auszug ab Zeile 54)

```
# Incremental processing allows multiple partial
# log files to be used
# instead of one huge one.
# Useful for large sites that have to rotate
# their log files more than once a month.
# The Webalizer will save its
# internal state before exiting,
# and restore it the next time run, in
# order to continue processing where it left off.
# This mode also causes
# The Webalizer to scan for and ignore
# duplicate records (records already
# processed by a previous run).
# See the README file for additional
# information.  The value may be 'yes' or 'no',
```

```
# with a default of 'no'.
# The file 'webalizer.current' is used to
# store the current state data,
# and is located in the output directory of
# the program (unless changed
# with the IncrementalName option below).
# Please read at least the section
# on Incremental processing in the README file
# before you enable this option.

#Incremental    no
```

Damit der *Webalizer* den Status der bisherigen Auswertungen speichert, ändern Sie die hervorgehobene Zeile in:

```
Incremental    yes
```

Falls dann Cron die Log-Dateien des Apache verkürzt, bleiben die Informationen über die vergangenen Wochen und Monate trotzdem erhalten. Wenn Sie die Voreinstellung beließen, würde *Webalizer* immer nur die Informationen darstellen, die sich aktuell in der Apache-Log-Datei befänden.

Webalizer kann nicht nur die Statistiken des Web-Servers auswerten, sondern auch die des FTP-Servers und des Proxy-Servers. Sie werden daher in den entsprechenden Kapiteln erneut auf dieses Programm stoßen.

6.10 Eine eigene Suchmaschine mit htdig

Wenn Ihre Web-Site anfängt zu wachsen, taucht schnell der Wunsch nach einer eigenen Suchmaschine auf. Eine Suchmaschine ermöglicht den Nutzern Ihrer Web-Site, Informationen gezielt auf Ihrem Web-Server zu suchen, unabhängig von der vorgegebenen Navigationsstruktur.

Ein sehr leistungsfähiges, aber trotzdem einfach zu konfigurierendes Programm ist *ht://Dig*, dessen aktuellste Version Sie im Web an der Adresse `http://www.htdig.org/` finden. Die aktuelle SuSE-Distribution installiert dieses Programm normalerweise mit. Ansonsten können Sie es über die Selektion *Einfacher Webserver* finden.

6.10.1 Konfiguration von ht://Dig

Die Konfigurationdatei finden Sie unter `/srv/www/htdig/conf/htdig.conf`, hier müssen Sie nur wenig ändern, vor allem Ihre Start-URL:

```
#
# Example config file for ht://Dig.
#
```

```
# This configuration file is used by all the programs that
# make up ht://Dig.
# Please refer to the attribute reference manual for more
# details on what can be put into this file.
# (http://www.htdig.org/confindex.html)
# Note that most attributes have very reasonable default values
# so you really only have to add attributes here if you want to
# change the defaults.
#
# What follows are some of the common attributes you might want
# to change.
#

#
# Specify where the database files need to go.  Make sure that
# there is plenty of free disk space available for the database.
# They can get pretty big.
#
database_dir:           /var/lib/htdig/db

#
# This specifies the URL where the robot (htdig) will start.
# You can specify
# multiple URLs here.  Just separate them by some whitespace.
# The example here will cause the ht://Dig homepage and related
# pages to be indexed.
# You could also index all the URLs in a file like so:
# start_url:            `${common_dir}/start.url`
#
start_url:              http://www.htdig.org/
```

Mit der vorgegebenen Einstellung würden Sie die Web-Site von `www.htdig.org` durchsuchen. Um die Suche in Ihrem lokalen Web-Server zu erlauben, ändern Sie die URL-Zeile dem Beispiel dieses Buchs folgend in:

```
start_url:              http://192.168.1.2/
```

Etwas später in der Konfigurationsdatei finden Sie den Abschnitt:

```
#
# The string htdig will send in every request to identify the
# robot. Change this to your email address.
#
maintainer:           unconfigured@htdig.searchengine.maintainer
Diese Mail-Adresse hinterlässt ht://Dig in den Log-Dateien der
besuchten Web-Server, von daher sollte sie auf Ihr System
verweisen.
#
```

```
# The string htdig will send in every request to identify the
# robot.  Change
# this to your email address.
#
maintainer:           debacher@boss.lokales-netz.de
```

Damit ist Ihre Suchmaschine bereits einsatzbereit.

Die Arbeit einer Suchmaschine besteht immer aus zwei Teilen:

- Indizieren der Seiten;
- beantworten von Suchanfragen.

6.10.2 Seiten indizieren

Bevor Sie das Suchen erlauben, müssen Sie zuerst einen Index für Ihre Suchmaschine aufbauen. Dazu rufen Sie an der Konsole auf:

```
/usr/bin/rundig
```

Dieser Befehl wird Ihren Linux-Server für einige Minuten beschäftigen. Der Zeitbedarf für das Indizieren hängt von der Leistungsfähigkeit und sonstigen Belastung des Linux-Servers und dem Umfang Ihrer Web-Site ab.

