Inhaltsverzeichnis

1. Vorwort / Aufgabenstellung

4

2. Projektvorbereitung

2.1 W-Fragen 5
2.2 Ziele 7
2.3 Literaturverzeichnis 8
2.4 Verlaufsplan 9

3. Projektergebnisse

3.1 Linux

3.1.1 Installation 10
3.1.2 Man beachte... 10
3.1.3 Kernel 10
3.1.4 Nach der Installation ist vor der Installation 11
3.1.5 Distributionen 11
3.1.6 Verzeichnisstruktur 12
3.1.7 Installation des Apache-Web-Servers und dessen Module mit YaST2 12
3.1.8 Benutzung des Konquerors als Dateimanager 13
3.1.9 Benutzung eines Texteditors zum Bearbeiten der Konfigurationsdateien 14
3.1.10 Die Datei hosts im Verzeichnis /etc 15
3.1.11 Die Datei networks im Verzeichnis /etc 16

3.2 Apache

3.2.1 Apache 17
3.2.2 Der Ursprung 17
3.2.3 Entwicklung des Apache 18
3.2.4 Leistungsmerkmale von Apache 18
3.2.5 Eigenschaften von Apache 19
3.2.6 Die httpd.conf 19
3.2.7 Die drei wichtigsten Funktionsabschnitte der httpd.conf 20
3.2.8 IP-basierte virtuelle Hosts 21
3.2.9 Zusammenhang zwischen IP-basierte virtuelle Hosts und boot.local 23
3.2.10 Namen-basierte virtuelle Hosts 25
3.2.11 Beispiel und Erläuterung eines virtuellen Hosts 25

3.3 Common Gateway Interface - CGI

3.3.1 Beschreibung von CGI 28
3.3.2 Die Kommunikation über ein CGI 28
3.3.3 Die Rechte eines CGI-Prozesses 29

3.4 PHP

3.4.1 Was ist PHP? 30
3.4.2 Zur Geschichte von PHP 30
3.4.3 PHP im Vergleich mit anderen bekannten Webentwicklungssystemen 31
3.4.4 Die Funktionsweise von PHP 33
3.4.5 Vorraussetzungen für PHP 33
3.4.6 Installieren des PHP-Moduls für Apache 33
3.4.7 Test einer PHP-Seite 34
3.4.8 Ein Beispiel 35
3.4.9 .php, .php3, .php4 36
3.4.10 Mime-Typen 38
3.4.11 Die Performance von PHP zu Perl im Vergleich 38
3.4.12 PHP als CGI oder Modul 39
3.4.13 Anzeige der Konfiguration des PHP-Interpreters 40
3.4.14 Die php.ini 40
3.4.15 Was safe_mode bewirkt 41
3.4.16 Parameterweitergabe einer PHP-Seite an ein anderes Skript 41
3.4.17 Anzeigen des PHP-Codes im Browser 42
3.4.18 Kompilieren von PHP-Dateien und Schutz vor Dritten 42

3.5 Web-Account erstellen und verwalten

3.5.1 Definition 43
3.5.2 FTP 43
3.5.3 Das Layout der Seiten 46
3.5.4 Erläuterung des PHP-Gästebuchs 48

4. Aufgabenverteilungsplan

52

5. Protokolle der einzelnen Projektsitzungen

53

6. Erfahrungsberichte

6.1 Thorsten Behrend 66
6.2 Felix Hock 67
6.3 Christian Keller 68
6.4 Bernd Wissmeier 69

7. Abschlusserklärung

70

1. Vorwort / Aufgabenstellung

Alle kennen es und die Meisten nutzen es oder wissen, wie man es benutzt – das Internet. Doch wie funktioniert diese weltweit ausgedehnte Informationsmaschine? Mit diesem Projekt wollen wir einen Teilaspekt des Ganzen ausleuchten und ein wenig den Vorhang lüften. Auf den nächsten Seiten beschreiben wir das Prinzip eines im Internet eingebundenen Servers, der HTML und dazugehörige Skripte verwalten kann. Der Blick hinter die Kulissen ist zwar interessant, aber auch recht schwer. Deshalb versuchen wir anschaulich und mit einfachen Mitteln die Sachverhalte klar darzustellen und Zusammenhänge einfach zu erklären.

Unsere Vorgehensweise bestand darin, den Web-Server als Solches nicht isoliert zu betrachten, sondern von Grund auf das System, in dem er arbeitet, ebenfalls zu beleuchten.

Außerdem gehen wir detailliert auf die Konfigurationen eines Apache-Web-Servers ein. Da es hier einige Einstellungsvarianten gibt, haben wir versucht alle zu beachten und einzubinden.

Des weiteren erklären wir das Zusammenspiel von Skriptsprachen mit dem Apache-Web-Server anhand von PHP. Heutzutage findet man kaum noch Webseiten ohne diesen dynamischen Inhalt. Deshalb soll dieses Projekt auch diesen Aspekt, mit Hilfe von kleinen und einfachen PHP-Skripten, behandeln, durch die wir die Zusammenhänge aufzeigen.

CGI (Common Gateway Interface) ist eine Schnittstelle für die Ausführung von Programmen auf dem Web-Server, die in verschiedenen Programmiersprachen wie z.B. C,C++, oder Java oder auch in speziellen Skriptsprachen wie Perl oder PHP geschrieben sein können. Ein wichtiger Aspekt ist die Sicherheit im Umgang und Einsatz dieser CGIs. Leider müssen wir an dieser Stelle feststellen, dass eine ausführliche Beschreibung dieses Aspekts unseren Projektrahmen bei weitem gesprengt hätte. Dieses Thema ist allein schon ein Projekt Wert.

Mit Hilfe von Skriptsprachen entsehen dynamische Webseiten, d.h. Seiten, deren Inhalt an die vom Client empfangenen Informationen angepasst werden kann. Wir haben uns innerhalb unseres Projektes für die Scriptsprache PHP entschieden. Dabei wollen wir einen Apache-Web-Server mit einer Grundkonfiguration starten, Webspace für einen imaginären User zur Verfügung stellen und diesem „Kunden“ das Verwalten von dynamischen Webseiten ermöglichen.

Die Dokumentation schließt mit der Betrachtung eines Nutzers eines Web-Servers. Damit soll der übliche Ablauf der Veröffentlichung einer Homepage unter realen Bedingungen geschildert werden.

Die Projektmitglieder danken an dieser Stelle den Menschen, die uns immer zur Seite standen und vieles möglich machten. Wir danken auch der Schule, dass sie uns die nötige Zeit und das Material bereitstellten.

2.1 W-Fragen

Was

Konfiguration eines Apache-Web-Servers unter SuSE-Linux. Des weiteren soll ein PHP- oder Perl-Skript geschrieben, dokumentiert und zum laufen gebracht werden

Wer

Die Gruppenmitglieder sind Thorsten Behrend, Felix Hock, Christian Keller und Bernd Wissmeier

Warum

In der heutigen Zeit ist das Internet unabdingbar geworden. Das Internet dient nicht mehr nur als Informationsquelle. Mehr und mehr erhält die Interaktivität ihren Einzug. Dazu gehören u.a. Formulare die es möglich machen eine Bestellung über das World Wide Web zu tätigen. Als Medienassistenten ist es von äußerster Wichtigkeit Kenntnisse von der Gestaltung bis zur Verwaltung dieser Internetseiten zu haben.

Deshalb haben wir uns entschieden einen Server für Internetseiten zu konfigurieren und auf diesem Server statische und dynamische Internetseiten zu verwalten. Die statischen Internetseiten begrenzen dabei auf die Scriptsprache PHP. Der Server wird mit Apache unter SuSE Linux laufen.

Wozu

Hierzu haben wir unsere Ziele definiert

Mit wem

Projektbetreuer sind Herr Scheib und Herr Schorr

Für wen

Auftraggeber ist die BBS Neustadt

Wie

Wir haben uns entschieden, uns frei an das Skript zu halten, das wir im Fach Projektmanagement erarbeitet haben

Womit

Die Auswahl der Informationsmaterialien und -quellen ist uns freigestellt

Wann

Projektsitzungen sind vorläufig jeden Montag und Mittwoch

Wo

Die Sitzungen finden entweder in der Schule oder bei einem der Gruppenmitglieder zuhause statt.

2.2 Ziele

Im Rahmen unserer Ausbildung zum Medienassistent wird das Thema Internetdesign (Gestaltung und Technik) in verschiedenen Fächern besprochen und angewandt. Dazu gehört das Fach Informatik, indem HTML als Grundlage zur Website-Gestaltung erlernt wird. Des weiteren das Fach Mediengestaltung mit dem Erlernen der gestalterischen Grundlagen.

Immer mehr Unternehmen setzen auf das Internet als Schnittstelle zwischen sich und Kunden bzw. Zulieferern. Dabei genügen keinesfalls mehr statische Websites die nur dazu dienen, Informationen bereitzustellen.

Um eine Produktdatenbank bereitzustellen, eine Kundenverwaltung über das Internet abzuwickeln etc. benötigt man dynamisch veränderbare Webseiten.

CGI (Common Gateway Interface) ist eine solche Schnittstelle für die Ausführung von Programmen auf dem Web-Server, die in verschiedenen Programmiersprachen wie z.B. C,C++, oder Java oder auch in speziellen Skriptsprachen wie Perl oder PHP geschrieben sein können.

Wir haben uns innerhalb unseres Projektes für die Scriptsprache PHP entschieden. Dabei wollen wir einen Apache-Web-Server mit einer Grundkonfiguration starten, festgelegten Festplattenspeicher auf dem Server einem imaginären User zur Verfügung stellen und diesem Kunden das Verwalten von dynamischen Webseiten ermöglichen.

2.4 Literaturverzeichnis

- Apache, Dr. Susanne Wigard, bhv-Verlag, Landsberg, 2000
- Apache und CGI – administration und programmierung, Rich Bowen und Ken Coar, Markt und Technik Verlag, München, 2000
- Linux – Apache Web Server Administration, Charles Aulds, Sybex Verlag, Düsseldorf, 2001
- PHP 4, Mark Kronsbein und Thomas Weinert, Sybex Verlag, Düsseldorf, 2001
- PHP 4, Uwe Hess und Günter Karl, bhv Verlag, Landsberg, 2000
- PHP, Rasmus Lerdorf, O`Relly Verlag, Köln, 2000
- Apache, Andrew Ford, O`Relly Verlag, Köln, 2000
- Linux, Michael Kofler, Addison-Wesley Verlag, München, 2001
- Kommunikation unter Linux, Rainer Krienke, Suse GmbH Verlag, Nürnberg, 2000

2.5 Verlaufsplan

09.11.01
Themenabgabe

↓

13.11.01
Zieldefinition
Literaturbeschaffung
Grobgliederung

↓

27.11.01
Einrichtung des Apache-Web-Servers
Rücksprache mit allen in Raum 210 arbeitenden Gruppen

↓

10.12.01
„laufendes“ PHP-Skript

↓

18.12.01
Aufgabenverteilungsplan
Zwischenpräsentation

↓

18.12.01 – 14.01.02

Thorsten Behrend	Felix Hock	Christian Keller	Bernd Wissmeier
Apache	httpd.conf	Linux	PHP 4

↓

14.01.02 –18.01.02
Erste Intensivphase
Zusammentragen sämtlicher Informationen
- Feingliederung
- Rücksprachen mit Herrn Scheib
Rücksprache mit allen in Raum 210 arbeitenden Gruppen

↓

21.01.02 – 25.01.02
Zweite Intensivphase
Komplettierung und Ausdruck des Skripts
Rücksprache mit allen in Raum 210 arbeitenden Gruppen
Erste Überlegungen zur Präsentation

↓

01.02.02
Projektabgabe

3.1. Linux

3.1.1 Installation

Auf eine Installation von Linux sollte man vorbereitet sein. Wer das erste Mal mit Linux in Kontakt kommt, könnte durch manche Einstellung verwirrt werden, die vor der Installation abgefragt wird. Damit auch der nicht versierte User eine Linux-Installation durchführen kann, gibt es Distributionen. Das sind Sammlungen von Softwarepaketen oder Programmen, die meist frei erhältlich sind (*Public License*) und nur der Einfachheit halber gesammelt und auf CDs oder DVD gebrannt wurden. Diese bieten einen Installationsmanager und führen den User mit sicherer Hand durch die vielen Einstellungen der Installation. Mit Hilfe der Standardeinstellungen lässt sich damit ein ganz ordentliches System installieren, mit dem gut zu arbeiten ist.

3.1.2 Man beachte ...

Vielleicht empfiehlt es sich aber von vornherein bestimmte Einstellungen zu tätigen, die im Nachhinein nicht mehr oder nur schwer zu verändern sind. Ich denke da zum Beispiel an die Wahl des Dateisystems (Ext2/ReiserFS...) oder die Größe der Partitionen und Vergabe der Mountpoints. WindowsUser sind ganz verwundert über die Vielfalt der Einstellungen. Nicht umsonst fällt das in den Bereich der Experten.

3.1.3 Kernel

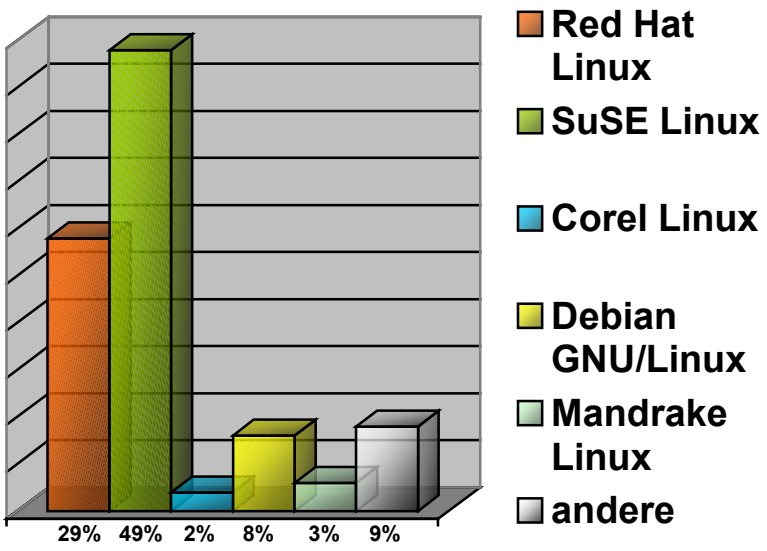
Bei Standard PCs verwendet man am besten Standard-Kernel. Diese sind meist genau auf diese zugeschnitten und sprechen die Hardware auch optimal an. Doch bei Servern oder ähnlich starken Maschinen müssen besondere Kernel her. Vielleicht ist ein selbst kompilierter Kernel sogar die beste Lösung. Doch was ist das überhaupt, ein Kernel? Er ist sozusagen des Pudels Kern, wobei Linux hier dem Pudel entspricht. Er sorgt dafür, dass sich die Programme und Anwendungen mit der Hardware verstehen, eine Kommunikationsleitstelle sozusagen.

3.1.4 Nach der Installation ist vor der Installation

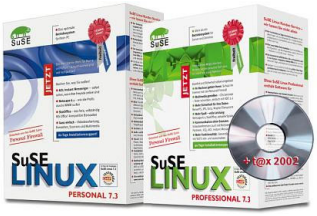
Ausgehend von einer Standardinstallation kann man Pakete einfach nachinstallieren oder vorhandene (unnötige) deinstallieren. Auch hier schaut der elektronische Helfer einem auf die Finger, damit für das System notwendige Dateien nicht unachtsam gelöscht werden und achtet auch auf die Gültigkeit der Paketabhängigkeiten zueinander.[1]

3.1.5 Distributionen

In Deutschland sind die Distributionen von SuSE[2] seit langem Marktführer. Eine aktuelle Umfrage auf *http://www.linux.de* mit über 25.000 Beteiligten zeigt eine repräsentative Verteilung der Massen.



Anteile der Distributionen in Deutschland



Zur Zeit bietet **SuSE** für 78,90 Euro seine **Version 7.3** an. Sie beinhaltet die aktuellen Kernel, Basispakete und zusätzliche 1500 Programme auf 7 CDs und einer DVD. Diese Version gibt es für Schüler und Studenten verbilligt für 49,90 Euro (*Stand Januar 2002*). Der Preis ergibt sich hauptsächlich aus den beiliegenden Büchern. Insgesamt 5 Handbücher bestehend aus einem „Quick-Install-Manual“, einem Buch zur „Konfiguration“, eine Beschreibung über „Die Programme“, ein Buch zur „Referenz“ und eines über „Netzwerke“. Mit Hilfe dieser grundlegenden Literatur lässt sich das System weit besser verstehen und nutzen.

Ein grundsätzliches Verständnis von Linux oder allgemein von unix-basierende Systemen ist von Vorteil, da man bei Problemen als Unwissender recht schnell an seine Grenzen stößt. Mit Hilfe der Bücher kann wohl so manches Problem gelöst werden, doch eine Frage bleibt: Erkennt man ein Problem insoweit, dass man weiß, wo im Handbuch nachzuschlagen ist?

---

[1] vgl:3.1.7 Installation des Apache Servers und dessen Module mit YaST2

[2] *http://www.suse.de*

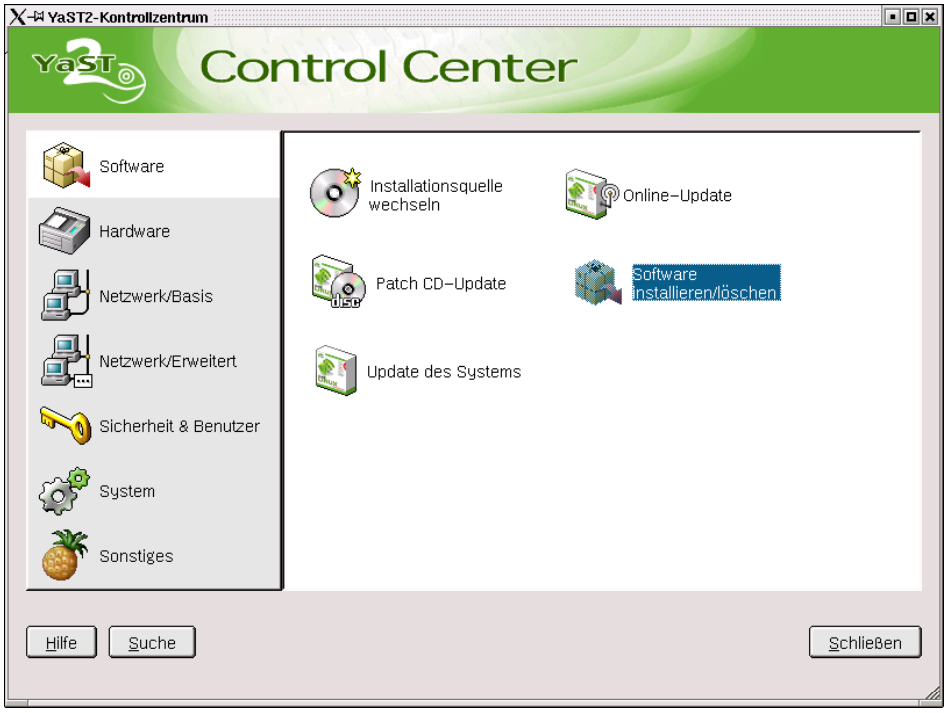
3.1.6 Verzeichnisstruktur

An dieser Stelle sollten wir noch auf die Verzeichnisstruktur eingehen. Linux baut auf einem *logischen* Dateisystem auf. Das heißt der Anwender arbeitet nicht auf Festplatten und Partitionen, sondern sieht immer nur die ihm schon bekannte Verzeichnisstruktur. Zum Vergleich: In Windows sieht man Festplatten und Partitionen als Buchstaben sozusagen *physikalisch* vor sich (C: / D: ...)

