INSTITUT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Implementierung eines Werkzeuges zur Konsistenzprüfung von HTML-Links

Frank Schütz

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Alexander Keller
Abgabedatum:	25. April 1996

Inhaltsverzeichnis

Kapitel 1

Einführung

Das Internet hat in den letzten Jahren einen explosionsartigen Zugang an Benutzern erfahren. Dies liegt insbesondere an den Möglichkeiten des World Wide Web (WWW). Mit WWW ist es möglich, verteilte Daten in ansprechender Form zu präsentieren und miteinander zu verknüpfen. Jedermann kann von seinem WWW-Server auf Dokumente von anderen WWW-Servern verweisen. Dies führt zu Inkonsistenzen, wenn einer der Server ein Dokument löscht, da er nicht wissen kann, wer alles auf dieses Dokument zeigt. Gegenstand dieses Fortgeschrittenenpraktikums war es, ein Werkzeug zu entwickeln, welches einen WWW-Server auf Erreichbarkeit aller verwiesenen Dokumente überprüft. Es bleibt zu erwähnen, daß dieses Fortgeschrittenenpraktikum in Zusammenarbeit mit BMW gestellt wurde und unter PERL 5.0 zu implementieren war.

1.1 Die theoretischen Grundlagen des WWW's

Das WWW basiert auf einem Client-Server-Konzept. Ein WWW-Server ist dabei ein System, das Dokumente zur Verfügung stellt. Ein WWW-Client ist ein Programm, welches der Anwender benutzt, um von verschiedenen WWW-Servern die gewünschten Dokumente anzufordern und entsprechend darzustellen.
Der WWW-Client und der WWW-Server kommunizieren über eine TCP/IP-Verbindung. Dabei wird ein WWW-spezifisches Protokoll, das Hypertext Transfer Protocol (HTTP) verwendet, welches derzeit in Version 1.1 als Internet-Draft vorliegt [2]. Die Dokumente selbst müssen in Syntax der Hypertext Markup Language (HTML) vorliegen, zumindest wenn man von einem Dokument aus auf andere verweisen will [1].
Beim Anfordern eines Dokuments geschehen folgende Dinge: Der Client baut eine TCP/IP-Verbindung zum angegebenen Server auf (standardmäßig auf Port 80). Anschließend schickt er eine Anfrage im HTTP ab. Der Server beantwortet diese Anfrage, indem er entweder das gewünschte Dokument oder eine Fehlermeldung schickt. Daraufhin wird die Verbindung sofort abgebrochen. Wird für das Dokument noch zum Beispiel eine Grafik benötigt, muß diese gesondert angefordert werden. Es findet also erneut ein Verbindungsaufbau statt. Erst in neueren Ansätzen erlaubt man die Verbindung für eine begrenzte Zeit zu halten [4].

1.2 Die Hypertext Markup Language (HTML)

HTML liegt derzeit in Version 2.0 als Internet-Draft vor. HTML ist eine Anwendung der Standard Generalized Markup Language (SGML). Sie stellt eine einfache Sprache dar, die zum Aufbau von plattformunabhängigen Hypertext-Dokumenten dient. Dazu werden in den Orginaltext der einzelnen Dokumente sogenannte Tags (Marken) eingefügt. Bis auf wenige Ausnahmen treten Tags paarweise auf, d.h. ein Starttag und ein Endtag. Das durch das Starttag gesetzte Attribut gilt für den Text vom Starttag bis zum Endtag. Ein Starttag hat die Form `<MARKE>` und ein Endtag die Form `<\MARKE>`. Entfernt man in einem HTML-Dokument alle Tags, so erhält man den ursprünglichen Rohtext (gilt für den Body-Teil).
Die geforderte Struktur eines HTML-Dokuments schreibt eine Zweiteilung in einen Kopf- (Head) und einen Rumpfteil (Body) vor.

1.2.1 Der HTML-Kopfteil

Der Kopf eines HTML-Dokuments wird mit dem Starttag `<HEAD>` eingeleitet und mit dem Endtag `</HEAD>` beendet. Der Kopfteil wird in der Regel nicht vom WWW-Client angezeigt. Er enthält eine ungeordnete Sammlung von Informationen über das Dokument. Dazu gehört zum Beispiel der Titel (Text zwischen den Tags `<TITLE>`,`</TITLE>`), die Basisadresse für relative Links und andere Metainformationen. Weitere Beispiele für Metainformation sind der Autor, Datum der letzten Modifikation, E-Mail-Adresse des Verantwortlichen, usw. .

1.2.2 Der HTML-Rumpfteil

Der Rumpf eines HTML-Dokuments wird mit dem Starttag `<BODY>` eingeleitet und mit dem Endtag `</BODY>` beendet. Der Rumpfteil enthält den eigentlichen Dokumenttext. HTML bietet eine Reihe von möglichen Tags im Bodyteil. Grundsätzlich werden diese Tags dazu verwandt, die Textdarstellung zu beeinflussen und Verweise auf andere Dokumente einzufügen.