Sowie Ihre Suchmaschine zufriedenstellend funktioniert, sollten Sie das Indizieren der Web-Site über einen Cronjob automatisieren:

```
20 03 * * 7 /usr/bin/rundig
```

Hiermit aktualisieren Sie an jedem Sonntag um 03.20 h Ihren Suchindex. Beim Planen dieses Cronjobs sollten Sie bedenken, dass *ht://Dig* beim Indizieren alle Seiten laden und auswerten muss, was den Web-Server belastet.

6.10.3 Beantworten von Suchanfragen

Sowie Sie den Index einmal aufgebaut haben, können Sie auch Suchanfragen starten. Die dafür notwendige Programmkomponente `htsearch` finden Sie im Verzeichnis `/srv/www/cgi-bin/`. Geben Sie in Ihrem Browser die URL

```
http://192.168.1.2/cgi-bin/htsearch
```

ein.

Als Antwort sollten Sie folgende Seite erhalten:

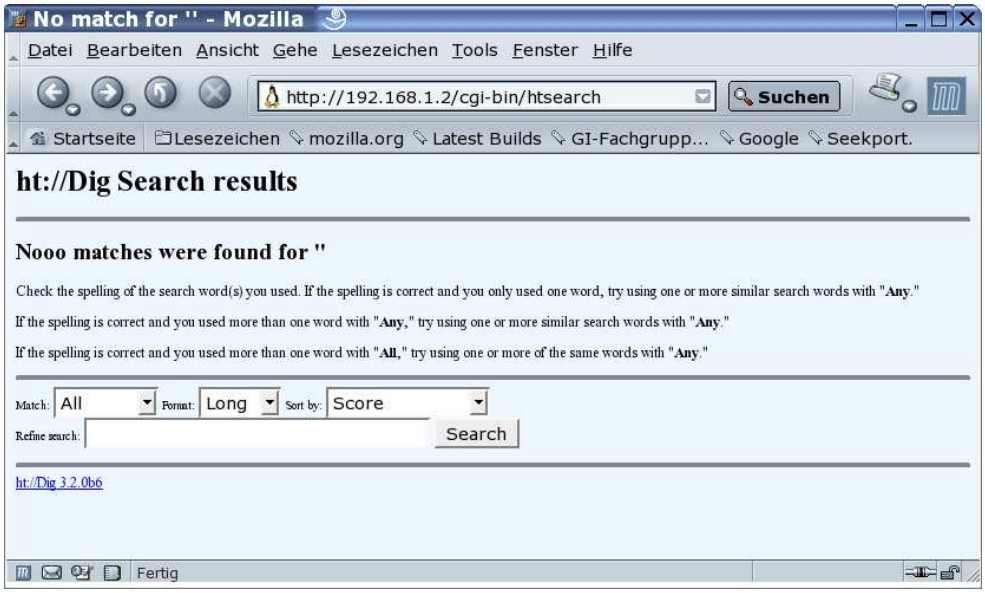

Abbildung 6.15: Erste Suche mit *ht://Dig*

Die Seite meldet einen Fehler, weil Sie dem Programm `htsearch` keinen Suchbegriff übergeben haben. Sie müssen dazu ein Formular im Stil Ihrer Web-Site erstellen, das den Suchbegriff übergibt. Für einen ersten Versuch können Sie das Eingabefeld auf der Seite mit der Fehlermeldung verwenden.

Das folgende Listing, das Sie im Verzeichnis `/srv/www/htdocs/suche.html` ablegen können, enthält ein Muster für ein eigenes Suchformular :

```
<html><head><title>Suche mit ht://Dig</title></head><body>
<h1>Suche mit ht://Dig</h1><hr noshade size="4">
<p>
<form method="get" action="/cgi-bin/htsearch">
<font size="-1">
Treffer: <select name="method">
<option value="and" selected>All
<option value="or">Any
<option value="boolean">Boolean
</select>

Format: <select name="format">
<option value="builtin-long">Long
<option value="builtin-short">Short
</select>

Sortiert nach: <select name="sort">
```

```
<option value="score" selected>Score
<option value="time">Time
<option value="title">Title
<option value="revscore">Reverse Score
<option value="revtime">Reverse Time
<option value="revtitle">Reverse Title
</select>

<br>Suchbegriff:
<input type="text" size="30" name="words" value="">
<input type="submit" value="Search">
</select>
</font>
</form>
</body></html>
```

Falls die Seiten, auf denen *ht://Dig* die Suchergebnisse präsentiert, nicht in Ihr Layout passen, können Sie diese nahezu beliebig anpassen. Vorschläge für passende HTML-Seiten finden Sie im Verzeichnis `/srv/www/htdig/common/`.

Ein Beispiel für die Anpassung der Suchmaschine an das eigene Layout ist die Suchfunktion auf den Seiten von `http://www.linuxbu.ch`.

Abbildung 6.16: *ht://Dig* im eigenen Layout

Die Suchfunktion fügt sich hier nahtlos in das Gesamtlayout ein.