Der Verzeichnisbaum von Linux

3.1.7 Installation des Apache Servers und dessen Module mit YaST2

Als erstes rufen wir im SuSE-Menü das YaST2-Kontrollzentrum und danach in der Abteilung ***Software*** den Punkt ***Software installieren/löschen*** auf.



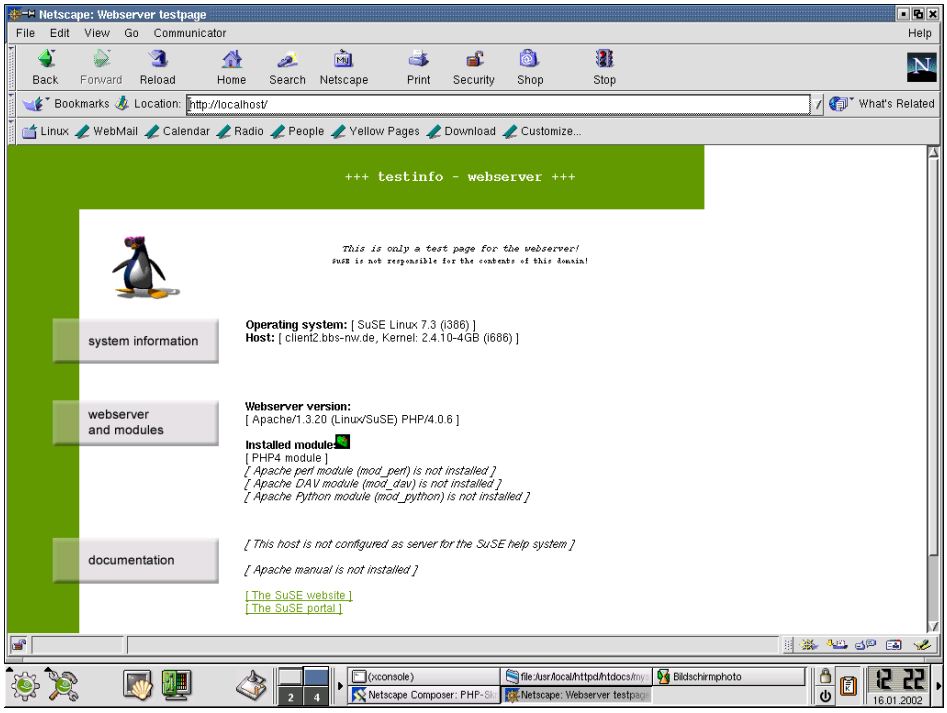
Das YaST2 Kontrollzentrum

Nun befinden wir uns in einem Modul, mit dessen Hilfe man Zugriff auf alle Pakete zum Installieren und Deinstallieren bekommt.

Die benötigten Pakete befinden sich unter der Paketserie **n** und heißen **apache**, **apache-contrib, mod_php4 und mod_php4-core**. Mit einem Klick auf die entsprechenden Pakete werden sie installiert (es erscheint ein großes X vor dem jeweiligen Paket). Mit **OK** wird alles angewählte installiert.

Nun startet man die Netzwerkdienste neu oder führt der Einfachheit halber einen Reboot durch.

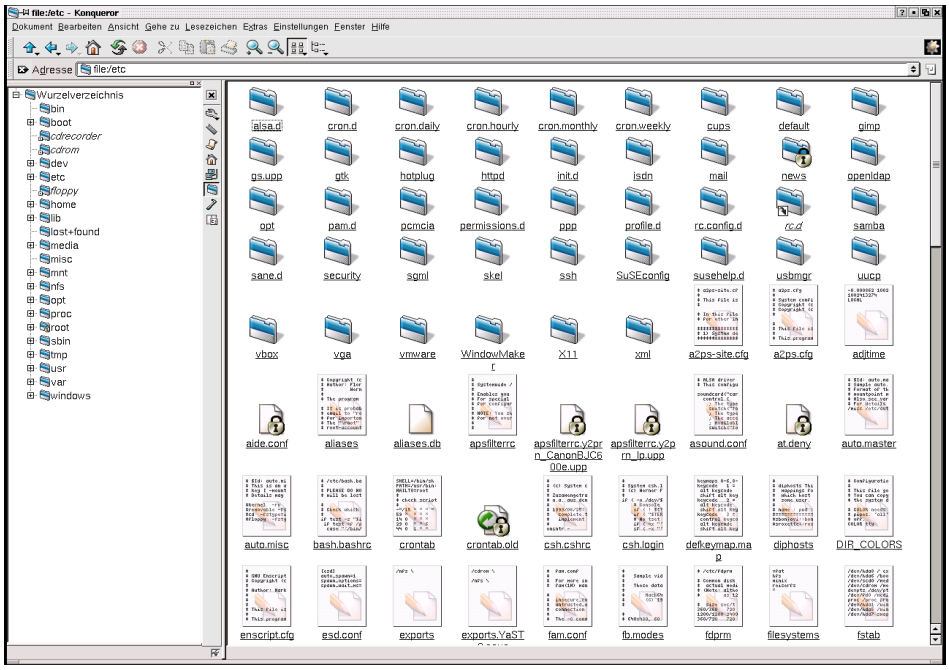
Öffnet man jetzt einen Browser und schreibt anstelle einer URL 'localhost' testen man damit, ob der Apache-Web-Server läuft.



Der Apache-Web-Server läuft, wenn der Browser die URL localhost anzeigt

3.1.8 Benutzung des Konquerors als Dateimanager

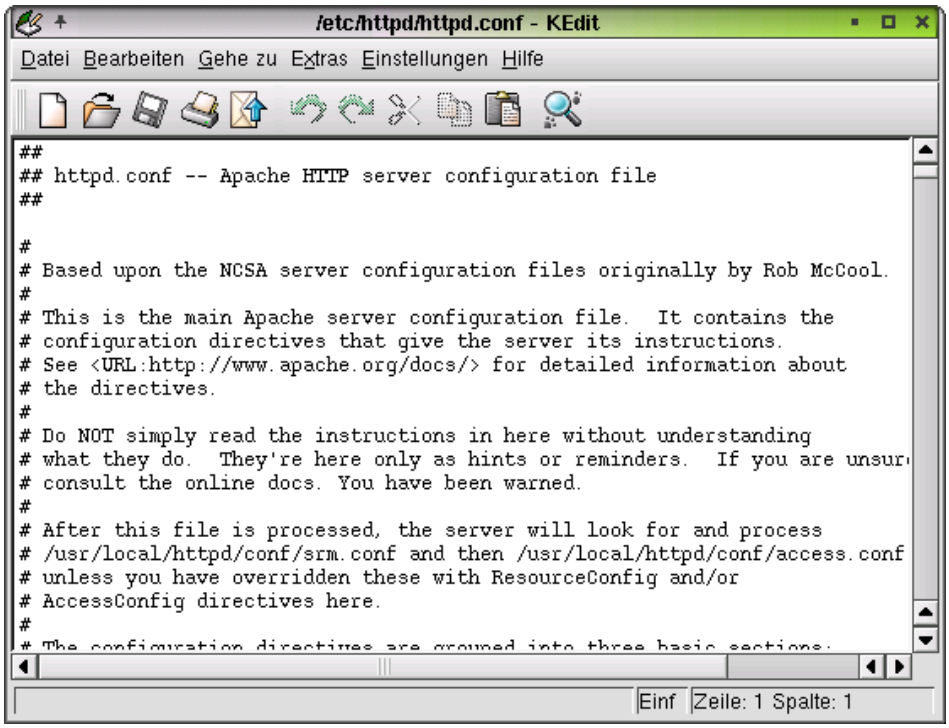
Der Konqueror ist vergleichbar mit dem Windows Explorer. Jeder WindowsUser kennt ihn und weiß nun was damit gemeint ist. Mit dem Konqueror der Version 2.1.1, der mit der **SuSE Version 7.3** geliefert wird, kann man in Dateiverzeichnissen arbeiten und auch im Internet surfen oder ihn als FTP-Filemanager einsetzen.



Der Konqueror als Dateimanager

3.1.9 Benutzung eines Texteditors zum Bearbeiten der Konfigurationsdateien

Zum Bearbeiten der Konfigurationsdateien braucht man einen Texteditor. In der **bash** würde man mit **vi**, **joe** oder **pico** arbeiten. Wir benutzen das grafische Pendant zu diesen bekannten Editoren – **kedit**, der das selbe vermag wie andere Editoren auch. Unter Linux gibt es immer einige Auswahlmöglichkeiten für kleine Programme wie Editoren.

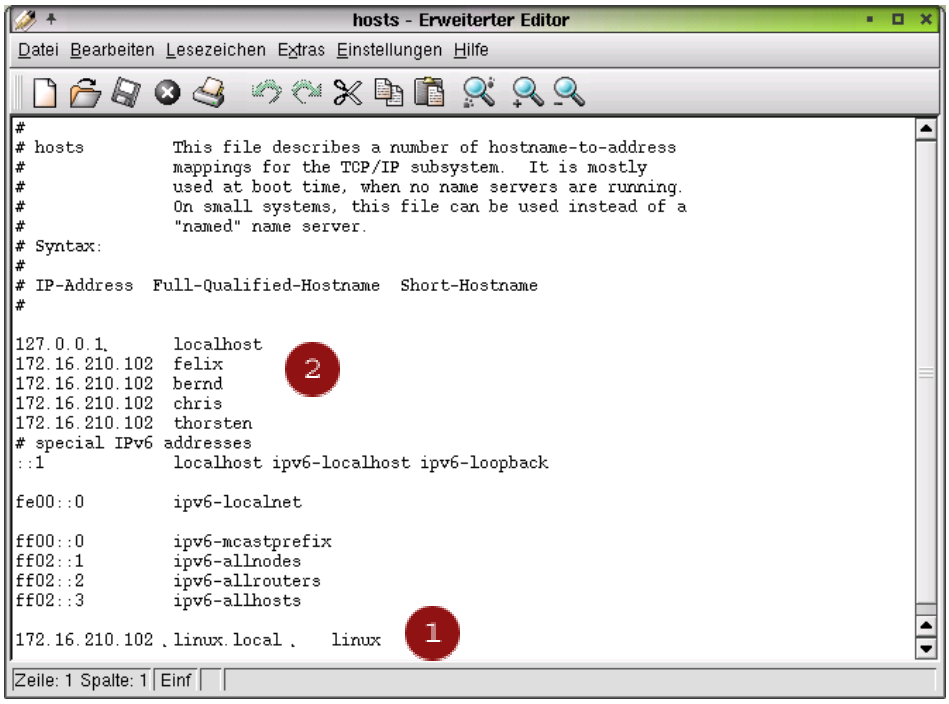


Ein Texteditor zeigt eine Konfigurationsdatei an

3.1.10 Die Datei hosts im Verzeichnis /etc

Diese Datei hat die Funktion IP-Adressen mit Computernamen in Verbindung zu bringen. Das erleichtert die Eingabe bestimmter Befehle erheblich, z.B. muss man nicht mehr **`ping 172.16.210.102`** tippen (siehe Grafik unten, Punkt 2), sondern nur noch **`ping felix`** . Das erleichtert das logische Ansprechen von IP-Adressen, weil der Mensch nun einmal mit Namen besser umgehen kann, als mit Nummern.

Die Datei könnte so aussehen:



Die Datei hosts zeigt IP-Adressen und die Computernamen

Man sieht, dass in der ersten Spalte IP-Adressen eingetragen werden. In der zweiten Spalte steht der Name des Computers und die Domain, die er angehört (durch Punkt getrennt). In der dritten Spalte steht nur der Computername.

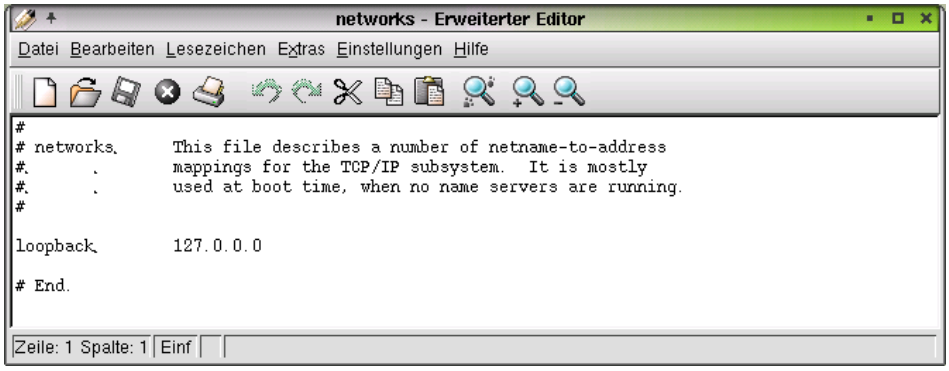
Bsp.: `172.16.210.102 linux.local linux` (*siehe Punkt 1*)

Es geht auch ohne die zweite Spalte, wie man in *Punkt 2* sehen kann. Eine Angabe des Computernamens mit der Domäne ist nicht unbedingt nötig.

3.1.11 Die Datei networks im Verzeichnis /etc

Diese Datei hat oder hatte eigentlich die Funktion Computernamen in IP-Adressen aufzulösen. Doch das scheint hinfällig zu sein, denn die Datei hosts im Verzeichnis /etc übernimmt diese Aufgabe auch. Allem Anschein nach nutzt das System diese Datei nicht mehr.

Die Datei könnte so aussehen:



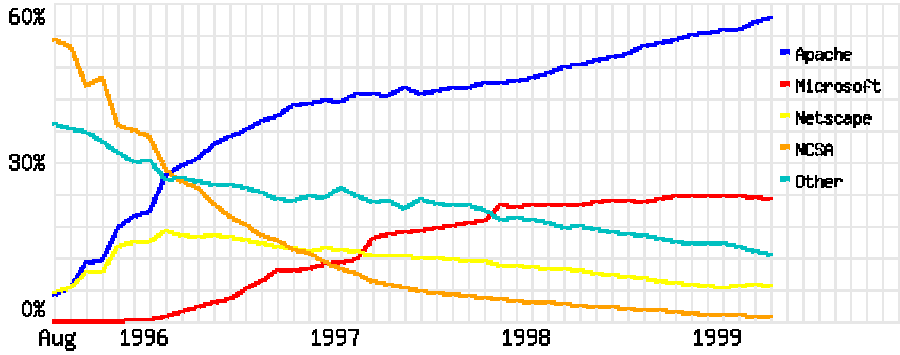
Die Datei verwandelt Computernamen in IP-Adressen

Man sieht, dass in der ersten Spalte der Name des Computers eingetragen wird. Mit einem Tab Abstand steht in der zweiten Spalte die zugehörige IP-Adresse. Eine dritte Spalte existiert hier nicht.

3.2 Apache

3.2.1 Apache

Wenn man sich mit dem Thema Web-Server auseinandersetzt, stößt man früher oder später auf den Namen Apache. Hinter diesem Namen verbirgt sich der meistgenutzte Web-Server. Mit einer Verbreitung von 56% liegt er weit vor den Server-Produkten der Konkurrenten Microsoft mit 23% und Netscape mit 7%.



Verbreitung von Apache im Vergleich zu vergleichbaren Servern[3]

Apache ist schnell und zuverlässig, aber vor allem preiswert, da es sich um Open-Source handelt. Dies bedeutet, dass jeder Benutzer von Apache Zugriff auf den Quellcode hat (GNU Public License). Wenn irgendein Fehler auftritt kann man ihn beheben und den Code an die Apache-Gruppe schicken, die diesen dann einbinden kann.

3.2.2 Der Ursprung

Apache basiert in vielen Zügen auf dem ersten Web-Server von NCSA. Apache ist daher als Ersatz zu NCSA/1.3 entstanden und in der Konfiguration zu NCSA/1.3 kompatibel. Nach dem Versuch von einer Gruppe interessierter Programmierer, einen schnellen und einfachen Web-Server zu schaffen, verdrängte Apache schnell den NCSA-Server von seinem bis dahin unangefochtenem Platz. Neben einer besseren Performance, der Behebung einiger Fehler und besserer Entsprechung des HTTP-Standards, bietet Apache einige nützliche, und bei anderen Servern oft vermisste Features.

[3] vgl. Netcraft-Survey: *http://www.netcraft.co.uk/survey/*

3.2.3 Entwicklung des Apache

Der Apache war im Dezember 1996 der weltweit am häufigsten eingesetzte Web-Server. Seine Popularität verdankt der Apache neben vielen erweiterten Funktionen seiner kostenlosen Verfügbarkeit inklusive Quelltexten. Anfang 1996 waren noch die Produkte von CERN und der NCSA die beliebtesten Server. Hier schob sich der NCSA-Server immer weiter in den Vordergrund, weil er schneller arbeitete und neue Methoden, wie z.B. „Server Includes", unterstützte. Nach dem Erscheinen der Harvest-Tools, die unter anderem einen „Object Cache" enthielten, der dem CERN-Cache weit überlegen war, trat der CERN-Server weiter in den Hintergrund.

Aber auch der NCSA-Server wies noch einige Verbesserungsmöglichkeiten auf. Dieser Probleme nahm sich die „Apache Group" an. Im Laufe der Zeit wurden so viele Patches geschrieben, dass schon fast ein neuer Server entstanden war. Schließlich wurde der Schritt zur Entwicklung eines neuen Web-Servers vollzogen. Von seinen Programmierern liebevoll „A Patchy Server“ genannt, wurde der Server den Anwendern unter dem Namen Apache zur Verfügung gestellt.