Bei den Verweisen bestehen die zwei Möglichkeiten, das verwiesene Dokument in das aktuelle Dokument einzufügen oder das verwiesene Dokument als mögliches Folgedokument anzubieten. Es sei darauf hingewiesen, daß der Name für Tags zur Beeinflußung der Textdarstellung oft irreführend eine Semantik „vorgaukelt“. So bedeutet das Tag `<ADDRESS>` nicht notwendigerweise, daß hier eine Adresse folgt, und auch das Format der möglicherweise folgenden Adresse ist in keinster Weise vorgegeben. Dieses Tag signalisiert dem WWW-Client nur, daß er eine andere Schriftform als für den Standardtext verwenden soll. Doch selbst dies ist nicht verpflichtend.

Für dieses Fortgeschrittenenpraktikum sind vor allem die Tags für Verweise auf weitere Dokumente von besonderem Interesse, da ja gerade deren Korrektheit und Erfüllbarkeit überprüft werden soll. Tags, die einen Verweis auf andere Dokumente enthalten können sind:
`<LINK>` Wird im Kopfteil benutzt um erforderlich Zusatzdokumente zu laden.
`<A>` An dieser Textstelle wird ein Verweis auf ein anderes Dokument eingefügt.
`<IMG>` An dieser Stelle wird ein weiteres Dokument eingefügt.
`<INPUT>` Läd Dokumente bzw. Bilder zu Eingabefeldern.

Alle diese Marken enthalten als ein Attribut einen Universal Resource Identifier (URI, beinhalten Uniform Resource Locators (URLs)), der ein weiteres Dokument bezeichnet. In diesem URI ist z.B. auch die Domäne enthalten [1]. Ein Teilproblem der Praktikumsaufgabe ist es, aus den vier oben angeführten Tags dieses Attribut, und speziell den URI zu extrahieren. Hier folgen nun einige Beispiele für oben erwähnte Tagtypen:

```
<A HREF=\"http://www.lfm.mw.tu-muenchen.de/sfb255-home.html\">
<IMG ALT=\"\" ALIGN=BOTTOM SRC=\"http:/images/logos/TUM_logo75.gif\">
```

Der extrahierte URI ist also im ersten Beispiel

```
http://www.lfm.mw.tu-muenchen.de/sfb255-home.html
```

und im zweiten

```
http:/images/logos/TUM_logo75.gif.
```

Kapitel 2

Implementierung

2.1 Aufgabenstellung

Die Aufgabe bestand darin, ein Werkzeug für die Konsistenzprüfung von HTML-Dokumenten zu entwickeln. Dabei sollte man mehrere zu durchsuchende Domänen angeben können. Zeigt ein Link auf ein Dokument außerhalb dieser Domänen, so soll noch die Erreichbarkeit dieses Dokuments überprüft werden. Eine weitere Verfolgung der Links auf diesem Dokument darf nicht stattfinden, da sonst die Gefahr bestünde, daß das ganze Internet durchsucht würde.
Wenn eine HTML-Seite einen Link auf ein nicht erreichbares Dokument enthält, so ist dem Verantwortlichen für diese HTML-Seite eine Mail zu schicken. Der Verantwortliche zu einer Seite ist auf der Seite vermerkt. Die Mail soll ausreichend Information enthalten, um den Fehler lokalisieren und beheben zu können.
Außerdem soll zusätzlich der Systemadministrator eine Aufstellung über alle Fehler erhalten.

2.2 Datenstruktur

Aus der Aufgabenstellung läßt sich unmittelbar die Datenstruktur ableiten. Man benötigt pro Dokument einen Datensatz, der die Felder Zugriffsmethode, Väter und den Betreuer beziehungsweise den Fehlercode speichert. Als Schlüssel für die Datensätze bietet sich der absolute URI des Dokuments an, zu dem die Infomation zu speichern ist, da dieser per Definiton eindeutig ist. Die Zugriffsmethode wird benötigt, da in der hier vorgesehenen Version nur Dokumente, auf die mit HTTP zugegriffen wird, zu überprüfen sind. Links die andere Zugriffsmethoden erfordern, werden ignoriert (FTP, eMail). Die Väter eines Dokuments A, also Dokumente, die einen Link auf das Dokument A haben, werden benötigt, da deren Betreuer benachrichtigt werden müssen, falls beim Zugriff auf Dokument A ein Fehler aufgetaucht ist. In dem letzten Feld wird entweder der Fehlercode oder der ermittelte Betreuer gespeichert. Dies ist machbar, da zu einem Dokument, welches nicht geladen werden kann, auch kein Betreuer festgestellt werden kann. Andererseits ist aber auch bei einem Dokument, zu dem sich der Betreuer ermitteln ließ, kein Fehler aufgetaucht.