3.2.4 Leistungsmerkmale von Apache

- Zum Schutz von Dokumenten vor unberechtigtem Zugriff können Passwort-Dateien auf verschiedene Arten indiziert werden. Die Suche nach einem Eintrag wird so wesentlich effizienter, als bei einfachen Textdateien.
- Auf Fehler bei Client-Anfragen kann flexibel reagiert werden. Es können CGI-Skripte eingebunden werden, die den Fehler analysieren. Den anfragenden Client und auch dem Verwalter des Servers wird eine aufschlussreichere Meldung gegeben.
- Es gibt keine Einschränkung für Aliase und Weiterleitungen. Die Direktiven Alias und Redirect können in beliebiger Anzahl und Tiefe verwendet werden.
- Ein Abgleich zwischen Client- und Server-Fähigkeiten in Bezug auf Qualität und Art der Dokumente und HTTP-Version („Content Negotiation") ist möglich.
- Mit Apache können mehrere „virtuelle" Server realisiert werden. Das heißt Anfragen, die an unterschiedliche IP-Adressen eines Rechners gestellt werden, können unterschieden werden. Je nach IP-Adresse kann dann entsprechend reagiert werden.
- Ein weiterer Punkt, der für Apache spricht, ist das Modulkonzept und das gut dokumentierte API (Application Programmers Interface). Dadurch sind Erweiterungen leicht zu realisieren und neue Features leicht in den Server einzubinden. So existiert zum Beispiel eine Implementierung des von Netscape entwickelten SSL (Secure Socket Layer) für Apache, das eine sichere Übertragung zwischen Server und Client ermöglicht.

3.2.5 Eigenschaften des Apache

Der Apache verfügt über drei besonders hervorzuhebende Eigenschaften.

1. Zum einen unterstützt er sowohl IP-basiertes als auch nicht-IP-basiertes virtuelles Hosting.
2. Weiterhin ist er in der Lage, mit UNIX-Datenbankdateien zu arbeiten und auf Routinen des GNU Datenbank Managers zurückzugreifen. Dadurch wird eine höhere Geschwindigkeit beim Durchsuchen von Autorisierungsdateien erreicht.
3. Die dritte besondere Eigenschaft des Apache ist der modulare Aufbau des Servers. Alle erweiterten Funktionen sind in speziellen Modulen enthalten. Der Server selbst steht im Modul „main". Der Anwender hat nun die Möglichkeit, sich den Server nach eigenen Anforderungen aus den verschiedenen Modulen zusammenzusetzen, indem die nicht benötigten Module in der Datei **`/etc/httpd.conf`** auskommentiert werden. Die Dokumentation zum Apache gibt einen genauen Überblick, welche Funktionen jedes einzelne Modul erfüllt.

Der modulare Aufbau bringt mehrere Vorteile mit sich. Indem nur die wirklich benötigten Module in den Server kompiliert werden, bleibt der Apache kleiner als eine Standardversion und belegt weniger Speicherplatz. Weniger Module bedeuten gleichzeitig weniger Angriffsfläche, was die Sicherheit des Systems erhöht. Wird beispielsweise das Modul „mod_userdir" nicht verwendet, ist der Server nicht mehr in der Lage, das Konstrukt „~username" auf das Heimatverzeichnis des Nutzers umzusetzen. Somit sind die Heimatverzeichnisse der Nutzer vor Zugriffen über den Web-Server sicher geschützt.

Über eine API besteht darüber hinaus die Möglichkeit, selbst programmierte Module in den Server zu kompilieren und den Apache noch besser auf die eigenen Anforderungen abzustimmen. Der dafür notwendige Aufbau eines Moduls sowie die Schnittstelle sind ebenfalls in der Dokumentation definiert.

3.2.6 Die httpd.conf

Bei der httpd.conf handelt es sich um eine Konfigurationsdatei, d.h. in der httpd.conf werden verschiedene Eigenschaften der Verhaltensweisen eines Servers beschrieben und festgehalten. Dies geschieht durch einzelne Befehlszeilen oder innerhalb sog. Container (zum Bsp.: `<VirtualHost>`), innerhalb denen wiederum einzelnen Befehlssätze sog. Befehlsblöcke untergeordnet sind. Da die einzelnen Befehlszeilen oder Befehlsblöcke in einer bestimmten Reihenfolge stehen müssen, spricht man bei der httpd.conf auch von einer hierarchisch aufgebauten Datei.

Ursprünglich benötigte man drei verschiedene Dateien um einen Server zu konfigurieren: Die httpd.conf als Hauptkonfigurationsdatei, die srm.conf als Ressourcen-Konfigurationsdatei und die access.conf als Konfigurationsdatei für die Zugangsberechtigungen.
Heute benötigt man nur noch die httpd.conf da alle Konfigurations-Direktiven in dieser Datei abgehandelt werden.[4]

[4] vgl. *http://selfaktuell.teamone.de/artikel/server/apacheconf/apconf08.htm*

3.2.7 Die drei wichtigsten Funktionsabschnitte der httpd.conf

Global Environment

Unter "Global Environment" oder „Globale Umgebung“ werden die allgemeinen Bedingungen verstanden, unter denen der Apache-Web-Server später arbeiten soll. Zu der „Global Environment“ gehören folgende Punkte:

- Angaben zum Servertyp
- die Definition des Verzeichnisses bzw. Laufwerks auf dem Rechner, in dem sich die Programm-, Konfigurations- und Protokolldateien befinden,
- ein paar Festlegungen für Zugriffszeiten und -zahlen
- die Liste der zur Laufzeit einzubindenden Module

„Main server configuration“

Unter „Main Server Configuration“ oder „Generelle Serverkonfiguration“ versteht man das Festlegen des Verhaltens oder die Arbeitsweise des Servers. Zu der „Main Server-Configuration“ gehören folgende Punkte:

- Festlegung des vom Server beanspruchten Ports (für das http-Protokoll in der Regel Port 80).
- Servername und Administrator-Adresse
- Pfadangaben zu Dokument-, Benutzer- und Skriptverzeichnissen
- Formatierung der Protokolldateien
- Aufgabenzuweisung an die geladenen Module

„Virtual Hosts“

Unter Verwendung der „Virtual Hosts“ oder „virtuellen Hosts“ lassen sich beliebig viele Web-Seiten simulieren. Will man mehrere Webseiten gleichzeitig verwalten, braucht man dafür weder einen gut ausgerüsteten Rechner oder Server, noch sind die Kosten dafür hoch. Normalerweise genügt ein gewöhnlicher Rechner um mehrere Webseiten gleichzeitig zu verwalten.

Für den Betrieb mehrere Webseiten werden virtuelle Hosts eingerichtet. Jeder Virtuelle Host kann zwar über die gleiche IP-Adresse verfügen, verweist jedoch auf verschiedene URLs. Durch die Einrichtung virtueller Hosts werden die Konfigurations- und Wartungszeiten sehr gering gehalten.

Wie die Bezeichnung "virtuell" schon besagt, handelt es sich um etwas vorgetäuschtes, imaginäres. Die Bezeichnung „Host" steht hier für einen Rechner. Virtueller Host = vorgetäuschter Rechner. Durch das „virtual hosting" wird vorgetäuscht, dass jede Webseite auf einem eigenem Rechner liegt. Das „virtual hosting“ wird in der httpd.conf innerhalb der VirtualHost-Container vorgenommen. [5]

[5] vgl.*http://selfaktuell.teamone.de/artikel/server/apacheconf/apconf08.htm*

3.2.8 IP-basierte virtuelle Hosts

Grundsätzlich kann auf einem System auf jeder IP-Adresse ein Server ausgeführt werden. Dieser Vorgang findet dann jedoch nur auf einem einzelnen Server statt. Für den Computer ist dies eine geringere Belastung, da alle Anforderungen von nur einem Prozess behandelt werden. Man realisiert dies mit IP-basiertem virtuellen Hosting. Bei dieser Methode spricht man von IP-Mulitplexing oder von Aliasing.

In der httpd.conf sieht der VirtualHost-Container für IP-basierte virtuelle Hosts wie folgt aus:

Syntax: NameVirtuelHost **Adresse**
Kontext: Serverkonfiguration

Beispiel:

```
<VirtualHost 172.16.210.102>
ServerName project_apache
ServerAdmin project@apache
DocumentRoot /www/docs/project_apache
</VirtualHost>

<VirtualHost 172.16.210.103>
ServerName project_gruppe
ServerAdmin project@gruppe
DocumentRoot /www/docs/project_gruppe
</VirtualHost>
```
[6]

Man kann entweder pro Adresse eine zusätzliche Netzwerkkarte einbauen oder, und das ist die kostengünstigere und elegantere Möglichkeit, einer Karte mehrerer Adressen zuordnen. Hierzu muss man der physikalischen Ethernet-Schnittstelle **eth0** des Servers je nach Anzahl der verschiedenen IP-Adressen virtuelle Ethernet-Schnittstellen zuordnen. Dabei ist zu beachten, dass die IPs dem selben Subnetz des Netzwerks gehören.

Um eine solche Zuordnung zu erreichen gibt man folgenden Befehl über die Konsole ein:

Beispiel:

```
ifconfig eth0:0 172.16.210.204
ifconfig eth0:1 172.16.210.205
```

ifconfig: dieser Befehl zeigt physikalische und virtuelle Netzwerk-Schnittstellen **eth0:0/1**: hier wird der real existierenden Ethernet Schnittstelle 0 die virtuellen Ethernet-Schnittstellen 0 und 1 zugeordnet über die die IP-Adresse 172.16.210.205 angefordert werden kann.[7]

[6] vgl. Apache, Dr. Susanne Wigard, bhv-Verlag, Landsberg, 2000

[7] vgl. Apache und CGI – Administration und Programmierung, Rich Bowen und Ken Coar, Markt und Technik Verlag, München, 2000

Beispiel:

```
Konsole
Datei  Sitzungen  Einstellungen  Hilfe
linux:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:E0:7D:92:20:09
          inet addr:172.16.210.102  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::2e0:7dff:fe92:2009/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:606 errors:0 dropped:0 overruns:0 frame:0
          TX packets:149 errors:0 dropped:0 overruns:0 carrier:0
          collisions:11 txqueuelen:100
          RX bytes:161643 (157.8 Kb)  TX bytes:17151 (16.7 Kb)
          Interrupt:10 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3912 (3.8 Kb)  TX bytes:3912 (3.8 Kb)

linux:~ #
Neu   Konsole
```

```
linux:~ # ifconfig eth0:0 172.16.210.204
linux:~ # ifconfig eth0:1 172.16.210.205
linux:~ # ifconfig
```

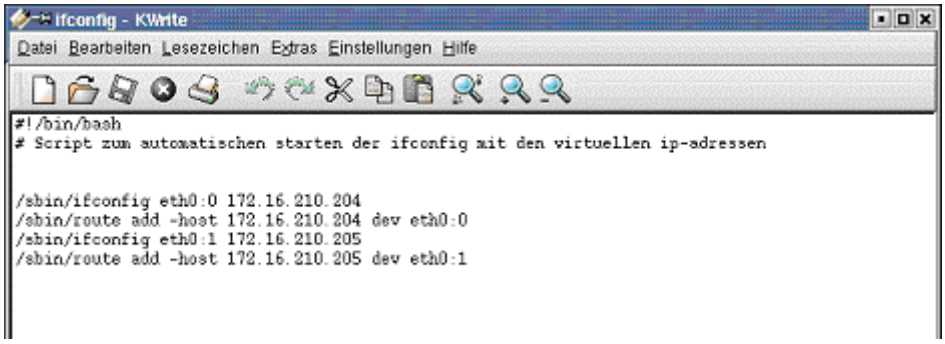
```
Konsole
Datei  Sitzungen  Einstellungen  Hilfe
linux:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:E0:7D:92:20:09
          inet addr:172.16.210.102  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::2e0:7dff:fe92:2009/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:141 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:22911 (22.3 Kb)  TX bytes:1178 (1.1 Kb)
          Interrupt:10 Base address:0x6000

eth0:0    Link encap:Ethernet  HWaddr 00:E0:7D:92:20:09
          inet addr:172.16.210.204  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:10 Base address:0x6000

eth0:1    Link encap:Ethernet  HWaddr 00:E0:7D:92:20:09
          inet addr:172.16.210.205  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:10 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3260 (3.1 Kb)  TX bytes:3260 (3.1 Kb)

linux:~ #
Neu   Konsole
```

Die Zuordnung mehrer IP-Adressen zu der Netzwerk-Schnittstelle, um zusätzliche IP-basierte virtuelle Hosts zu unterstützen, hat den Nachteil, dass jeder IP-basierter virtueller Host eine IP-Adresse verbraucht von denen für gewöhnlich nur eine begrenzte Anzahl zur Verfügung steht.[8]

[8] vgl. Linux – Apache Web Server Administration, Charles Aulds, Sybex Verlag, Düsseldorf, 2001

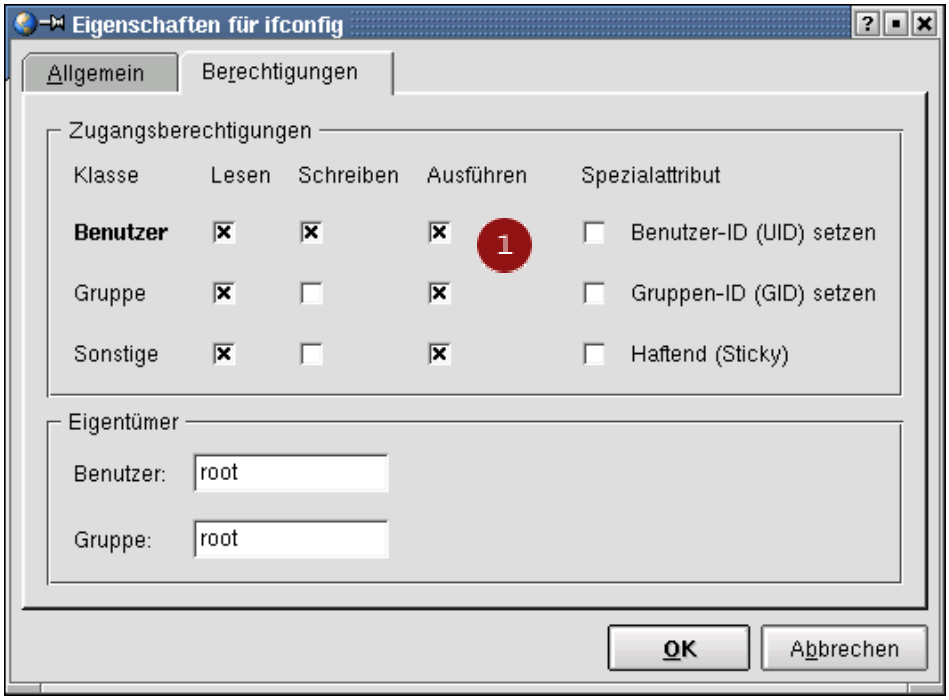
3.2.9 Zusammenhang zwischen IP-basierten virtuellen Hosts und boot.local

Damit nicht nach jedem Neustart des Servers die virtuellen Netzwerkanschlüsse erneut konfiguriert werden müssen, bietet es sich an, eine bash-Datei, die die Zuweisungen von IP-Adressen auf Netzwerkanschlüsse beinhaltet, zu schreiben. In die `boot.local` muss dann eine entsprechende Befehlszeile, in der auch die Pfadangabe zu der bash-Datei steht, geschrieben werden, sodass bei jedem Neustart die bash-Datei ausgeführt wird.



Die bash-Datei

Steht `#!/bin/bash` am Anfang einer Datei, bedeutet das, dass diese Datei in einer Shell ausgeführt werden kann. Sie ist vergleichbar mit einer .exe-Datei in Windows. Durch `ifconfig eth0:0 172.16.210.204` und `ifconfig eth0:1 172.16.210.205` werden der Netzwerkkarte weitere, sozusagen virtuelle, IP-Adressen zugeordnet.[9]



Eigenschaften der bash-Datei

Unter den Eigenschaften der bash-Datei muss die Berechtigung zum Ausführen der Datei für „root“ angewählt werden (*Punkt 1*), andernfalls könnte die Datei beim Booten des Systems nicht ausgeführt werden.

[9] vgl. Linux – Apache Web Server Administration, Charles Aulds, Sybex Verlag, Düsseldorf, 2001

Da nicht jeder im grafischen Modus arbeitet, wollen wir hier auch die Möglichkeit der Änderung der Rechte einer Datei im nicht grafischen Modus von Linux zeigen.

Wie gezeigt werden die Rechte der Datei `ifconfig` im Verzeichnis `/www/ifconfig/autostart/` verändert. Lassen wir uns auf der Konsole die Rechte der Datei anzeigen, sehen wir diese Ausgabe:

```
linux:/www/ifconfig/autostart # ls -o ifconfig
-rw-r--r--    1 root          262 Jan 17 09:36 ifconfig
linux:/www/ifconfig/autostart #
```

Der Befehl `ls -o ifconfig` zeigt uns unter anderem die Rechte der Datei:

`r w - r - - r - - [...] ifconfig`

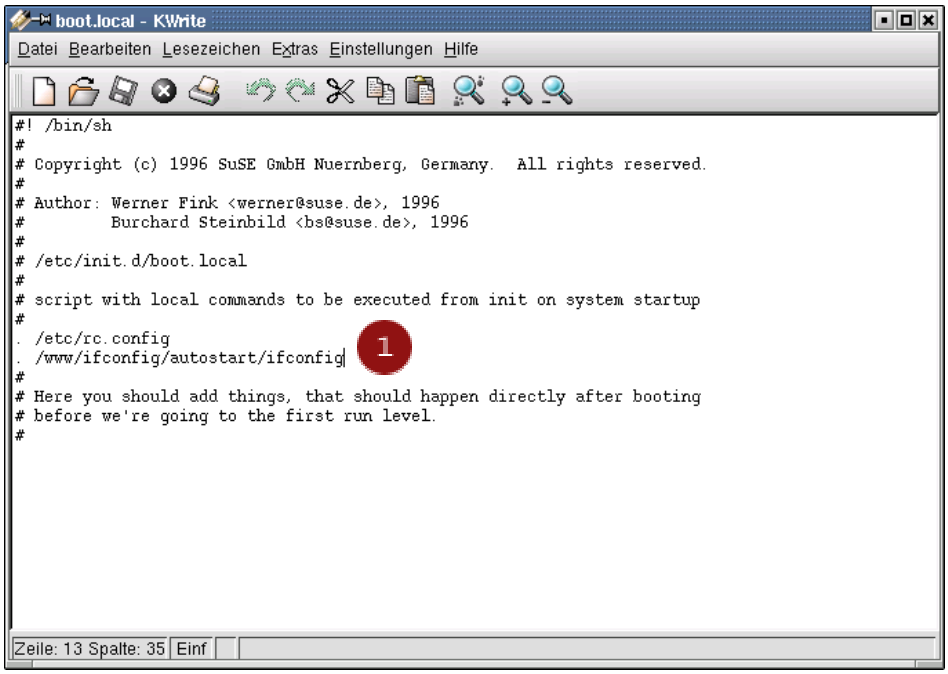
Das bedeutet: Die Datei hat Standardrechte. `r w -` heißt, dass der Besitzer der Datei auf die Datei lesend und schreibend zugreifen kann. `r - -` an zweiter Stelle bedeutet Leserechte für die Gruppe, die der Besitzer angehört und das nächste `r - -` steht für das Leserecht für alle anderen Nutzer.

Zum Ändern dieser Rechte reicht diese Eingabe:

```
linux:/www/ifconfig/autostart # chmod 755 ifconfig
linux:/www/ifconfig/autostart # ls -o ifconfig
-rwxr-xr-x    1 root          262 Jan 17 09:36 ifconfig
linux:/www/ifconfig/autostart #
```

Der Befehl `chmod 755 ifconfig` ist gleichbedeutend zu `chmod u+x,g+x,o+x ifconfig`

Beide Befehle bewirken dasselbe. Jeder Rechte-Gruppe wird ein x hinzugefügt, welches soviel heißt wie „execute“, also ausführbar.



Die Datei boot.local

Der Pfad `./www/ifconfig/autostart/ifconfig` (*Punkt 1*) zeigt dem Computer welche Datei er bei jedem Start ausführen soll.

3.2.10 Name-basierte virtuelle Hosts

Namens-basierte virtuelle Hosts werden fast auf die gleiche Weise wie IP-basierte virtuelle Hosts konfiguriert, mit dem wichtigen Unterschied, dass die Direktive *NameVirtualHost* den Server darüber informiert, auf welcher Adresse Anforderungen für virtuelle Hosts empfangen werden. Die VirtualHost-Abschnitte sehen genau so aus wie bei der Verwendung von IP-basierten virtuellen Hosts, nur dass alle auf dieselbe IP-Adresse verweisen. Des weiteren wird **`ifconfig`** nicht berührt. Die Konfiguration von Namen-basierten virtuellen Hosts ist also die einfachere Methode da hierzu nur in der Hosts-Datei verändert werden muss[10]

Syntax: NameVirtuelHost Adresse/IP
Kontext: Serverkonfiguration

Beispiel:

```
NameVirtualHost 172.16.210.102
```

(Mit dieser NameVirtualHost-Direktive können nun unterschiedliche Einstellungen an den unterschiedlichen virtuellen Server vorgenommen werden).

```
<VirtualHost 172.16.210.102>
ServerName project_apache
ServerAdmin project@apache
DocumentRoot /www/docs/project_apache
</VirtualHost>
```

```
<VirtualHost 172.16.210.102>
ServerName project_gruppe
ServerAdmin project@gruppe
DocumentRoot /www/docs/project_gruppe
</VirtualHost>
```
[11]

3.2.11 Beispiel und Erläuterung eines virtuellen Hosts

In der Datei httpd.conf wird unter der „Section 3“, in der sich die VirtualHost-Container befinden, folgendes eingetragen:

1. <VirtualHost 172.16.210.102>
2. ServerName project_apache
3. ServerAdmin project@apache.de
4. DocumentRoot /www/docs/project_apache
5. </VirtualHost>

[10,11] vgl. .Apache und CGI, Rich Bowen und Ken Coar, Markt + Technik Verlag

1. <VirtualHost 172.16.210.102>

Syntax: NameVirtualHost *addr/:port*
Kontext Serverkonfiguration

Die VirtualHost-Container beginnen jeweils mit einer IP-Adresse. Hieraus wird jedoch nicht ersichtlich, ob es sich um einen Name- oder IP-basierten virtuellen Host handelt. Sollte sich jedoch nach der IP-Adresse eine durch einen Doppelpunkt getrennte Zahl befinden, weist dies auf eine Port-Nummer hin. Die Port-Nummer ist eine Zusatzinformation, die den gewünschten Internetdienst kennzeichnet. FTP verwendet normalerweise die Ports 20 und 21 und HTTP den Port 80.

Sie gehören zu den Ports der öffentlichen Anwendungen und werden durch die Nummern **unter 255** gekennzeichnet. Die Nummern von **255 bis 1023** sind für marktfähige Anwendungen reserviert. Daneben gibt es Port-Nummern ohne feste Zuordnung, die mit Nummern **über 1023** beziffert und vom System meist dynamisch zugewiesen werden. Die maximale Anzahl der Port-Nummern beträgt **65536**.

Für jede gleiche IP-Adresse innerhalb der **`/etc/httpd.conf`** wird eine bestimmte Port-Nummer verwendet. So unterscheidet der Apache-Web-Server die virtuellen Hosts. Dieser hat immer die gleiche Adresse jedoch nach dem Doppelpunkt immer unterschiedliche Portnummern.

Beispiel:

Adresse [**:**] **Portnummer**

```
172.16.210.102:1025
172.16.210.102:8371
172.16.210.102:5339
```

2. ServerName

Syntax: ServerName *domain-name*

Kontext: Serverkonfiguration, virtueller Host

Sollte der Server einen registrierten DNS-Namen haben, wird der Name des Servers an dieser Stelle angegeben. Der DNS-Name ist ein Domain-Name, zu dem der DNS-Dienst die zugehörige IP-Adresse ermitteln kann.

Beispiel: `ServerName project_apache`

Sollte jedoch im internen Netz kein DNS-Dienst verfügbar sein, wird einfach die IP-Adresse des Servers verwendet

Beispiel: `ServerName 172.16.210.102`

Es kann jedoch auch die Datei **/etc/hosts**[12] verwendet werden, um die Zuordnung von Servernamen zu IP-Adressen vorzunehmen.

Beispiel:

IP-Adresse	**Servername**
`172.16.210.102`	`project_apache`[13]

[12] siehe *3.1.10 Die Datei hosts im Verzeichnis /etc*

[13] vgl. Apache, Dr. Susanne Wigard, bhv-Verlag, Landsberg, 2000

3. ServerAdmin

Syntax: ServerAdmin *e-mail-adresse*

Kontext: Serverkonfiguration, virtueller Host

Damit die Anwender den Server-Administrator bei auftretenden Problemen benachrichtigen können, sollte man dessen E-Mail-Adresse in der Direktive ServerAdmin angeben. Diese Adresse wird bei jedem Fehler zusammen mit der Fehlermeldung an den Client gesendet.

Beispiel: `ServerAdmin project@apache.de`

4. DocumentRoot

Syntax: DocumentRoot *Verzeichnisname*
Kontext: Serverkonfiguration, virtueller Host

Es wird eine Verzeichnisstruktur für den Server unter **`/www/docs/project_apache`**[14] angelegt. Hier werden alle Dokumente abgelegt, die beim Aufruf des Servers im World Wide Web gezeigt werden sollen.

Die Syntax DocumentRoot ist gleichzeitig eine Pfadangabe auf welches Verzeichnis der betreffende Server zugreifen soll.

5. </VirtualHost>

Als letztes wird der VirtualHost-Container mit dem Zeichen [**/**]abgeschlossen [15]

[14] dies ist nur ein Beispiel, es kann ein beliebiges Verzeichnis erstellt werden

[15] vgl. Apache, Dr. Susanne Wigard, bhv-Verlag, Landsberg, 2000

3.3 Common Gateway Interface - CGI

3.3.1 Beschreibung von CGI

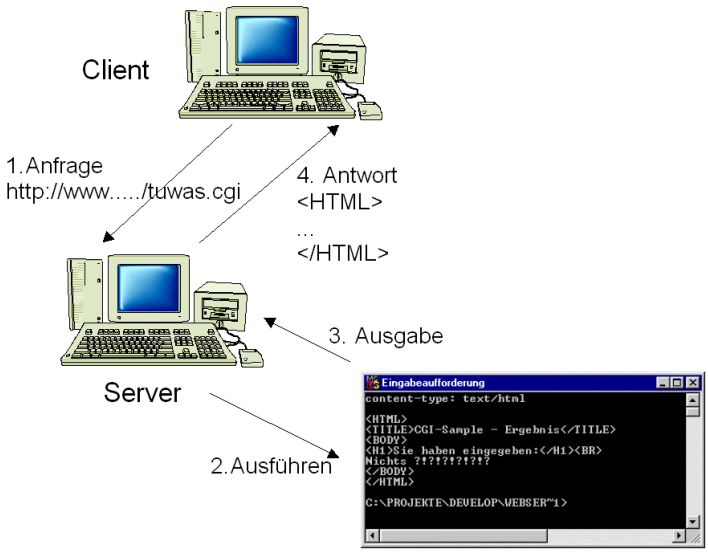
CGI (Common Gateway Interface) ist eine Schnittstelle für die Ausführung von Programmen auf dem Web-Server, die in verschiedenen Programmiersprachen wie z.B. C, C++, Java oder auch in speziellen Skriptsprachen wie Perl oder PHP geschrieben sein können.

Häufig handelt es sich bei CGI um sogenannte Shellskripte, die von der Befehlsausführungs-Shell, dem sogenannten Kommando-Interpreter des Betriebssystems ausgeführt werden. Solche CGI-Skripte erledigen beispielsweise die Auswertung von HTML-Formularen. Die CGI-Schnittstellenspezifikation regelt die Einzelheiten, wie der Server Informationen vom Client an das Skript übergibt und wie die Ausgaben des Skripts an den Server und von dort aus an den Client zurückgehen.

Die CGI-Programme schreiben eine Angabe des Inhalttyps (Content Type) sowie den eigentlichen Inhalt – beispielsweise HTML-Code – auf die Standardausgabe, in der Regel der Bildschirm. Dieser vom Skript erzeugte HTML-Quelltext wird vom Webserver mit den nötigen HTTP-Headern versehen, an den Browser weitergeleitet und dort wie ein normales HTML-Dokument angezeigt.

3.3.2 Die Kommunikation über ein CGI

1. Der Browser sendet eine Anfrage an den Web-Server. Diese enthält eine URL, die auf ein CGI-Programm verweist. Optional können noch Daten eines HTML-Formulars mitgesendet werden.



2. Der Apache-Web-Server führt das CGI-Programm aus.

3.+4. Die Ausgabe des Programms wird an den Client zurückgeschickt und von diesem normalerweise als HTML-Seite interpretiert und dargestellt

3.3.3 Die Rechte eines CGI-Prozess

Die Rechte des CGI Prozesses hängen von der Konfiguration des Webservers ab. Einige Provider lassen den Apache als **Root**[16] laufen, denn das bringt gewisse Vorteile in der Administration. Mit Root-Rechten darf der Apache-Web-Server beispielsweise in jedem Homepageverzeichnis die Logfiles schreiben, was andererseits einen erhöhten Verwaltungsaufwand erfordern würde, wenn das nicht so wäre.

Der Einwand besteht darin, dass jeder CGI-Prozess, welcher als Kindprozess des HTTP-Daemons Apache gestartet wird, **als Root** läuft und das gewisse Sicherheitsprobleme aufwirft. Um das zu verhindern, lässt man ein CGI unter den Rechten **group** oder **other** laufen.

[16] Root ist der Superuser unter Linux, quasi der Administrator

3.4 PHP[17]

3.4.1 Was ist PHP?

Wenn Sie beim Besuch einer Webseite wie z. B. einer Suchmaschine oder einer Produktdatenbank eine Abfrage ausführen, dann werden Sie sicher hin und wieder einen Blick auf die Status- oder Adresszeile werfen. Dort ist in der Regel die aktuelle URL bzw. die aufzurufende URL sichtbar. Am Ende dieser Adresse taucht oft die Dateiendung php3 auf - hier haben Sie es dann mit einem so genannten PHP-Skript zu tun. Genau genommen hat in erster Linie der Server damit zu tun, und was es damit auf sich hat, das wird das Thema dieses Abschnittes sein.

PHP ist eine Skriptsprache, deren Befehle in dem HTML/XHTML-Code einer Webseite eingebettet sind. Diese Technik an sich ist nicht gerade neu - PHP stand von Anfang an im Schatten der allgegenwärtigen Programmiersprache Perl. Dies war 1995 - inzwischen ist PHP in der Version 4 verfügbar und hat sich einen festen Platz im World Wide Web erobert. Das Geheimnis seines Erfolges liegt unter anderem in der einfachen Erlernbarkeit, seiner hohen Leistungsfähigkeit und vor allem in der einfachen Zugriffsmöglichkeit auf verschiedene Datenbankmodelle. Da inzwischen gerade der zuletzt genannte Punkt ein wesentliches Kriterium bei der Auswahl der Software für ein modernes Webprojekt darstellt, erfreut sich PHP einer rasant steigenden Beliebtheit.

3.4.2 Zur Geschichte von PHP

Im Jahre 1994 arbeitete Rasmus Lerdorf an einer neuen Version einer Skriptsprache. Anfangs nutzte er die ersten Versionen, um die Seitenzugriffe auf seiner privaten Homepage festzustellen. 1995 veröffentlichte er die erste Version von PHP unter dem Namen „Personal Homepage Tools“. Bald darauf wurde diese erste Version überarbeitet und offiziell unter dem Namen PHP verfügbar gemacht. Anfangs bestand die neue Skriptsprache lediglich aus einigen Makros[18].

Wie bei allen erfolgreichen Open-Source-Projekten stellte sich sehr bald eine Nachfrage nach weiteren Funktionen ein. Auf Grund dieser Nachfrage erschien im Jahre 1996 die PHP-Version 2.0 mit weiteren programmiersprachenähnlichen Funktionen, wie zum Beispiel einem Parser[19]. Zu diesem Zeitpunkt konnten bereits HTML-Daten verarbeitet und Datenbankoperationen durchgeführt werden, weil der verbesserte Parser die gleichen Konstrukte unterstützte wie alle modernen modularen Programmiersprachen.

Im Oktober 1997 erschien die erste Alpha-Version von PHP 3.0, die mit einem anderen, neuen Interpreter von Zeev Suraski und Andi Gutsman ausgestattet war und zu dem auch Shane Caraveo, Stig Bakken, Jim Winstead und viele andere mehrere wichtige Beiträge geleistet haben. Die offizielle Version PHP 3.0 erschien im Juni 1998.

---

[17] vgl.: Webserver verstehen und tunen von Kristian Köhntopp

[18] Kleine Skripte, die Anweisungen enthalten

[19] Ein Parser wandelt den PHP-Quellcode in Bytecode um

Inzwischen ist die Version4.x erhältlich. Auch diese wurde mit einem vollständig neuen Parser ausgestattet. Eine nützliche Eigenschaft der Version 4.0 ist die Möglichkeit, binären Code zu interpretieren. PHP 4.0 ermöglicht den Web-Site-Entwicklern auf diese Weise die ausschließliche Verwendung von binärem Code anstelle von Quellecode. Die Ausführung dieses Codes verläuft erwartungsgemäß wesentlich schneller, als das wiederholte Interpretieren des gleichen PHP-Codes bei jedem Zugriff auf eine Seite.

PHP erwies sich bisher als sehr stabil und als leicht erlernbar. Dies und seine hohe Funktionalität haben dieser Skriptsprache inzwischen einen festen Platz im Web gesichert.

Die aktuellen Statistiken von Netcraft[20] belegen, dass mehr als eine Million Domänen eine PHP-Version einsetzen[21].

3.4.3 PHP im Vergleich mit anderen bekannten Webentwicklungssystemen

PHP ist in Version 3 eine reine Interpretersprache und in Version 4 ein Bytecode-Compiler, der das Script beim Aufruf kompiliert. PHP kann als CGI-Programm oder als Bestandteil des Webservers (mod_php) ausgeführt werden. Als CGI-Programm ist es beliebig portabel und als Modul ist es für eine Reihe von populären APIs verfügbar (Apache, Netscape, Microsoft und fhttpd). PHP unterstützt Datenbankzugriffe nicht nur über ODBC (oftmals Treiber von schlechter Performance und mit unvollständigen API-Implementierungen), sondern auch über die Native API. Dazu gehört LDAP, IMAP und eine Reihe anderer Datenbanken, außerdem HTTP, FTP-Protokolle (Intelligent Agent-Programmierung) und Direktzugriff auf Dbase und DB/DBM-Dateiformate. Dynamische Generierung von PNG-Bildern mit der libgd und der freetype library. Volltextindices und Suchmaschinen über externe Open Source Programme (htdig und andere).

Cold Fusion ist eine Interpreter-Markup-Language. Es handelt sich um eine reine Interpretersprache mit der Option Seitenquelltext auf der Serverseite durch "Codierung" zu verschleiern (das ausgegebene HTML muß selbstverständlich lesbar sein). Cold Fusion kann als Modul in Apache ausgeführt werden oder als CGI-Programm. Da es nicht Open Source ist, ist seine Verfügbarkeit über Plattformgrenzen beschränkt. Linux-Support ist angekündigt und im Betatest. Datenbankzugriff erfolgt über ODBC. LDAP, IMAP und einige andere Dinge werden unterstützt, andere Protokolle müssen gegebenenfalls über Dritthersteller oder Erweiterungen von Herstellerseite zugekauft werden. Intelligent Agents können erstellt werden. Keine Generierung von GIF-Bildern on-the-fly. Cold Fusion kann Texte mit einer eigenen Engine (Verity) volltextindizieren und dann darauf zugreifen. Zahlreiche Spracherweiterungen sind vorhanden (Custom Tag Library beim Hersteller). Syntax: Tagschreibweise, um bei Anfängern den Eindruck einer Programmiersprache zu vermeiden. Dieser Eindruck wird von Cold Fusion-Studio unterstützt, einer Spielversion von Homesite, die um teilweise lächerlich wirkenden Insert-tag Support für die CF-Standardtags erweitert wurde.