Verwirklicht wurde diese Datenstruktur innerhalb des PERL-Skripts mittels eines assoziativen Arrays [7]. Charakteristisch ist hierfür, daß man als Index für den Array einen String benutzen kann. Dieser entspricht also genau unserem URI. Der Inhalt eines Arrayselements kann leider maximal wieder ein String sein. Deshalb wurden die einzelnen Felder mit einem „;“getrennt zu einem String konkateniert. Das Feld Väter entsteht dabei aus der Konkatenation der einzelenen URI's der Väter, getrennt durch „| “.

Insgesamt werden drei solche Arrays erstellt und verwaltet. Zum einen ist dies der Array NEU. Er enthält alle Links, die noch zu prüfen sind. Als zweites wird der Array ERL benutzt, der die erfolgreich getesteten Links enthält. Zu guter Letzt wird noch der Array ERR benötigt, der die Links aufnimmt, auf die nicht erfolgreich zugegriffen werden konnte.

2.3 Parameter

Explizite Eingaben, die ein in der Aufgabenstellung skizziertes Werkzeug braucht, sind einfach zu erkennen. Zum einen ist dies in jedem Fall eine Liste der zu durchsuchenden Domänen. Andererseits muß auch ein erstes Dokument angegeben werden, mit dem die Suche beginnt. Außerdem sollten folgende Parameter nicht durch die Programmierung fest „verdrahten“werden:

- Eine Liste von Dokumentendungen, die anzeigen, daß in einem Dokument mit einer dieser Endungen nicht nach Links gesucht werden muß, da hier keine weiteren Links enthalten sind. Beispiele hierfür sind .tif, .gif, .mpg, .ps und .jpg .
- Der Mailtext, der bei einem Fehler verschickt wird. Zusätzlich wird in diesem durch Platzhalter festgelegt, wo aktuelle Angaben zu aufgetretenen Fehlern einzusetzen sind.
- Die Mailadresse, die den Systemadministrator für die gesammelten Fehlermeldungen spezifiziert.

Zusätzlich wurden noch spezielle Parameter vorgesehen:

- Eine maximale Anzahl von Links, die überprüft werden sollen. Damit kann verhindert werden, daß bei einem „Ausbrechen“ aus den festgelegten Domänen ins Internet Links überprüft würden, bis dieser Mißstand auffällt.
- Einen regulären Ausdruck in PERL-Syntax, der die Marke spezifiziert, in der die Mailadresse des Betreuers zu finden ist.
- Einen regulären Ausdruck in PERL-Syntax, der die Mailadresse des Betreuers innerhalb der gefundenen Marke ermittelt.

Wegen der großen Anzahl von gewünschten Parametern wurde entschieden, das Skript durch eine Initialisierungsdatei gesteuert zu gestalten (vgl. Seite 32). Alle Paramter können dabei in der Initialisierungsdatei festgelegt werden.

2.4 Ausgaben

Als Ausgabe soll nach Aufgabenstellung eine Liste der aufgetretenen Fehler generiert werden. Diese Liste wird in eine sogenannte Log-Datei geschrieben. Somit besteht die Möglichkeit, die Auswertung der Fehler erst zu einem anderen Zeitpunkt durchzuführen. Es bietet sich auch sofort an, diese Log-Datei zu archivieren und durch Vergleich auf Korrektur der Fehler hin zu überprüfen.

Eine weitere Art der Ausgabe ist das Mailen der Fehlermeldungen. Dazu wird in den Parametern der Initialisierungsdatei der Text der Mail mit Platzhaltern definiert. Eine Aufgabe der Ausgabe ist es also noch diese Platzhalter durch die tatsächlichen Werte zu ersetzen.

Damit ist auch die Ausgabe spezifiziert.

2.5 Algorithmus

Das Skript besteht aus einem Initialisierungsteil, einer Schleife in der die einzelnen Dokumente geholt und nach neuen Links durchsucht werden und einem Ausgabeteil.

Im Initialisierungsteil wird die Initialisierungsdatei gelesen und die entsprechenden Variablen gesetzt. Speziell ist hier zu erwähnen, daß das Topdokument aus der Initialisierungsdatei in den Array NEU eingetragen wird. Somit enthält dieser genau ein Element.