[20] siehe *http://www.netcraft.com/*

[21] siehe *http://www.php.ent/usage.php3*

Microsoft ASP ist keine Scriptsprache, sondern ein Framework für die Einbindung einer solchen. Es stellt mehr eine API dar, die beliebige Scriptsprachen verwenden können, um Variablen über die Lebensdauer einer Seite hinaus persistent zu machen, um mit dem Webserver und anderen Scriptsprachen kommunizieren zu können und um andere, Microsoft-spezifische Eigenheiten ansprechen zu können. Mit MS-ASP werden zwei Scriptsprachen mitgeliefert (JavaScript und VisualBasic), aber es existieren weitere, von Microsoft unabhängige Sprachen (etwa Perl), die ebenfalls ASP verwenden. ASP ist fester Bestandteil des IIS auf NT bzw. Windows 2000/XP, es gibt aber auch andere Webserver (z.B. Samba Server), die ASP unterstützen. Da es sich beim IIS mit ASP um ein Microsoft-Produkt handelt, sind Supportoptionen, -preise und -qualität sofort klar (Hinweis: Seit Windows 2000 verlangt Microsoft eine zusätzliche Internet Connection License für den IIS, wenn dieser am Internet betrieben werden soll). Unterstützung anderer Plattformen ist kaum möglich, da ASP starken Gebrauch von den OLE/COM/DCOM-Fertigkeiten der MS-Plattform macht: Selbst nach der Portierung (etwa durch Chilisoft oder Software AG) steht man immer noch vor dem Problem, die entsprechenden OLE/COM/DCOM-Objekte auf der Nicht-Microsoft Plattform verfügbar zu machen. LDAP, IMAP und andere Dinge sind über den OLE/COM/DCOM-Mechanismus möglich, entsprechende Objekte sind eventuell von Drittanbietern zu kaufen.

mod_perl und **embedPerl** sind ein Mechanismus, mit denen man den Perl-Interpreter als Bestandteil des Apache-Web-Servers ausführen kann und mit dem man dann Perl-Programme mit HTML mischen kann. Programme werden proaktiv geladen und compiliert, sodass die typische Startup-Latenz von Perl-Programmen als CGI wegfällt. Der Speicherverbrauch ist beträchtlich, die Geschwindigkeit ist größer als bei PHP Version 3 - PHP Version 4 und mod_perl liegen in etwa gleichauf. Unterstützt wird der Apache-Web-Server auf allen Plattformen. mod_perl ist prinzipiell in der Lage, alle Perl-Module auf dem CPAN auszuführen, daher werden alle von Perl unterstützten Datenbanken (einschließlich ODBC und einer Menge nativer Interfaces) unterstützt, ebenso LDAP, IMAP, LibGD. Negativ fällt hier vor allen Dingen der übermäßige Speicherverbrauch bei größeren Projekten auf und die auf Anfänger abschreckend wirkende Vielfalt von Sprachkonstrukten und Bibliotheken. Besondere Pluspunkte für erfahrene Programmierer sind die Ausdrucksstärke und Vielfalt der Sprache und ihrer Konstrukte und die wirklich umfassende Sammlung von Bibliotheken.

Next/Apple Webobjects sind ein System, mit dem ein kleines, im Source vorliegendes Rumpf-C-CGI-Programm über einen remote procedure call mit einem als Coprozess laufenden Anwendungsprogramm kommuniziert. Das Anwendungsprogramm ist mit dem Webobjects Toolkit in Objective-C geschrieben und wird dann kompiliert. Es ist ein Maschinenprogramm. Kommunikation erfolgt durch das Datenbank-Kit von Next über die Native API der unterstützten Datenbanken mit beliebigen Datenbanken. Next bietet außerdem eine exzellente Entwicklungsumgebung mit GUI-Designer und einem sehr schönen ER-Modeller und Debugger. Webobjects skaliert sich ausgezeichnet durch das verwendete RPC-Schema (Next Portable Distributed Objects auf einem beliebigen etablierten RPC-Mechanismus aufsetzend) und die Möglichkeit, den Application Server zu replizieren. Interessant ist die ungeschlagene Geschwindigkeit und die extrem gute Entwicklungsumgebung.

3.4.4 Die Funktionsweise von PHP

Im Gegensatz zu anderen Programmiersprachen erstellt man mit PHP keine eigenständige und ausführbare Datei, sondern bettet den Code lediglich in den Quellcode einer Webseite ein. Die Ausführung des PHP-Codes ist dann die Aufgabe eines speziellen Interpreters. Dazu muss der Server, an den die Anfrage oder die Webseite gesendet wird, den eingebetteten Code erkennen und diesen an den entsprechenden Interpreter weiterleiten. Mit den Befehlen der auszuführenden Skriptsprache lassen sich auf dem Server die vielfältigsten Aufgaben durchführen. So sind z.B. Datenbankabfragen, Dateizugriffe oder auch Auswertungen von Benutzereingaben realisierbar. Das Ergebnis der durch das Skript durchgeführten Aktivitäten muss natürlich irgendwie dem Absender der Anfrage oder der Webseite mitgeteilt werden. Um dies zu erreichen, kann man mit speziellen Befehlen Standardausgaben an den Server vornehmen. In diesen Standardausgaben sind dann ganz normale HTML/XHTML-Befehle enthalten, die der Server in der eingehenden Reihenfolge ausführt und damit die Ausgabe einer Webseite vornimmt. Diese sendet der Server dann an den Empfänger. Auf diese Weise können Sie beim Erstellen des PHP-Skriptes genau das Aussehen und den Inhalt der Antwortseite festlegen. Hier zeigt sich der Vorteil bei der Verwendung einer Skriptsprache wie z.B. PHP: Sie sind in der Lage, auf die Anfragen oder Reaktionen des Besuchers zu reagieren und können genau auf diese Bedürfnisse abgestimmte Webseiten zur Verfügung stellen. Damit fällt dann auch der Begriff dynamische Webseiten - denn genau darum handelt es sich hierbei.

3.4.5 Voraussetzungen für PHP

Hier kommen wir zu einem Punkt, der für die Entwicklung eines zukünftigen Webprojektes von entscheidender Bedeutung ist - die Software-Plattform. In der Regel wird PHP auf UNIX- oder Linux-Servern eingesetzt. Dies schon deshalb, weil die meisten Datenbanken, auf die PHP zugreifen kann, ebenfalls in einer solchen Umgebung laufen. Ungeachtet dessen kann PHP genauso gut in einer Windows-Umgebung installiert und betrieben werden. Da PHP sparsam mit den Speicherressourcen eines Rechners umgeht, steht seinem Einsatz auf einem Standard-Windows-Rechner nichts mehr im Wege. Man kann also ein Webprojekt in aller Ruhe auf einem Windows-Rechner entwickeln und dieses anschließend 1:1 auf den UNIX-Server des Providers übertragen.

3.4.6 Installieren des PHP-Moduls für Apache

Das Installieren des PHP-Moduls unter SuSE Linux 7.x erweist sich als recht einfacher Vorgang. Mittels YaST2 wird PHP als Software hinzugefügt[22]. Als einfachste Methode bedient man sich dabei der Suchfunktion im Fenster der Installation indem man den Begriff „PHP“ eingibt. Unter der angezeigten Auswahl markiert man den Begriff mod_php. Weiterhin kann die PHP-Dokumentation aufgespielt werden.

Nach Bestätigung der Installation sollte der Apache-Web-Server neu gestartet werden (`/usr/init.d/httpd restart`). PHP sollte nun mit der Standard-Konfiguration laufen.

[22] siehe 3.1.7 Installation des Apache Servers und dessen Module mit YaST2

3.4.7 Test einer PHP-Seite

Um eine PHP-Seite zu testen, speichert man eine .php-Datei in Ihrem definiertem Web-Ordner auf dem Apache-Web-Server und ruft diese unter der vorgesehenen Adresse auf.

PHP-Code wird mit Hilfe von SGML Processing Instructions (PIs) in HTML eingebettet. Der Code steht zwischen <?php und ?> :

Beispiel:

```
<?php
  phpinfo();
 ?>
```

Alternativ kann auch ein Script-Tag für Server-Side Scripting verwendet werden:

Beispiel:

```
<script language="php">
  phpinfo();
</script>
```

Falls die Konfigurationsvariable **short_open_tag** gesetzt ist, kann man den Namen der Scriptsprache in den PIs weglassen:

Beispiel:

```
<?
  phpinfo();
 ?>
```

Falls die Konfigurationsvariable **asp_tags** gesetzt ist, kann man statt <? auch <% benutzen.

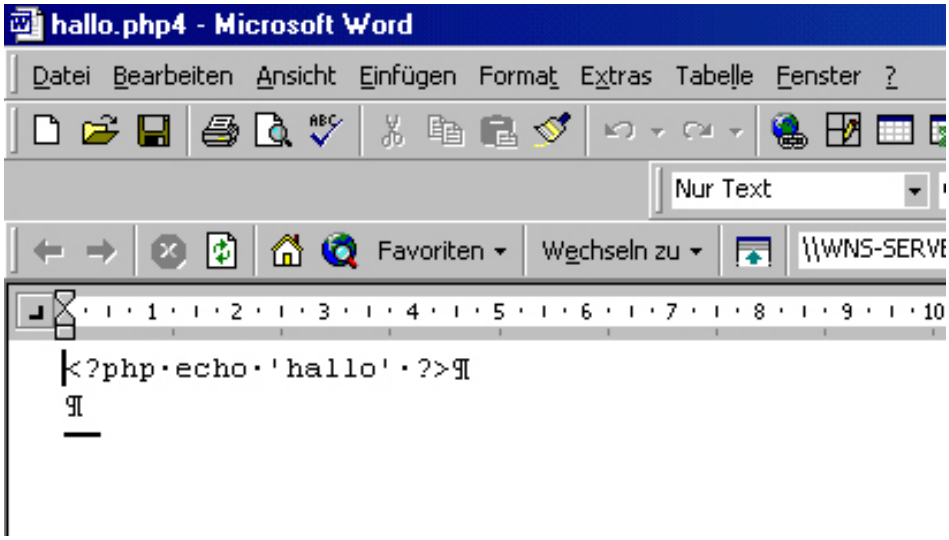
Beispiel:

```
<%
  phpinfo();
%>
```

3.4.8 Ein Beispiel:

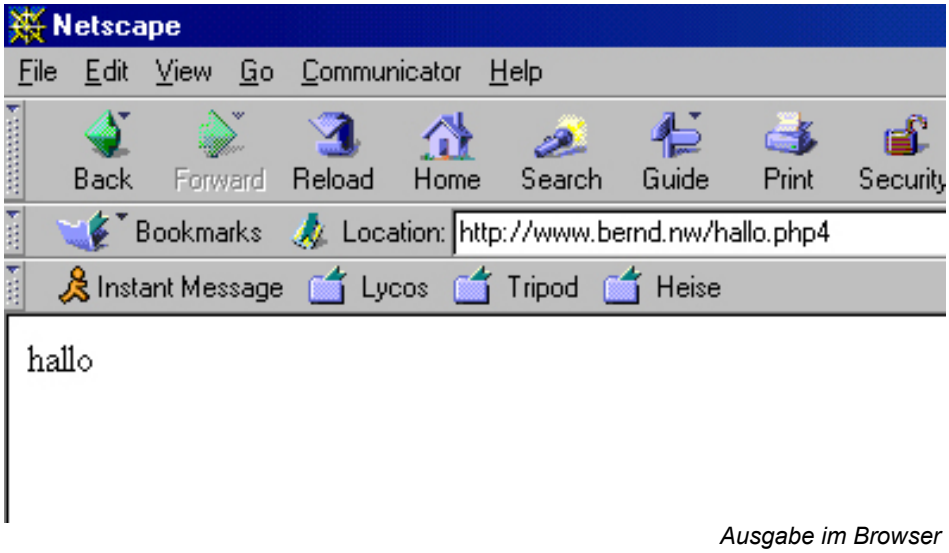
Mit einem Texteditor wird folgendes geschrieben und unter test.php4 gespeichert:

```
<?php echo ' hallo' ?>
```



Eingabe im Editor

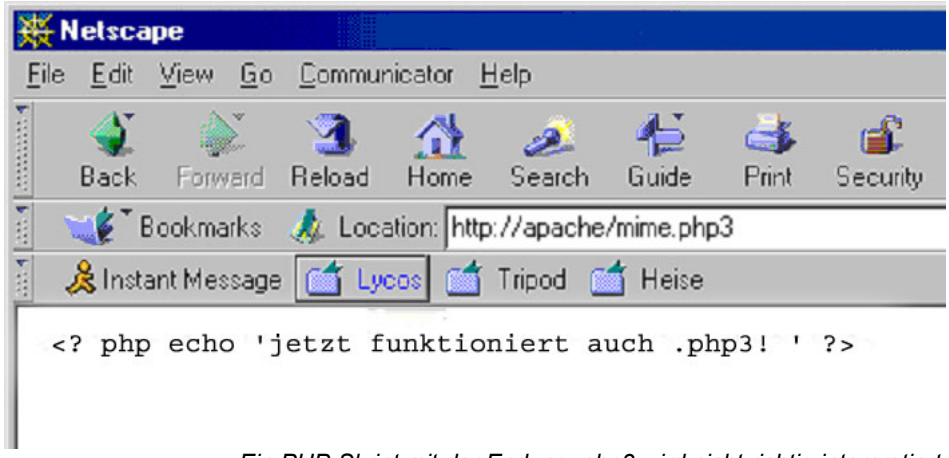
Der Apache-Web-Server greift bei Aufruf des Localhost standardmäßig auf das Verzeichnis `/usr/local/httpd/htdocs` zu. Hierher wird das Script „test.php4“ kopiert. Nun kann mit einem Browser die Webpage `localhost/test.php4` aufgerufen werden. Ist im Browserfenster das Wort "hallo" zu sehen, wird PHP vom Server interpretiert.



Ausgabe im Browser

3.4.9 .php, .php3, .php4

Unser PHP4-Script funktioniert. Was aber passiert, wenn wir unser Programm auf die Endung .php3 umbenennen?

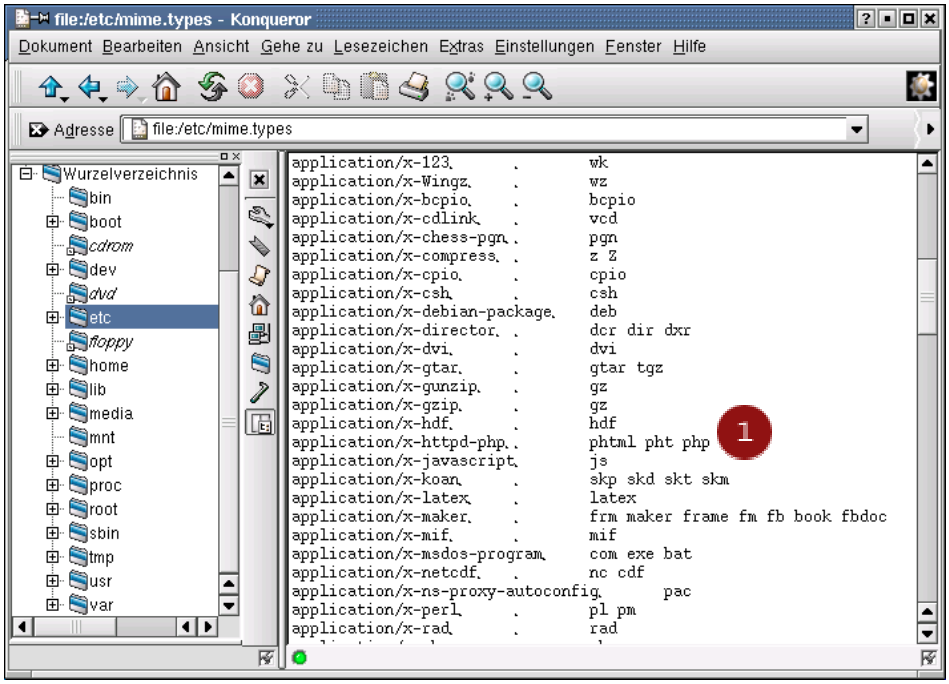


Ein PHP-Skript mit der Endung .php3 wird nicht richtig interpretiert

Der Server erkennt den PHP-Code nicht und gibt Text aus!

Wir müssen kontrollieren, welche MIME-Types[23] für PHP und den Apache-Web-Server eingestellt sind. Dieses geschieht mit der Datei mime.types im /etc-Verzeichnis.

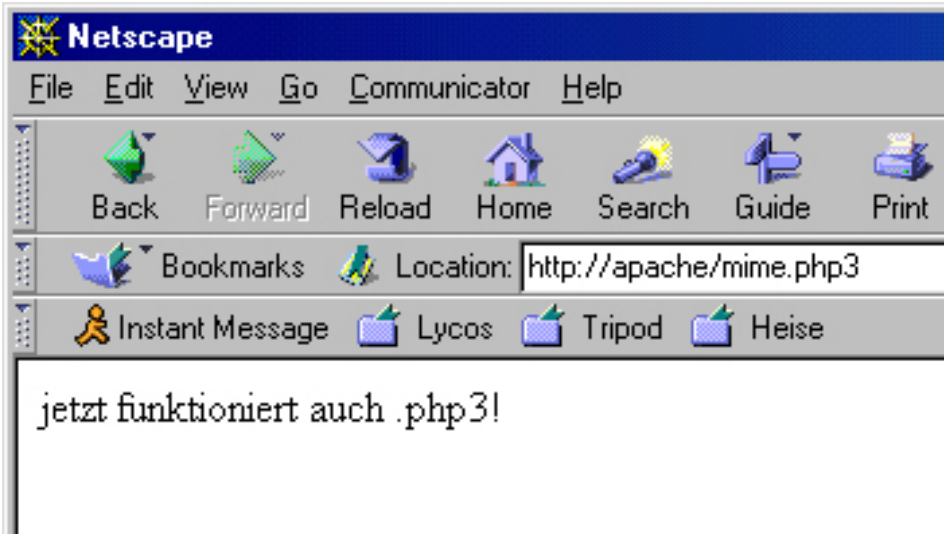
In der Datei `/etc/mime.types` fehlt der Eintrag php3 und sollte nachgetragen werden.



Unter Punkt 1 fehlt der Eintrag .php3

[23] vgl. *3.4.10 MIME-Typen*

Nach einem Neustart des Apache-Web-Servers haben wir unser Ziel erreicht, der Code wird interpretiert.



Richtige Interpretation von .php3

3.4.10 MIME-Typen

MIME (Multipart Internet Mail Extensions) ist eine Methode, mit der der Typ eines bestimmten Dokuments bzw. einer bestimmten Ressource angegeben wird, wobei es sich um eine tatsächliche Datei auf dem Server handeln kann oder auch nicht.

MIME-Typen setzen sich aus einem Haupttyp (z.B. *text* oder *image*) und einem Untertyp (z.B. rtf oder gif) zusammen.

Geschrieben wird ein MIME-Typ wie folgt: *text/rtf*

Dieser MIME-Typ legt fest, dass es sich bei einer Datei um ein Text handelt und dass ihr Dateiformat vom Typ *rtf* ist. Damit kann der Client erkennen, was mit dem Bitstrom zu tun ist, den er empfängt.

Die Registrierungsdatenbank für die Medientypen (Media Types) wird von der IANA (Internet Assigned Numbers Authority) verwaltet.

Das Feld *Content-type* im Antwort-Header, das zusammen mit der eigentlichen Antwort an den Client zurückgesendet wird, kann mehr als nur den Medientyp enthalten (z.B. Zeichensatz in dem das Dokument kodiert ist).

Obwohl diese Zusatzinformation nicht wirklich Teil des eigentlichen MIME-Typs ist, kann man durch den gleichen Apache-Mechanismus, mit dem man den Medientyp manipulieren kann, auch diese Faktoren steuern.

3.4.11 Die Performance von PHP zu Perl im Vergleich

Eine Zählschleife in PHP3 ist um etwa den Faktor 10 langsamer als dieselbe Zählschleife in Perl, weil PHP3 die Zeilen der Schleife bei **jedem Durchlauf** erneut parsed[24] und dann ausführt, während Perl **beim Start** des Interpreters Bytecode erzeugt und dann ausführt.