Die Hauptaufgabe des Skripts wird in der Schleife erledigt. Zuerst wird das erste Element im Array NEU ermittelt und damit gleichzeitig überprüft, ob der Array leer ist. Ist dies der Fall, wird die Schleife beendet. Außerdem wird die Schleife noch beendet, wenn die Anzahl der Schleifendurchläufe, die Angabe über die maximal zu prüfenden Links überschreitet.
Es muß auch noch auf einen anderen wichtigen Sachverhalt hingewiesen werden: Die Reihenfolge, in der die Links abgeprüft werden ist völlig willkürlich. Sie hängt von der PERL-internen Verwaltung des assoziativen Arrays ab [7]. Dies spielt aber keine Rolle, da eine Domaine sowieso fast immer komplett durchsucht werden soll.
Nachdem nun ein zu prüfender Link ermittelt wurde, wird entschieden, ob eine Anfrage mittels der Methode HEAD ausreicht, oder ob das Dokument komplett mittels der GET-Methode geholt werden muß. Die Methode HEAD reicht aus, wenn sich der Link auf einem nicht zu durchsuchenden WWW-Server befindet, oder wenn sich die Endung in der Liste, der nur mit der HEAD-Methode zu prüfenden Links befindet. Charakteristisch für die HEAD-Methode ist, daß nur der Kopf des Dokuments übermittelt wird. Man bekommt also nur die Meldung, daß dieses Dokument erreichbar ist. Da man den Inhalt solcher Dokumente nicht kennt, kann man hiermit aber keine neuen Links ermitteln. Der Vorteil der HEAD-Methode ist die geringe Netzbelastung und die kurze Übertragungszeit.
Erkennt das Skript die HEAD-Methode als ausreichend, so wird das Unterprogramm *HoleKopf* aufgerufen, anderenfalls das Unterprogramm *HoleLinksZuEinerSeite*. Diese Unterprogramme liefern einen assoziativen Array zurück, der das gleiche Format hat, wie die Arrays NEU, ERL, ERR. In diesem Array befinden sich die Links, die in dem Dokument gefunden werden konnten (bei GET-Methode). Außerdem enthält der Array, falls das Dokument nicht erreichbar war, einen Eintrag „FEHLER “. Unter diesem Eintrag wird der Fehlercode gespeichert. Falls das Dokument erreichbar war und ein Betreuer ermittelt werden konnte, exisitert noch der Eintrag „BETREUER “. Unter diesem Eintrag wird die eMail-Adresse des Betreuers übergeben.

Die nächste Aufgabe des Skripts ist also, zu prüfen ob ein Fehler auftrat. Ist dem so, wird der Datensatz mit Fehlercode in den Array ERR geschrieben.
Trat kein Fehler auf, so wird der Datensatz zum geprüften Link mit eMail-Adresse des Betreuers in den Array ERL geschrieben. Nun wird eine Funktion aufgerufen, die die in dem Dokument gefundenen Links in den Array NEU aufnimmt. Dabei wird darauf geachtet, daß sich ein Link nicht schon in einem der drei Arrays befindet. Dies ist unbedingt erforderlich, um Zyklen auszuschließen.

Nachdem nun noch der gerade getestete Link aus dem Array NEU gestrichen wurde, ist ein Schleifendurchlauf beendet.

Was ist also mit den Arrays geschehen? Ein Eintrag aus dem Array NEU wurde entweder in den Array ERR oder in den Array ERL verschoben, je nachdem ob ein Fehler beim Prüfen des Dokuments aufgetreten ist oder nicht. Außerdem wurden neu entdeckte Links in den Array NEU eingetragen.

Der Ausgabeteil am Schluß sichert zuerst die drei Arrays NEU, ERR, ERL mit einem

Kopf in die drei Log-Dateien LinkTest.NEU.LOG, LinkTest.ERR.LOG, f.ERL.LOG. Dabei wird jeder Datensatz in übersichtlicher Darstellung und in interner Darstellung eingetragen. Die interne Darstellung wird von den zwei Zusatzskripten benötigt. Außerdem wird zu jeder Sitzung auch eine Log-Datei LinkTest.log erstellt. In dieser Datei werden die an die jeweiligen WWW-Server verschickten Requests mitprotokolliert. Nach Erstellung der Log-Dateien wird, falls Mails erwünscht sind, das Unterprogramm zum Verschicken der Mails aufgerufen.

Nach einem kleinen Hinweis auf die beiden Zusatzskripten ist das Programm beendet. Die Beziehungen der erstellten Skripten untereinander und zur Umgebung stellt die Abbildung auf Seite 10 dar.

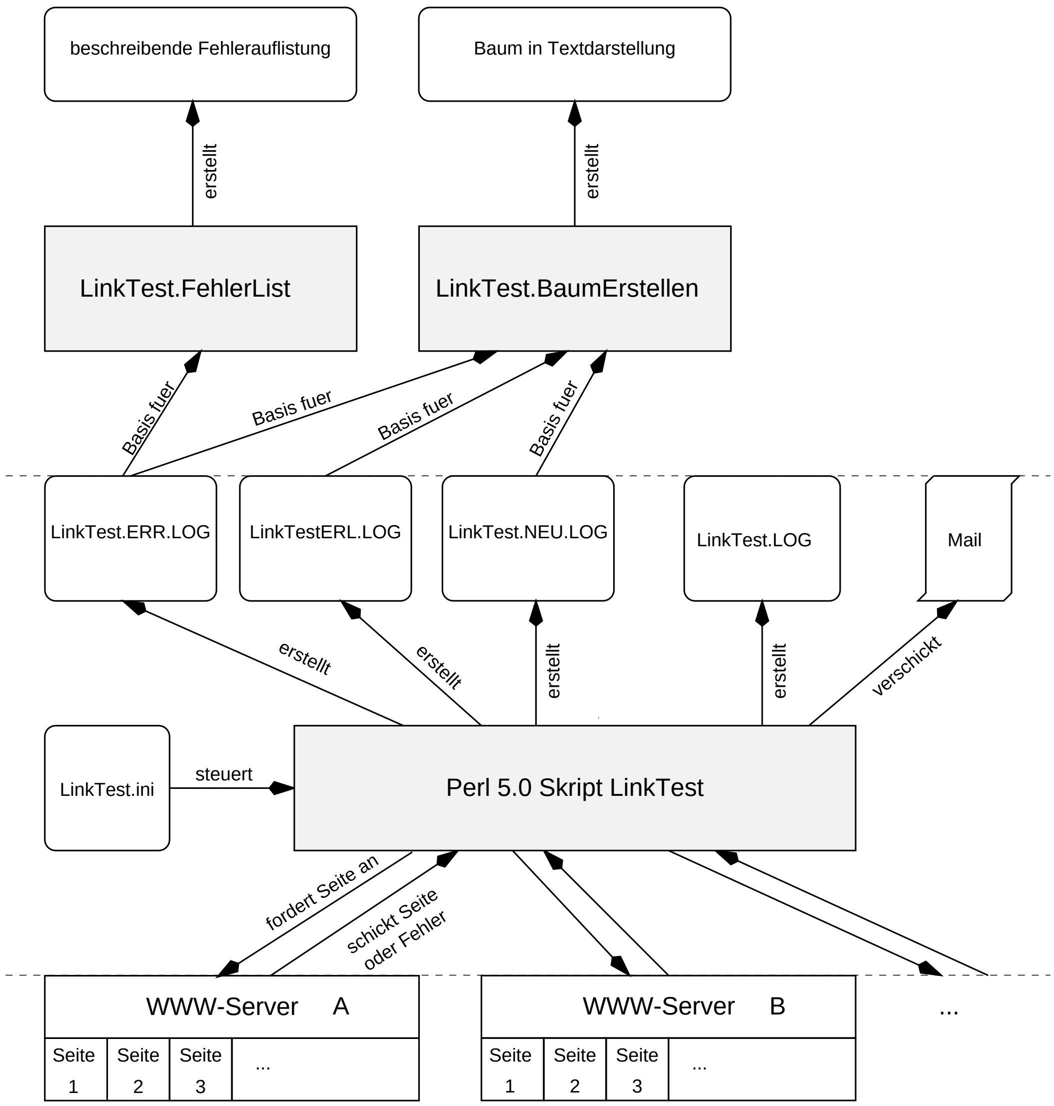

Abbildung 2.1: Beziehungen zwischen den Skripten und der Umgebung

Kapitel 3

Die Skriptenreihe im Einsatz

3.1 Initialisieren der Konfigurationsparameter

Vor dem Starten des Skripts, sind in der Initialisierungsdatei LinkTest.ini die gewünschten Parameter einzugeben. Als erstes werden die Domänen angegeben, die durchsucht werden sollen (durch Komma getrennt). Wenn man zum Beispiel wünscht, daß die Domänen www.tu-muenchen.de und www.informatik.tu-muenchen.de durchsucht werden, so gibt man in der Initialisierungsdatei

```
[WWW_SERVER]=www.tu-muenchen.de,www.informatik.tu-muenchen.de;
```

ein. Besonders wichtig ist dabei, daß keine Leerzeichen in der Zeile eingegeben werden. Auch der abschließende Strichpunkt darf nicht vergessen werden. Das Programm beginnt mit der Kontrolle der ersten Domäne und weitet die Kontrolle auf die weiteren Domänen nur aus, wenn Links von der ersten darauf bestehen. Dies liegt daran, daß das Skript zu Beginn nur von der ersten Domäne ein Dokument kennt.

Als nächstes wird das Dokument angegeben, mit welchem die Suche beginnen soll. Dies wird in den meisten Fällen `/` sein. Der Eintrag in der Ini-Datei sieht dazu wie folgt aus: `[TOP_LINK]=/;`

Der nächste Eintrag ist die Angabe der maximal zu testenden Dokumente. Dieser Eintrag soll als letzte Sicherheit verhindern, daß das Programm auf ungewünschten Servern des Internets stunden- bzw. tagelang sucht. Man sollte hier also eine Grenze angeben, die größenordnungsmäßig mit der Anzahl der Dokumente auf den gewünschten Servern übereinstimmt. Angenommen es befinden sich auf dem zu durchsuchenden Server 800 Dokumente, so könnte der Eintrag der Ini-Datei beispielweise so aussehen:

```
[MAX_LINKS]=1000;
```

Um die Kontrolle der Links effizienter zu gestalten, kann man in dem Feld `[ENDUNGEN]` der Initialisierungsdatei Endungen von Dokumenten angeben bei denen nur eine Head-Anfrage durchgeführt werden soll. Das heißt, anstatt die ganze Datei zu übertragen, wird nur ein Kopf mit Informationen zu dieser Datei übertragen. Dies entlastet das Netz besonders bei Bildern sehr stark und liefert bedeutend bessere Ausführungszeiten. Ein Beispieleintrag könnte somit wie folgt aussehen:

```
[ENDUNGEN]="gif","tif","jpg","GIF","TIF","JPG","mpg","MPG","ps";
```

Damit würden dann zur Kontrolle eines MPEG-Files nicht mehrere Kilobyte bis Megabyte, sondern nur die wenigen Bytes des Kopfes übertragen. Eine Einstellung ähnlich der oben vorgestellten, ist also unbedingt anzuraten.