Typische Programme bestehen nicht aus Zählschleifen, sondern aus komplexen Funktionen, mit denen man das eigentliche Ziel des Programms bewirken will. Solche Funktionen sind in Perlprogrammen in der Regel in Perl realisiert, während diese Funktionen in PHP3 eher in C geschrieben sind. C ist in solchen Fällen etwa um den Faktor 5-10 schneller als Perl. Bytecode-Compiler haben außerdem eine Startup-Zeit, die ebenfalls in die Laufzeit des Programms mit eingeht.

Welche Skriptsprache vorne liegt, hängt extrem von der Aufgabe und dem verwendeten Code ab.

Im Zweifel kann man in beiden Sprachen die entsprechende Funktion als Spracherweiterung in C implementieren und ist so in jedem Fall schneller als mit einer PHP- oder Perlfunktion. PHP3 und PHP4 liegen hier wegen des im Vergleich zu Perl5 sehr viel besser strukturierten Sourcecodes vom Entwicklungsaufwand deutlich vorne. Ein Perl-Projekt, das den Perl-Interpreter neu schreiben will, ist derzeit in Arbeit.

PHP4 erzeugt wie Perl beim Start des Programms Bytecode und führt diesen dann aus. Dabei liegt PHP4 mit Perl geschwindigkeitsmäßig in etwa gleichauf. Ebenso wie Perl braucht auch PHP4 eine größere Startup-Zeit, in der das Programm analysiert und übersetzt wird.

Der Mitte März zum Betatest freigegebene Zend-Optimizer optimiert den PHP4-Bytecode noch einmal und holt je nach Anwendung noch einmal Geschwindigkeitsgewinne heraus. Diese Gewinne können in Benchmarks sehr signifikant sein - ihre praktische Bedeutung wird ebenfalls merkbar sein, aber sicherlich nicht so extrem wie in den Benchmarks.

[24] parsen: Bytecode aus Quelltext erzeugen

Der Zend-Cache kann häufig verwendeten Bytecode erkennen und im Interpreter im Speicher halten. Der Interpreter braucht diesen Bytecode dann nicht mehr zu laden, analysieren und zu übersetzen, sondern kann ihn direkt aufrufen. Durch den Zend-Cache erhöht sich der Speicherverbrauch des Interpreters stark und bei unzureichendem Speicherausbau der Maschine kann sich das negativ auf die Performance des Gesamtsystem auswirken. Der Zend-Cache wird ein „closed source“ Produkt sein und nach Abschluss der Beta-Phase käuflich zu erwerben sein.

Man sagt, dass PHP3 und PHP4 leichter zu erlernen sind als Perl und dass PHP-Code leichter zu lesen und damit billiger zu warten sei als Perl-Code. Das ist sicherlich eine Frage der Übung - man kann in beiden Sprache nicht mehr wartbare Programme entwickeln bzw. in beiden Sprachen selbstdokumentierenden Code abliefern.

In PHP ist die Versuchung, kryptischen Code abzuliefern möglicherweise etwas kleiner als in Perl, aber speziell PHP3 ist hier in einigen Punkten behindert, weil in der Sprache Referenzen[25] keine Objekte erster Ordnung sind und nur sehr eingeschränkt verwendet werden können und daher manchmal einige Workarounds notwendig machen. Dieses Problem ist erst in PHP4 behoben.

3.4.12 PHP als CGI oder Modul

Jedesmal, wenn eine Seite mit PHP-Code auf dem Server ausgeführt werden muss, muss der PHP-Interpreter gestartet werden. Wird PHP als CGI-Programm installiert, bedeutet dies, dass am Anfang der Seite ein neuer Prozeß erzeugt werden muss und der PHP-Interpreter in diesen Prozess mit geladen wird. Dies verbraucht eine ganze Menge Systemressourcen. Am Ende der Seite endet auch der Interpreterprozess und aller Speicher wird freigegeben. Ebenso werden alle Filehandles geschlossen und damit alle Datenbankverbindungen aufgegeben.

Installiert man PHP dagegen als Modul, etwa in einem Apache-Web-Server, dann ist das PHP-Modul Bestandteil des Web-Servers und ständig geladen. Es kann außerdem Datenbanklinks über die Lebensdauer einer PHP-Seite hinaus offen halten, was speziell bei Oracle-Datenbanken große Performancevorteile bringt.

Nur mit einem CGI-PHP ist es möglich, PHP als Scriptsprache zur Erstellung von "Shellscripten" einzusetzen.

Viele Massen-Webhoster setzen bevorzugt CGI-PHP ein, weil es sich leichter auf eine Weise installieren lässt, die die Systemsicherheit nicht gefährdet. CGI-PHP erlaubt den Einsatz der Apache-Erweiterung `suexec`, die die Erstellung einer `chroot()`-Umgebung und die bessere Kontrolle der durch den Anwender verbrauchten Systemressourcen ermöglicht.

Praktisch alle großen PHP-Sites setzen PHP als Modul ein, weil die Performance hier deutlich besser ist.

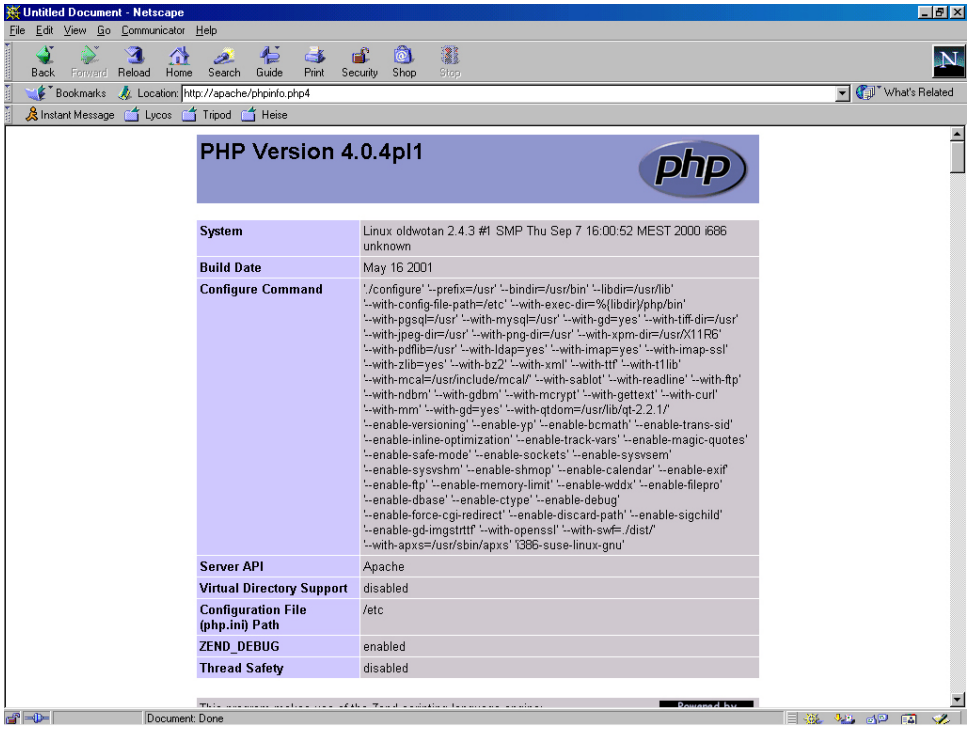
[25] Zeiger auf Variablen

3.4.13 Anzeige der Konfiguration des PHP-Interpreters

Die PHP-Funktion **`phpinfo()`** liefert eine Fülle von Information über die aktuelle Konfiguration des Interpreters. Das typische Testprogramm sieht so aus:

Beispiel:

```
<?php
  phpinfo();
 ?>
```



Ausgabe von php.ini

3.4.14 Die php.ini?

Die Ausgabe von **`phpinfo()`** sagt unter anderem, wo PHP4 seine Konfigurationsdatei php.ini sucht. Steht hier nur php.ini , hat PHP keine Datei gefunden und arbeitet mit Default-Einstellungen (diese entsprechen etwa denen aus der php.ini-dist); weniger irreführend ist die Ausgabe von **`echo get_cfg_var('cfg_file_path')`**. Sie ist in diesem Fall einfach leer.

3.4.15 Was safe_mode bewirkt

Es gibt eine Konfigurationsvariable **safe_mode**, die in der php3.ini gesetzt werden kann. Weiterhin gibt es die Konfigurationsvariablen **safe_mode_exec_dir** und **sql_safe_mode**.

Wenn **safe_mode** aktiv ist, sind verschiedene PHP3-Funktionen privilegiert oder eingeschränkt. Zumeist heißt die Einschränkung **safe_mode** , dass auf eine Datei oder ein Verzeichnis nur eingewirkt werden darf, wenn die Datei oder das Verzeichnis denselben Eigentümer hat wie das Script. Im einzelnen:

Alle Dateifunktionen einschließlich **include()** und **require()** können nur noch mit lokalen Dateien arbeiten, die denselben Eigentümer (uid) haben wie der Eigentümer des Scriptes.

Auch ftp-Pfadnamen und zlib-Pfadnamen werden so geprüft.

Als Fazit könnte man sagen: **safe_mode** ist nicht sicher: Ein Fehler in der **popen()** - Funktion ist erst mit 3.0.14 korrigiert worden, ein weiterer Fehler in der **mail()** - Funktion erst in 3.0.15. Man sollte stattdessen die CGI-Version in einem chroot-Environment verwenden und mit **setrlimit** noch weitergehende Einschränkungen definieren.

3.4.16 Parameterweitergabe einer PHP-Seite an ein anderes Skript

Man kann Parameter an ein PHP-Skript als HTTP-GET- oder HTTP-POST-Parameter übergeben.

Die Abfolge ist immer gleich. Die Daten werden eingegeben, an den Server gesendet und verarbeitet. Dann sendet das Skript eine neue Seite an den Webbrowser. Im Formular-Tag legt man das aufzurufende Skript fest. Auch die Übertragungsart wird hier bestimmt.

```
<form action="feedback.php" method="post">
```

Dies ruft die Datei *feedback.php* auf. Der Aufruf kann durchaus der aktuellen Datei entsprechen.

Die Daten werden mit der Methode POST übertragen. Dies ist bei Formularen die am häufigsten genutzte Übertragungsmöglichkeit. Da die Daten jedoch von der URL getrennt in einem eigenen Bereich übertragen werden, weiß der Browser anscheinend nicht so recht, wie er sich bei einem Reload der Seite verhalten soll. Microsoft Internet Explorer und Netscape Navigator sind auch hier nicht einig.

Eine zweite Variante ist GET. Dabei werden die Eingaben an die URL angehängt. Hier kann die Seite mit den Parametern in den Bookmarks abgelegt werden, jedoch ist die Menge der Daten im Gegensatz zu POST begrenzt: Es sind circa 1 KB Zeichen erlaubt.

Einen HTTP-GET-Request erzeugt man, indem man einfach einen Link auf das gewünschte Skript erzeugt und die Parameter mit der Funktion **urlencode()** codiert anhängt.

Das empfangende Skript wird diese Parameter ganz normal entgegennehmen, automatisch decodieren und als globale Variablen mit den Namen $para1 und $para2 bereitstellen.

Die Länge der durch einen GET-Request übergebaren Parameter ist begrenzt. Im einem GET- oder POST-Request übergebene Parameter sind durch den Anwender leicht manipulierbar. Es ist wesentlich besser, Sessionvariablen zu verwenden, wie sie etwa durch PHPLIB oder mit Hilfe der Sessionfunktionen von PHP4 realisiert sind - im Gegensatz zu dem hier gezeigten manipulierbaren Verfahren sind Sessions nämlich sicher.

3.4.17 Anzeigen des PHP-Codes im Browser

Wenn der Webserver ordnungsgemäß konfiguriert ist und er PHP unterstützt, bekommt der Besucher der Seite den PHP-Code nicht zu sehen, da er vom Webserver geparst wird und nur der generierte HTML-Code an den Browser geschickt wird.

Folgende Szenarien sind jedoch denkbar, bei denen der Besucher den PHP-Code trotzdem zu sehen bekommt:

Auf dem Webserver ist kein PHP-Parser installiert. In diesem Fall behandelt der Webserver den PHP-Code wie "normalen" HTML-Code und schickt ihn unbearbeitet (ergo ungeparst) an den Browser.

Aufgrund einer Fehlfunktion des Webservers ist der PHP-Parser ausgefallen oder der Webserver benutzt ihn nicht mehr: In diesem Fall sieht der User ebenfalls den ungeparsten Source-Code im Browserfenster, da kein Parsing mehr stattfindet.

Jemand verschafft sich per FTP unberechtigt Zugriff auf die Daten: Auch hier hat der User die Möglichkeit, den Source-Code einzusehen, da der PHP-Parser nur Dateien parst, die über den Webserver abgefragt werden. Andere Schnittstellen wie zum Beispiel FTP berücksichtigt er nicht.

Ein User verschafft sich Zugriff auf den Webserver mit Telnet oder SSH: Auch hier kann er den PHP-Code ohne weiteres einsehen, indem er zum Beispiel **cat dateiname.php** eingibt und der Inhalt der Datei **dateiname.php** auf dem Bildschirm ausgegeben wird.

In jedem Fall ist es sinnvoll, Code mit Datenbankpassworten in einem Verzeichnis abzulegen, das nicht durch eine URL erreichbar ist oder das mit einer .htaccess-Datei gegen Zugriff gesichert ist. Dieser Code kann dann mit Hilfe von `include()` oder `require()` eingebunden werden.

3.4.18 Kompilieren von PHP-Dateien und Schutz vor Dritten

Anders als beispielsweise bei Java wird ein PHP-Code erst zur Laufzeit (wenn das Script aufgerufen wird) kompiliert, d.h. der Programmcode liegt im "Klartext" in den .php -Dateien. Für normale Besucher einer von PHP erzeugten Webseite ist der Programmcode nicht einzusehen. Solange man die Skripte nur auf dem eigenen Webserver einsetzt, ist der Code für andere unsichtbar. Gibt man PHP-Dateien jedoch an Kunden weiter, erhalten diese damit den Source Code. Wenn der Code nicht "Open Source" ist, sollte man per Vertrag das Copyright und die Nutzungsbedingungen genau regeln.

Von Zend gibt es als kommerzielles Produkt den Zend Encoder Unlimited, der PHP-Code in ein binäres Format überführt und damit weitgehend vor Ausspähung, Veränderung und Re-engineering schützt. Das Produkt ist nicht gerade billig, außerdem braucht man natürlich auf jedem Webserver, der kompilierten Code ausführen soll, ein Zusatzmodul (dieses allerdings ist kostenlos).

Nach dem gleichen Prinzip funktioniert der PHP Bytecode Compiler, der im derzeitigen Beta-Stadium allerdings noch keinen objektorientierten Code verarbeiten kann. Die Kompilierung erfolgt außerdem "online".

Auch microCODE macht aus PHP-Code einen nicht lesbaren Bytecode, der durch ein einzubindendes .so-Modul vom PHP-Interpreter ausgeführt werden kann.

Einen etwas anderen Weg gehen die Tools Code Obfuscator und POBS - sie kommen ohne serverseitige Module aus, da sie den Code nur für Menschen, nicht aber für den PHP-Interpreter "unleserlich" machen. Der Code Obfuscator ersetzt die meist sprechenden Variablennamen durch nichtssagende Konstrukte, so dass es schwer ist, die Funktion eines so bearbeiteten Skripts nachzuverfolgen. POBS macht dasselbe, geht aber noch weiter, indem auch Konstanten und Funktionsnamen verändert und Kommentare, Einrückungen sowie Leerzeilen entfernt werden. Der resultierende Code ist immer noch gültiger PHP-Code, aber seine Funktionsweise ist kaum mehr zu entziffern.

3.5. Web-Account erstellen und verwalten

3.5.1 Definition

Um eine eigene Web-Seite ins Internet stellen zu können, muss man auf dem Server seines Internet Service Providers Platz in Form von Festplattenspeicher auf einem Server anmieten. Der Provider sorgt dafür, dass der Benutzer auf seinem Server die persönlichen Seiten ablegen kann und jedermann im Internet auf diese Dateien zugreifen kann. Ein Provider hat direkte Verbindungen zu anderen Internet Service Providern und zentralen Internet-Austauschpunkten. Man muss deshalb für seine Internet-Präsentation einen solchen "Web-Account" einrichten und Speicherplatz im World Wide Web mieten.

Um Daten aufzuspielen oder zu verändern greift der Benutzer für gewöhnlich über FTP auf den Server zu und benutzt hierfür ein FTP-Programm welches es ihm erlaubt, beliebige Dateiformate innerhalb eines TCP/IP basierten Netzwerks auf den ihm zugewiesenen Web-Space zu übertragen.

Für unser Projekt wollen wir ein solches Netz simulieren. Dabei ergibt sich folgende Situation:

Die Apache Projektgruppe möchte einen Account erstellen, um ihre Ergebnisse im Intranet zu präsentieren. Das betrifft die Administration des Servers. Der Administrator des Servers legt dafür einen neuen User, nämlich "projekt" auf dem Linux-System an. Der zugehörige Ordner lautet `/www/docs/projekt-apache`.

Er teilt der Projektgruppe diesen Account mit dem zugehörigen Passwort mit. Zusätzlich richtet der Administrator die URL ein, mit der die Intranetseite zu erreichen ist. Dieses lautet in unserem Fall *http://www.projekt-apache.bbs*. Die Projektgruppe kann nun diesen Speicherplatz per FTP-Zugang verwalten.

3.5.2 FTP

Um auf den Web-Server zugreifen zu können, bedient man sich der Hilfe eines FTP-Programms. Dieses übernimmt den Part des Clients und kommuniziert mit dem Web-Server.

Der erste Schritt besteht im Verbindungsaufbau. Das heißt Client und Server einigen sich auf eine Basis auf der sie kommunizieren und Daten austauschen können. Außerdem muss der Client vom Server Rechte und Regeln zugewiesen bekommen, um überhaupt Dateien hin und her schieben zu können.

Wir wollen am Beispiel des Programms WS_FTP Pro die Benutzung eines FTP-Programms unter Windows zeigen. Wir denken, dass das einem realen Szenario sehr nahe kommt, denn heutzutage erstellt man seine Homepage meist mit einem Gestaltungsprogramm unter Windows und lädt anschließend seine erstellte Homepage via FTP auf den Web-Server.

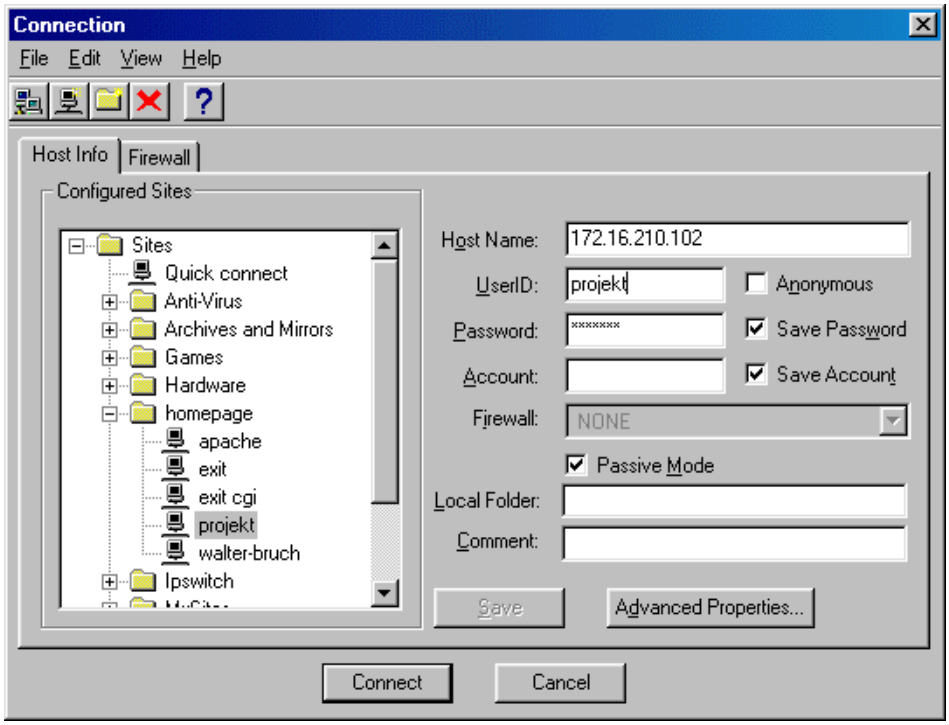
Andere FTP-Programme sind vom Aufbau her WS_FTP Pro sehr ähnlich und deshalb genau so zu bedienen.

WS_FTP Pro

Im Feld *Hostname* ist eine URL (z.B. *ftp://ftp.suse.de*) oder direkt die IP-Adresse (z.B. 176.13.134.234) eines FTP-Servers einzutragen.

Im nächsten Feld wird der einem zugewiesene Benutzername eingetragen. Dieser sagt dem FTP-Server, welcher Gruppe man angehört. Mit Hilfe dieser Angabe ordnet er Benutzern Rechte und Regeln zu.

Nun wird das festgelegte Passwort im Feld *Password* eingetragen. Meistens erhält man von vornherein einen festen Benutzernamen und Passwort.



Der Verbindungsaufbau zu einem FTP-Server mit WS_FTP Pro

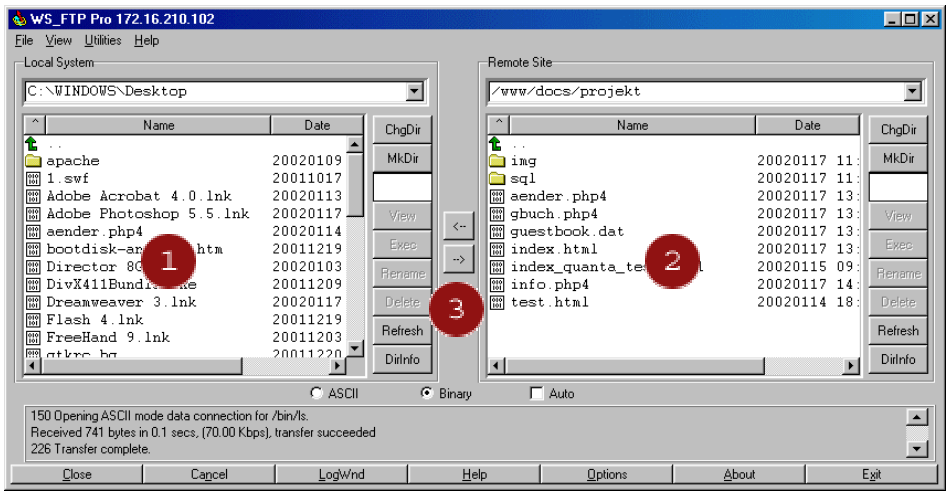
Die restlichen Felder bleiben frei. Manche FTP-Server brauchen allerdings noch die Angabe eines *Accounts*, um den Client zu authentifizieren und den Zugang zu einem Standardverzeichnis zu gewähren.

Das Feld *Local Folder* beschreibt den Ordner, auf dem man mit der Session beginnen will. Dies könnte ein speziell für diesen User freigeschalteter Ordner sein, auf den nur dieser User Zugriff hat.

In das Feld *Comment* trägt man sich ein paar Sätze zur Erinnerung an diese Seite ein.

Nach dem Klick auf den Button „Connect“

Nach dem erfolgreichen Einloggen auf einem FTP-Server erhält man ein Fenster, das an einen Dateimanager erinnert. Links (*Punkt 1*) sieht man den lokalen Rechner, also die eigene Festplatte und rechts (*Punkt 2*) ist die Verzeichnisstruktur des FTP-Servers zu sehen. Idealerweise befindet man sich schon im richtigen Ordner, der einem aufgrund des Benutzernamens zugewiesen wurde.

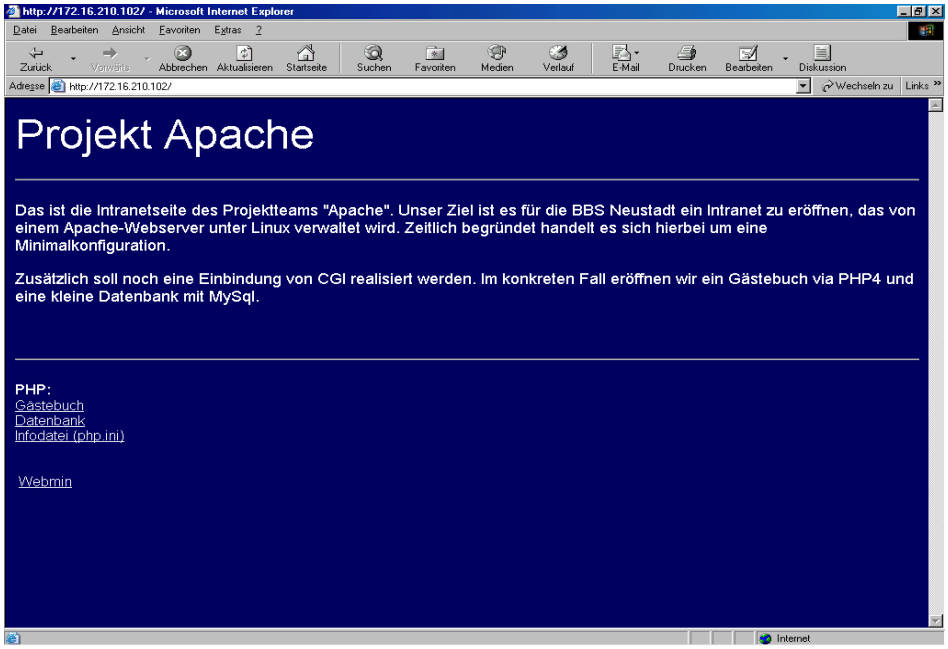


Der Explorer, mit dem man Dateien hin und her schieben kann

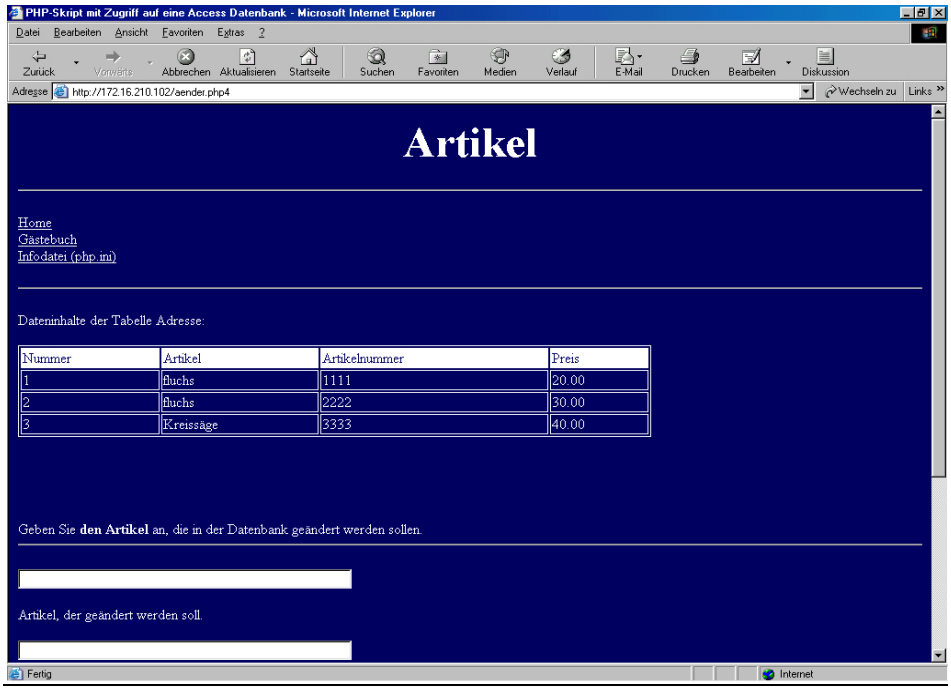
Dateien lassen sich nun einfach mit den Pfeilen über *Punkt 3* zwischen beiden Systemen hin und her schieben. Außerdem können Dateien verändert oder gelöscht und Ordner erstellt werden.

3.5.3 Das Layout der Seiten

Hier stellen wir das Layout unserer selbst erstellten HTML-Seite vor. Diese Seiten sind sehr einfach aufgebaut, weil die Betonung auf der Funktion liegt und nicht auf der Gestaltung.

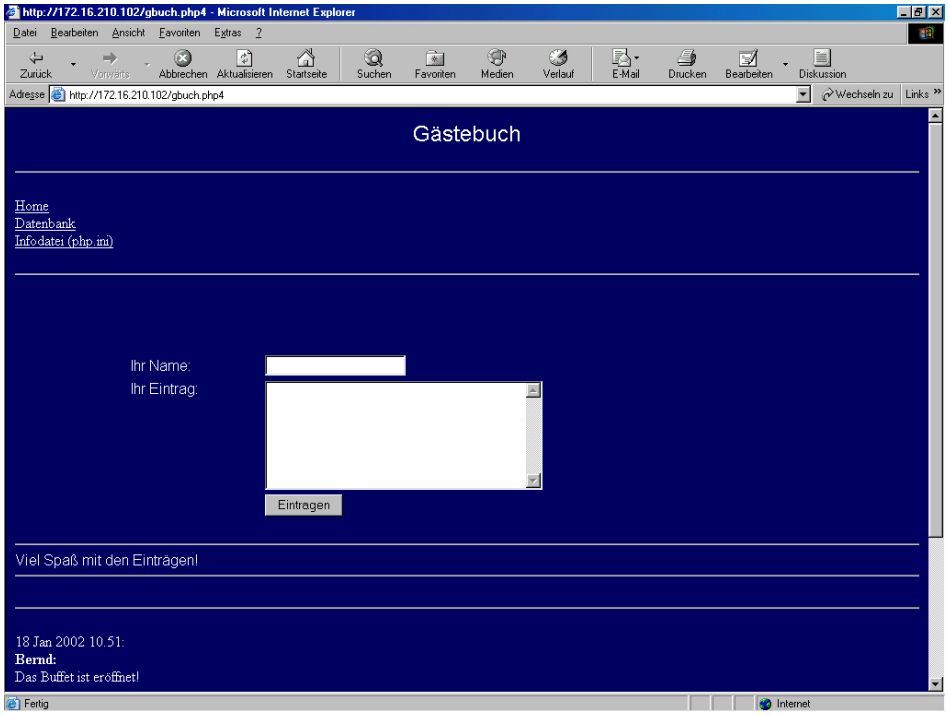


Die Startseite Index.html



Eine Datenbank

Die Datenbank war ein kleines Versuchsprojekt innerhalb des eigentlichen Projektes und bleibt unkommentiert



Das Gästebuch

3.5.4 Erläuterung des PHP-Gästebuchs

Um einen praxisorientierten Einblick in PHP zu erhalten wird der HTML-, und PHP-Code aufgelistet und interpretiert.

```
<html>
```

<!-- Das "Gesicht" des Gästebuchs -->

```
<body bgcolor="#000066" text="#FFFFFF" link="#FFFFFF" vlink="#FFFFFF"
alink="#FFFFFF">
<center>
<p><font face="Arial, Helvetica, sans-serif" size="5">Gästebuch</font>
</p>
<hr>
```

<!-- Links zu den anderen Seiten des Intranets -->

```
<p align="left"><a href="index.html">Home</a><br>
<a href="aender.php4">Datenbank</a><br>
<a href="info.php4">Infodatei (php.ini)</a> </p>
<hr>
```

<!-- Hier beginnt das Formular für den Dateneintrag -->

```
<form action="gbuch.php4" method="get">
<br>
<br>
<br>
<table width="75%" border="0">
<tr>
<td><font face="Arial, Helvetica, sans-serif">Ihr Name:</font></td>
<td>
```

<!-- Texteingabefeld mit der Bezeichnung "new_name" -->

```
<input type="text" name="new_name">
</td>
<td> </td>
</tr>
<tr>
<td valign="top"><font face="Arial, Helvetica, sans-serif">Ihr
Eintrag:</font></td>
<td>
```

<!-- Texteingabefeld mit der Bezeichnung "new_input" -->

```
<textarea name="new_input" value="" rows="7" cols="35"></textarea>
</td>
<td> </td>
</tr>
<tr>
<td> </td>
<td>
```

<!-- Schaltfläche um Daten abzuschicken -->

```
<input type="Submit" name="send" value="Eintragen">
</td>
<td> </td>
</tr>
</table>
<br>
</form>
```

<!-- Ende des Formulars -->

```
<hr>
</center>
<div align="left"><font face="Arial, Helvetica, sans-serif">Viel
Spa&szlig; mit
den Eintr&auml;gen! </font>
<hr>
</div>
```

<!-- Beginn des eingebetteten PHP-Scripts -->

```
<?php

// ### Die Varialbe "$bookfile" ist eine Art "Schlüssel" bzw. eine
// ### Zuordnung zu der Datei "guestbook.dat" damit das Php-Script die
// ### Datei überhaupt erkennt

$bookfile="guestbook.dat";

// ### Wenn im Feld "new_input" des Formulars zum Zeitpunkt des
// ### eintragens etwas steht, also die Bedingung <>"" erfüllt, wird
// ### die If-Abfrage ausgeführt.

If ($new_input<>"")
```

```
// ### Ist die Bedingung erfüllt, wird die Funktion "write_input"
// ### aufgerufen dabei werden die Variablen des Formulars
// ### "$new_name","new_input" und "$bookfile"an die Funktion
// ### übergeben.

write_input($new_name, $new_input,$bookfile);

// ###   Aufruf der Funktion "read_file"

read_file($bookfile);

// ###   Funktion "write_input"

function write_input($new_name, $new_input,$bookfile){
$res= time();

// ###   "time()" ließt per php die aktuelle Zeit und Datum des
// ###    Unix-Systems ein.

$dateform= gmdate("d M Y H.i", $res);

// ### formatieren von Zeit und Datum

$fd = fopen($bookfile,"a");

// ### $fd nimmt die Zeigerposition auf
// ### öffnet die Datei
// ### "a" setzt den Zeiger an den Schluss der Textdatei und
// ### und ist bereit zum schreiben.

fwrite ($fd, "<hr><br>".$dateform.":<br><b>".$new_name.":</b><br>
".$new_input."\r \n");

// ### Die Werte der Variablen, also die Formulareingaben
// ### werden in die Datei eingetragen.

fclose ($fd);
}
```

<!-- Funktion "read_inpunt" --!>

```
function read_file ($bookfile){

// ### ist nach wie vor die  Datei "guestbook.dat"

$fd = fopen($bookfile,"r");

// ### fopen = Datei wird geöffnet,
// ### "r" setzt den Zeiger an den Dateianfang
// ### $fd nimmt die Zeigerposition auf

while (!feof($fd)) {

// ### ! bedeutet "ungleich",
// ### eof ist das Dateiende
// ### bedeutet: Schleife wird solange
// ### durchlaufen bis der Zeiger am Ende ist.


$zeile = fgets($fd,1000);

//### fgets ließt eine Zeile aus in der der Zeiger steht ($fd)
//###  mit einer Länge von 1000 bytes

echo $zeile . "<br>";

//### Gibt den Zeileninhalt im Browser aus
//### Zeilen werden solange ausgegeben (while) bis das Dateiende
//### erreicht ist

}
fclose ($fd);
}

?>
<hr>

</body>
</html>
```

4. Aufgabenverteilungsplan

Thorsten Behrend

- Apache
- Geschichte von Apache
- Vorteile von Apache
- Mime-Type Definition

Felix Hock:

- Apache-Web-Server Konfiguration
- httpd.conf
- virtual hosting
- Linux

Christian Keller:

- Linux
- Installation
- Konfiguration
- YaST2

Bernd Wissmeier:

- PHP-Skriptdokumentation
- Erstellung von PHP-Skripten
- Gästebuch und Dokumentation erstellen
- Intranet (Erstellen von vier Seiten)
- Cover

5. Protokolle

Protokoll

Projektsitzung	B-Gebäude, Raum 208
Datum:	05.11.2001
Uhrzeit:	13.00 Uhr
Protokollführer	Felix Hock

Thema: 1. Allgemeines Brainstorming
2. Themenformulierung

Top1: Brainstorming

Bei der heutigen Sitzung wurden allgemeine Ideen zu dem Thema Apache-Web-Server der einzelnen Gruppenmitglieder zusammengetragen und niedergeschrieben.

Top2: Themenformulierung

Jedes Gruppenmitglied soll bis zur nächsten Projektsitzung eine Themenformulierung schriftlich festhalten.