Ist in jeder HTML-Seite die eMail-Adresse eines Verantwortlichen für diese Seite eingetragen, so gibt es in der Ini-Datei zwei Felder, um anzugeben, wie daß Programm diese Adresse ermitteln kann. Nehmen wir an, es wurde festgelegt, in den Kopf jedes HTML-Dokuments folgende Zeile einzutragen:

```
<META NAME="AUTHOR" VALUE="emailadresse des Autors">
```

Dann würde man mit dem Eintrag `[BETREUER_TAG]='<META NAME="AUTHOR"';` dem Programm die Möglichkeit geben, die Marke, welche den Betreuer enthält, zu erkennen. Dabei entspricht der Eintrag in der Ini-Datei einem regulären Ausdruck in PERL 5.0 Syntax. Das Programm vergleicht diesen Ausdruck mit jeder Marke im HTML-Dokument. Wird eine Übereinstimmung entdeckt, so wird diese Marke einem zweiten Test unterzogen. Dieser zweite Test benutzt den zweiten, im Feld `[BETREUER_ADRESSE]` angegebenen regulären Ausdruck, um aus der Marke den Betreuer zu extrahieren. Der Ausdruck muß dabei so gestaltet sein, daß als Platzhalter für die Adresse folgende Kombination von Ausdrücken steht: `(.*?)` . Diesen Platzhalter darf man insgesamt zweimal verwenden. In unserem obigen Beispiel wäre also folgender Eintrag in die Ini-Datei nötig: `[BETREUER_ADRESSE]='VALUE="(.*?)"';` . Wie zu erkennen ist, müssen die regulären Ausdrücke mit ' geklammert werden, um eine interne Ersetzung durch die Shell oder PERL zu verhindern(vgl. [7]).

Soll außer den Betreuern noch eine Person eine gesammelte Auflistung der Fehler erhalten, so ist deren Adresse wie folgt anzugeben:

```
[NACHRICHT_AN]=schuetzf@informatik.tu-muenchen.de;
```

Dies könnte zum Beispiel der Webmaster oder eine vorgesetzte Person sein. Es ist nur möglich, eine eMail-Adresse anzugeben.

Die letzten beiden Felder beziehen sich auf die zu verschickende Nachricht. Das erste Feld ist `[NACHRICHT]` und kann folgende Werte annehmen:

- KEINE : Es wird keine Nachricht generiert.
- DEBUG : Es werden Nachrichten generiert. Diese werden aber nicht weggeschickt, sondern in spezielle Files ausgegeben. Dabei wird die Mail mit der gesammelten Auflistung in die Datei LinkTest.NACHRICHT und die jeweiligen Mails an die Betreuer zusammengefaßt in der Datei f.NACHRICHT.BETREUER gespeichert.
- SENDE : Die Mails werden an die entsprechenden Personen (falls bekannt) verschickt.

In das zweite Feld wird der Text der zu versendenden Nachricht eingegeben. Dabei besteht die Möglichkeit, Platzhalter für gewisse Informationen zu benutzen. Folgende Platzhalter sind erlaubt:

- {{Datum}}: Dieses Feld wird mit dem Datum des Prüflaufes, bei dem der Fehler entdeckt wurde, ersetzt.
- {{Link}}: Anstelle dieses Platzhalters wird im jeweiligen Mail der URI des nicht erreichten Dokuments ausgegeben.

- {{Seite}}: In der Mail wird dieser Platzhalter durch den URI der Seite ersetzt, auf der sich der nicht erreichte Link befindet.
- {{Zeile}}: Dieser Platzhalter wird durch die Nummer der Zeile ersetzt, in der sich der nicht erreichte Link auf der Seite befindet.
- {{Fehlercode}}: Hier trägt das Programm im Mailtext den erhaltenen Fehlercode ein.
- {{FehlerLang}}: Dieser Platzhalter wird durch eine ausführliche, textuelle Erklärung des Fehlers ersetzt.

Sollen sich an bestimmten Stellen Tabulatoren oder Seitenumbrüche befinden, so sind diese an den entsprechenden Stellen einzugeben. Die gesamte Nachricht muß mit „<<“und „>>“geklammert werden.

Nachdem nun die Initialisierungsdatei den Gegebenheiten entsprechend angepaßt wurde, kann das Programm gestartet werden. Das Programm geht dabei davon aus, daß die Initialisierungsdatei LinkTest.ini heißt. Wurde ein anderer Name vergeben, so muß dieser Name als Parameter beim Aufruf des Skripts angegeben werden.

Nachdem das Programm gestartet wurde, kontrolliert es die Links, bis alle überprüft wurden oder die eingestellte Grenze der maximal zu testenden Links erreicht wurde. Anschließend verschickt es gegebenenfalls die Mails. Am Ende zeigt das Programm noch an, wie viele Links als erreichbar erkannt wurden, und bei wie vielen ein Fehler auftrat. Außerdem wurden einige Log-Dateien erstellt, die Auskunft über den Ablauf und die gewonnen Ergebnisse geben. Zur weiteren Verarbeitung empfiehlt es sich, die zwei Zusatzskripten f.FehlerList und LinkTest.BaumErstellen zu benutzen.

3.2 Skripten zur besseren Darstellung der Ergebnisse

Da die bloße Ausgabe in Log-Dateien doch etwas spartanisch wirkt, wurden noch zwei kurze PERL-Skripts zur Darstellung der Ergebnisse geschrieben.

3.2.1 Ausführliche Fehlerauflistung

Die ermittelten fehlerhaften Links verdienen sicherlich ein spezielles Augenmerk. Deshalb wurde das kurze Skript LinkTest.FehlerList entwickelt. Dieses Hilfsmittel erwartet als optionalen Parameter den Namen der Fehler-Log-Datei. Wird dieser weggelassen, wird nach der Standard-Fehler-Log-Datei gesucht und diese benutzt.

Das Programm listet die Information der Log-Datei übersichtlich auf. Dabei wird zu jedem Fehlercode eine zusätzliche Einordnung und Erklärung gegeben. Die Ausgabe dieses Programms sieht zum Beispiel wie folgt aus:

```
Link      : www11.informatik.tu-muenchen.de/local/lectures/prakt_uebersicht.html
Vaeter    : www.informatik.tu-muenchen.de/stud_info/ws1995/prakt.html Zeile:48
Fehlercode: 404 Not Found
Kategorie : Client-Fehler
Klartext  : Seite konnte unter dieser Adresse nicht gefunden werden.

Link      : www.buxtehude.de/images/icons/info/abac_doc.gif
```

```
Vaeter    : www.informatik.tu-muenchen.de/ Zeile:66
Fehlercode: 602 (intern), Connect failed
Kategorie : Fehler waehrend Skriptausfuehrung
Klartext  : Es war nicht moeglich, von diesem Rechner aus eine Verbindung
            zu dem, der Domain entsprechenden Server aufzubauen.
```

3.2.2 Erstellen eines Link-Baumes

Ein weiteres Hilfsmittel, welches auf Wunsch der BMW AG erstellt wurde, ist ein einfaches Skript zur Darstellung der Verknüpfungen zwischen den Dokumenten. Dabei ist natürlich klar, daß eine ausführliche und genaue graphische Darstellung einen erheblichen Aufwand bedeutet hätte. Es wurde sich deshalb darauf geeinigt, daß ein Baum in Textform, ausgedrückt durch Einrückungen, als Darstellung ausreicht.

Daß es sich bei der Struktur der Linkverbindungen zwischen den Dokumenten nicht um einen Baum handelt, ist ebenfalls klar. Die Struktur wurde zu einem Baum vereinfacht, indem bei einem Verweis auf ein Dokument, welches schon im Baum ausgegeben wurde der Vermerk „Zeilennummer“angefügt wurde. Zeilennummer steht dabei für die Zeile im Baum, in der Ast zum ersten Mal auftauchte. Will man also diesen speziellen Ast weiterverfolgen, so kann man in dieser Zeile weiterlesen. Dadurch werden Zyklen verhindert.

Das Skript besitzt zwei optionale Kommandozeilenparameter:

- Maximale Zeilenlänge

  Hier gibt man die maximale Zeilenlänge, die ausgegeben werden soll Anzahl in Zeichen an. Würde eine Zeile länger als die angegebene maximale Zeilenlänge, so wird sie vom Programm gekürzt. Dies wird durch Anhängen von „`-->` “markiert.

- Gewünschte Wurzel

  Falls man an den Verknüpfungen, ausgehend von einem speziellen Dokument interessiert ist, kann man als zweiten Kommandozeilenparameter das Wurzeldokument angeben, von wo aus der Baum erstellt werden soll. In diesem Fall ist der erste Kommandozeilenparameter, die maximale Zeilenlänge, explizit anzugeben.