Protokoll

Projektsitzung	B-Gebäude, Raum 208
Datum:	08.11.2001
Uhrzeit:	13.00 Uhr
Protokollführer	Felix Hock

Thema: 1. Themenabgabe

Top1: Themenabgabe

Alle Themenformulierungen wurden vorgetragen. Aus den einzelnen Schriften ließen sich gedanklichen Gemeinsamkeiten und Unterschiede ableiten. Hieraus lies sich folgendes Thema ableiten:

Konfigurieren eines Apache-Web-Servers unter Linux mit CGI-Anbindung mittels PHP

Protokoll

Projektsitzung	B-Gebäude, Raum 208
Datum:	14.11.2001
Uhrzeit:	15.30 Uhr
Protokollführer	Felix Hock

Thema: 1. Festlegung der Ziele
2. Abarbeitung der W-Fragen

Top 1: Festlegung der Ziele

Es wurden die allgemeinen sowie die persönlichen Ziele festgelegt (siehe allgemeine und persönliche Ziele)

Top 2: Abarbeitung der W-Fragen

Die W-Fragen (Wer, Wie, Was) wurden abgearbeitet (siehe W-Fragen)

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	20.11.2001
Uhrzeit:	13.00 Uhr
Protokollführer	Felix Hock

Thema: 1. Installation von Linux
2. Literaturbeschaffung

Top1: Installation von Linux

Das Betriebssystem Linux wurde auf dem für unser Gruppe reservierten Computer installiert.
Eine erste Orientierungsphase folgte (wo liegen welche Verzeichnisse, Konqueror, Festlegung des Anmeldenamens, Festplattenaufteilung, Verwendung von YaST2, Installation verschiedener Module(Staroffice, Netscape, Apache etc..), Zuweisung der IP-Adresse (172.16.201.102)).

Top 2: Literaturbeschaffung

Um sich einen schnellen und fundierten Einblick in das Betriebsystem Linux zu verschaffen und um überhaupt einen Apache-Web-Server zu installieren zu können ist es unabdingbar sich Fachliteratur zu besorgen. Thorsten Behrend und Felix Hock sollen hierfür in die Landesbibliothek Speyer fahren sowie in der Neustädter Gemeindebibliothek nach Büchern suchen. Im Besonderen Bernd Wissmeier aber auch alle anderen Gruppenmitglieder sollen nach Informationen im Internet suchen. Christian Keller soll seine schon vorhandene Literatur mitbringen.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	27.11.2001
Uhrzeit:	15.30 Uhr
Protokollführer	Felix Hock

Thema: 1. Literatur/Informationen zu Linux/Apache
2. Vorbereitung zur Serverkonfiguration

Top 1: Literaur/Informationen zu Linux/Apache

Folgende Literatur wurde zusammengetragen:

- Apache, Dr. Susanne Wigard, bhv-Verlag, Landsberg, 2000
- Apache und CGI – administration und programmierung, Rich Bowen und Ken Coar, Markt und Technik Verlag, München, 2000
- Linux – Apache Web Server Administration, Charles Aulds, Sybex Verlag, Düsseldorf, 2001
- PHP 4, Mark Kronsbein und Thomas Weinert, Sybex Verlag, Düsseldorf, 2001
- PHP 4, Uwe Hess und Günter Karl, bhv Verlag, Landsberg, 2000
- Apache, Andrew Ford, O'Reilly Verlag
- Linux, Kofler, Adison Wessly Verlag

Die Literatur wird als ausreichend betrachtet und soll vorerst als alleinige Informationsquelle zwecks der Übersichtlichkeit dienen.

Top 2: Vorbereitung zur Serverkonfiguration

Allgemeine Kenntnisse aus dem bisher gelesenem über die Serverkonfiguration wurden zusammen getragen. Die Gruppenmitglieder sollen sich bis zur nächsten Sitzung eingehend über die Serverkonfiguration informieren, damit dann eine solche durchgeführt werden kann.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	04.12.2001
Uhrzeit:	15.30 Uhr
Protokollführer	Felix Hock

Thema: Apache

Top1: Apache

Es soll versucht werden einen virtuellen Host einzurichten. Hierzu werden die httpd.conf, die rc.config sowie die hosts-Datei verändert. Zur Absicherung soll von jeder der Originaldateien eine Sicherungskopie erstellt werden. Erstes Ergebnis: nichts geht mehr, selbst der Localhost ist nicht mehr aufrufbar. Auch nach dem Löschen der veränderten Dateien und Einsatz der Originaldateien tut sich nichts.

Da sich auch nach mehreren Versuchen und unterschiedlichster Einstellungen keine wesentlichen Veränderungen sichtbar machen, sieht die Gruppe nur noch zwei Möglichkeiten das Problem zu beheben: entweder soll Herr Scheib um Hilfe gefragt oder das Betriebssystem neu installiert werden.

Die Projektsitzung wird vertagt.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	10.12.2001
Uhrzeit:	13.00 Uhr
Protokollführer	Felix Hock

Thema: 1. Neuinstallation von Linux
2. Apache-Web-Server
3. PHP

Top 1: Neuinstallation von Linux

Da unser System nun nur noch sehr fehlerhaft lief und auch Herr Scheib die ursprüngliche Ausgangskonfiguration nicht mehr herstellen konnte mehrten sich die Verdachtsmoment dafür, das an unser System manipuliert worden ist. Das System wurde vom Computer deinstalliert und wieder neu aufgespielt. Sämtliche Module wurden erneut installiert und das Kennwort für „root“ geändert.

Top 2: Apache-Web-Server

Nach mehreren verschiedenen Konfigurationsversuchen innerhalb der Konfigurationsdateien liefen mehrere Web-Server auf unserem Rechner.

Top 3: PHP

Aufgrund unseres soeben verbuchten Erfolges wurde beschlossen sich bis zur nächsten Sitzung mit der Funktionsweise von PHP auseinander zusetzten. Wenn möglich soll bis dahin ein kleines PHP-Skript angefertigt werden.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	13.12.2001
Uhrzeit:	13.00 Uhr
Protokollführer	Felix Hock

Thema: 1. PHP
2. Aufgabenverteilung

Top 1: PHP

Insgesamt wurden 2 Skripte geschrieben die jeweils in Gruppenarbeit entstanden. Die einzelnen Befehlszeilen wurden den Gruppenmitgliedern gegenseitig erklärt. Offene Fragen bezüglich PHP wurden gemeinsam erörtert und gelöst. Ein weiteres Skript wurde geschrieben und unter Linux-Apache ausgeführt.

Top 2: Aufgabenverteilung

Da die Ferien in 6 Tagen beginnen und noch mehrerer Klausuren zu schreiben sind, verzichten wir auf eine weitere Projektsitzung und eine Aufgabenverteilung wird vorgenommen.

Hiernach übernehmen folgende Gruppenmitglieder folgende Aufgaben:

Thorsten Behrend: Zusammentragen von Informationen über die Entstehung und Vorteile eines Apache-Web-Servers

Felix Hock : Zusammentragen von Informationen über die drei Konfigurations-dateien

Christian Keller: Zusammentragen von Informationen über die Geschichte von Linux und dessen allgemeine Konfiguration

Bernd Wissmeier: Zusammentragen von Informationen über die Funktionsweise von PHP im Allgemeinen und auf Apache bezogen

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	14.01.2002
Uhrzeit:	8.00 Uhr – 18.00 Uhr
Protokollführer	Felix Hock

Intensivphase

Die einzelnen Gruppenmitglieder widmeten sich gemäß dem Aufgabenverteilungsplan ihren Bereichen zu.

Gegen Ende der Arbeitsphase fanden sich alle Mitglieder im Raum 210 zusammen um sich gegenseitig ihre Ergebnisse zu präsentieren und zu erklären

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	15.01.2002
Uhrzeit:	8.00 Uhr – 18.00 Uhr
Protokollführer	Felix Hock

Intensivphase

Die einzelnen Gruppenmitglieder widmeten sich gemäß dem Aufgabenverteilungsplan ihren Bereichen zu.

Es wurden zusätzlich kleiner Aufgaben wie Definitionen einzelner Punke oder Erfassen der gesamten Literatur etc. von den einzelnen Gruppenmitgliedern erledigt.

Des weiteren wurde das wu-ftp-Modul installiert und konfiguriert welches zum „herauf-“ und „herunter-“ laden der einzelnen Programme benutz werden soll.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	16.01.2002
Uhrzeit:	8.00 Uhr – 18.00 Uhr
Protokollführer	Felix Hock

Intensivphase

Die einzelnen Gruppenmitglieder widmeten sich gemäß dem Aufgabenverteilungsplan ihren Bereichen zu.

Zusätzlich wurde ein Gästebuch in PHP erstellt und mittels wu-ftp auf den Server gestellt.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	17.01.2002
Uhrzeit:	8.00 Uhr – 18.00 Uhr
Protokollführer	Felix Hock

Intensivphase

Die einzelnen Gruppenmitglieder widmeten sich gemäß dem Aufgabenverteilungsplan ihren Bereichen zu.

Zusätzlich wurde eine Datenbank mittels mysql erstellt und auf den Server geladen.

Alle bis heute erstellten Schriften wurden zusammengetragen und in eine sinnige Reihenfolge gebracht. Die Gliederung wurde durch die fortschreitenden Forschungsergebnisse angeglichen.

Protokoll

Projektsitzung	B-Gebäude, Raum 210
Datum:	18.01.2002
Uhrzeit:	8.00 Uhr – 16.00 Uhr
Protokollführer	Felix Hock

Intensivphase

Die einzelnen Gruppenmitglieder widmeten sich gemäß dem Aufgabenverteilungsplan ihren Bereichen zu.

Die Projektgliederung wurde besprochen und diskutiert einzelne, Punkte wurden ergänzt, verschoben oder weggelassen.

Es wurden Überlegungen zur Präsentation angestellt und begonnen daran zu arbeiten.

6. Erfahrungsberichte

6.1 Thorsten Behrend

Im Vergleich zu den vorherigen Projektthemen war dieses, aus meiner Sicht, das schwerste. Ich habe gemerkt, dass bei mir das Verständnis im Allgemeinen dafür zwar vorhanden ist, aber die technische Umsetzung mir schwer fällt. Erschwerend kam hinzu, dass die anderen Projektteilnehmer schon Wissen mitbrachten, zumindest in Linux. Sie nahmen sich aber Zeit, um mit mir die Lücken zu schließen, so dass ich nicht nur daneben stehen musste.

Die Gruppe an sich war gut zusammengestellt und ich könnte mir vorstellen ein weiteres Projekt mit ihnen zu bearbeiten.

Es war interessant zu sehen, wie große Internetportale, wie z.B. web.de arbeiten und vor allem diese Technik bei uns im Kleinen zu simulieren.

In Zukunft werde ich mich mehr mit Linux beschäftigen und auf das im Projekt, sowie in der Schule, erlernte Grundwissen aufbauen.

6.2 Felix Hock

Der Anfang des Projektes war kein guter.

Voller Enthusiasmus und Vorfreude bald das Wundersystem Linux auf meinem Rechner zu haben um dann kräftig irgendwelche sagenhaften Sachen zu programmieren, ließ mich nicht ganz rational an das Projekt herangehen.

Als der Erste von uns Linux auf seinem Rechner parallel zu Windows installiert hatte und keine wesentlichen Probleme auftraten, fing auch ich an Linux auf meinen Rechner zu installieren, natürlich nicht ohne mir vorher ein teures Buch für Linux-Einsteiger zuzulegen.

Seit diesem Tag wusste ich, dass ich wohl nie zu den Leuten gehören würde, die den Pinguin am liebsten mit ins Bett nehmen würden um zusammen „Windows“ kaputtzuschlagen.

Es hat zwei Wochen gedauert, um einen meiner zwei Computer (leider hatte ich zu dieser Zeit auch den Computer meiner Eltern bei mir zuhause der auch daran glauben musste) von Linux wieder zu befreien und zum laufen zu bringen. Dabei wirkte das winkende Linux-Pinguin-Monster bei jedem Neustart besonders erregend auf mein Gemüt (selbst jetzt, in dem Moment in dem ich diese Zeilen schreibe fängt mein Herz an schneller zu schlagen - vor lauter Gram und Ärger).

Hätte ich diese Zeit sinnvoll nutzen können bzw. wäre dies nicht so außerordentlich ärgerlich abgelaufen, wäre ich dem Projekt ab hier nicht ganz so demoralisiert gegenüber gestanden.

Es hat mich viel Überwindung gekostet, mich trotzdem hinzuhocken und Bernd über die Schulter zu schauen um nicht völlig den Bezug zum Thema bzw. zu unserem Projekt zu verlieren.
Hier sehe ich auch den eigentlichen Anfang des Projektes da wir von hier an immer öfter gemeinsam im Raum 210 arbeiteten.

Wir hatten einen geregelten Sitzungsturnus und befanden uns stets im Zeitplan. Das Klima innerhalb des Teams war immer gut und es herrschte ein reger Gedanken- und Informationsaustausch, so dass die einzelnen Gruppenmitglieder (fast) immer über neu erarbeitete Aspekte informiert waren und so über den Lauf der Dinge bescheid wussten.

Der ungleiche Wissensstand bezüglich Linux hatte den Vorteil, dass diejenigen, die relativ ahnungslos bezüglich Linux waren, relativ schnell von 0 auf 30 kamen, da man sich gewisse Grundkenntnisse und Handhabungen schnell erklären und zeigen lassen konnte. Der Nachteil war, dass die Verlockung bei einigen Unklarheiten, einfach zu fragen wie etwas funktioniert, sehr groß gegenüber einer meist mühseligen Phase des „alleine Herausfindens“ war, von dem man doch am meisten profitiert.

Besonders überrascht hat mich wie lange man suchen muss, um ein bestimmtes Ereignis hervorzurufen, besonders wenn dies mit nur drei oder vier veränderten Zeilen geschehen kann.

Die Hauptarbeit wurde, wie schon in den Projekten zuvor, in der Intensivphase erledigt, so dass wir gegen Ende genügend Zeit hatten unser Skript in Ruhe durchzugehen und zu korrigieren.

Als abschließender Abschluss ist zu sagen, dass mich, nach dem niederschmetterndem Anfang, der Forscherdrang meiner Teammitglieder wieder selbst dazu anregte, ernsthaft an die Arbeit zu gehen.

Letzten Endes machte es auch Spaß, auch wenn der erhoffte Erfolg öfter mal ausblieb. Ich bin mal wieder ein bisschen schlauer geworden und an Erfahrung reicher und darum geht es ja.

Mein Fazit: Nie wieder Linux auf meinem Rechner!

PS: Das Buch habe ich verkauft.

6.3 Christian Keller

Am Anfang stand ich, genauso wie der Rest der Gruppe, vor einem Projekt, das vielversprechend klang und interessant zu werden versprach. Doch wie aller Anfang war auch dieser schwer, da ich wusste worum es ungefähr ging, aber keinen blassen Schimmer davon hatte, was mich wirklich erwartete.

Meiner Meinung nach ist das genau die richtige Ausgangsposition, um ein solches Projekt in Angriff zu nehmen. Wir alle standen vor einem Berg Informationen, den es abzuarbeiten galt.

Durch systematisches Erarbeiten des Sachverhaltes kamen wir unserem Ziel immer näher. Bald stand auch der Apache-Web-Server in seinen Grundfesten. Die Feinarbeiten, Fehlersuche und die Suche nach Lösungen für diverse Probleme nahmen aber die meiste Zeit in Anspruch.

Die Arbeit am Projekt war aus meiner Sicht spannend und aufregend, da ich immer wusste, dass ich im späteren Berufsleben an ähnlichen Projekten zu tun haben werde und unserer Projekt nahe bei der Praxis liegt.

Mich interessiert vor allem die praktische Arbeit mit alternativen Betriebssystemen, besonders jenen auf Unix-Basis und die Konfiguration von Systemen, die mit anderen Systemen kommunizieren müssen. Die Vertiefung meiner Kenntnisse mit Hilfe dieses Projekts kam mir sehr entgegen. Ich habe wieder viel dazugelernt und aus vielem meine Schlüsse gezogen.

Die Zusammenarbeit im Team war bis zuletzt hervorragend und ich bin im Nachhinein sehr froh über die Zusammenstellung unserer Projektgruppe. Vor allem möchte ich an dieser Stelle die Arbeit von Bernd Wissmeier hervorheben. Er hat über die ganze Zeit viel mehr Motivation gezeigt und mehr gearbeitet als ich selbst. Ihm gebührt von meiner Seite aus einen besonderen Dank.

Weiterhin möchte ich Herrn Scheib für seine gelegentliche Unterstützung danken. Leider haben wir seine Hilfe nicht allzu oft gebraucht, denn seine Lösungen für Probleme sind sehr erfrischend.

6.4 Bernd Wissmeier

Als die Projektgruppe sich für das Thema „Konfiguration eines Apache-Web-Servers unter Linux mit CGI Anbindung mittels PHP“ entschied, wusste ich, dass dieses ein sehr komplexes Thema werden würde. Zudem musste ich mich mit einem Betriebssystem auseinander setzen, mit dem ich bis dato recht wenig zu tun hatte. In eigener Regie installierte ich auf meinem privaten Computer SuSE Linux, um mich auf das Projekt vorzubereiten. Das war für mich die Voraussetzung, um ein wenig Praxis in die Sache zu bringen. Die Grundkenntnisse, die wir im Unterrichtsfach „Kommunikationstechnik-Linux“ vermittelt bekamen, konnte ich dabei sehr gut anwenden.

Da der Apache-Server für mich ebenfalls Neuland war, musste ich auch zu diesem Thema immens viel Zeit investieren. Zahlreiche Stunden im Projektraum waren die Regel. Da parallel noch Unterrichtsstoff zu bewältigen war, konnte ich das Projekt nicht nach meinem Geschmack ausweiten.

Die gleiche Situation bei dem Thema PHP. Es fehlte mir an Zeit, um mich nach meinen Wünschen mit der Materie auseinander zu setzen. Um richtig tief in den Stoff einzusteigen

Da sich das Projektthema als sehr aufwendig und umfangreich erwies, hat sich die Projektgruppe für einen modularen Aufbau der Themenanalyse entschieden. Das Thema wurde in die Bereiche Linux, Apache-allgemein, Apache-Konfiguration und PHP/Intranet unterteilt.

Der unterschiedliche Wissensstand, sowie die differierende Motivation machte es schwierig, das Projekt auf einem adäquaten Niveau zu halten. Am schwierigsten war das Zusammenfassen der Module, sodass jeder Projektteilnehmer am Schluss das gleiche Wissen aufweisen konnte.

Letztendlich bin ich mit dem Ergebnis unseres Projektes sehr zufrieden. Meine persönlichen Ziele sind fast erreicht (die Sicherheitsaspekte für Server, Userordner, Daten und Scripte konnten aus zeitlichen Gründen leider nicht mehr erarbeitet werden).

Ich verstehe die Funktionsweise eines Apache-Web-Servers der unter Linux läuft und dem User ein CGI zur Verfügung stellt. Als praktische Anwendung kann ich ein simples Gästebuch in PHP realisieren und eine kleine Datenbank in MySql erstellen.

7. Abschlusserklärung

Wir erklären, dass die Projektarbeit selbstständig und ohne fremde Hilfe verfasst wurde und keine anderen als die angegebenen Mittel verwendet wurden.

Wir versichern, dass alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche gekennzeichnet wurden.

_____

Thorsten Behrend

_____

Felix Hock

_____

Christian Keller

_____

Bernd Wissmeier