Als Ausgabe erhält man einen Baum, der zum Beispiel wie folgt aussieht:

```
Logdateien einlesen: ...Fertig
1www.informatik.tu-muenchen.de/
2|   www.informatik.tu-muenchen.de/images/icons/info/abac_post_d.gif
3|   www.informatik.tu-muenchen.de/admin/www-people.html
4|   |   www.leo.org/~loevenic/loe_1.gif
5|   |   www.leo.org/~fuhrmann/bea_s2.gif
6|   |   www.leo.org/~stumpf/stumpf_s.gif
7|   |   www.leo.org/~gruner/gruner_s.gif
8|   |   wwwpaul.informatik.tu-muenchen.de/personen/loevenic.html
9|   |   www.informatik.tu-muenchen.de/images/icons/info/abac_post_d.gif 2
...
```

3.3 Bewertung

Nach der ausführlichen Beschreibung sollen hier einige Leistungsmerkmale besonders herausgehoben werden:

- Das Programm ist äußerst einfach zu konfigurieren. Es sind nur einige Eintragungen in der Initialisierungsdatei erforderlich.
- Wegen der Verwendung einer Initialisierungsdatei ist nur einmal eine Konfiguration nötig.
- Die Möglichkeit verschiedene Initialisierungsdateien beim Start anzugeben, erlaubt für verschiedene Zwecke unterschiedliche Initialisierungsdateien.
- Das Skript LinkTest.FehlerList gibt eine ausführliche Erklärung zu jedem Fehler aus.
- Das Skript LinkTest.BaumErstellen erstellt einen Baum der Struktur des durchsuchten WWW-Servers.
- Das Programm erfordert keine Root-Berechtigung. Es reicht normaler Internet-Zugriff auf den zu durchsuchenden WWW-Server.
- Das Skript durchsucht den kompletten WWW-Server www.informatik.tu-muenchen.de in 35 Minuten (1426 Links)
- Möglichkeit Endungen von Dokumenten anzugeben, bei denen eine HEAD-Anfrage ausreichend ist.
- Mit zwei regulären Ausdrücken in PERL-Syntax äußerst flexible Suche nach dem Betreuer.
- Die Skripten sind kurz und ohne unnötigen Ballast. MomSpider hingegen benötigt Libraries und ist auf viele Files verteilt.
- Flexible Mails an Betreuer und Vorgesetzten.
- Plattformunabhängig, da in PERL geschrieben und dieses ein Interpreter ist.

Für die Zukunft könnte man sich noch einige Verbesserungen vorstellen. Dazu zählen folgende Punkte:

- Parallelisieren der HEAD-Anfrage.

  Da bei HEAD-Anfragen nur der Erfolg wichtig ist, kann man diese sehr einfach parallelisieren. Alle Anfragen, auch die GET-Anfragen parallelisieren bringt keine Vorteile, da ein WWW-Server eine maximale Zahl von Anfragen gleichzeitig beantwortet. Diese Zahl liegt typischerweise bei fünf. Der Aufwand des parallelen Aufnehmens von neu entdeckten Links in den Array NEU ist somit nicht gerechtfertigt.

- Bessere Unterstützung bei Standardfehlern.

  Wenn man etwas Erfahrung mit dem Programm gesammelt hat, erkennt man vielleicht, wie charakteristische Fehler in der Ausgabe des Programms erkennbar sind und wie diese vielleicht sogar teilautomatisch behoben werden können. Zum Beispiel fiel bei der Kontrolle des Servers www.informatik.tu-muenchen.de auf, daß des öfteren das schließende Anführungszeichen bei Angaben von URI's vergessen wurde. Dies erkennt man ganz charakteristisch in der Ausgabe des Programms.

- Eine Anbindung an eine Managementumgebung.

Da der letzte Punkt dieser Auflistung für sehr wichtig erachtet wurde, wurde das Fortgeschrittenenpraktikum dahingehend erweitert. Man entschloß sich, einen DPI-Subagent zu entwickeln. Dieser sollte auch in PERL 5.0 implementiert werden, um die Portabilität des Skripts zu erhalten.

Kapitel 4

Entwicklung eines DPI-Subagenten in PERL 5.0

Aus der ersten, gelösten Aufgabenstellung erwuchs eine neue, die Implementierung eines DPI-Subagenten in PERL 5.0 zur Steuerung und Abfrage des Skripts LinkTest. Dieser Subagent soll soweit modular als möglich konzipiert sein, damit die Erstellung von weiteren DPI-Subagenten unter PERL 5.0 einfach und schnell wird. Dazu muß man sich aber zuerst den generellen Ablauf einer Sitzung mit einem DPI-Subagenten vor Augen führen.

4.1 Genereller Ablauf bei SNMP-DPI-Kommunikation

Grundsätzlich beinhaltet die Welt, in der sich die weiteren Ausführungen bewegen drei verschiedene Komponenten: einen Manager, einen Agenten (eventuell auch mehr) und einen Subagenten (auch hier eventuell mehr). Zuerst betrachten wir hier nur den Manager und den Agenten. Das Wesen des Agenten ist es, Information in Form von Variablen bereitzustellen. Diese Variablen können vom Manager abgefragt werden. Dies geschieht in unserem Fall mittels SNMP (Simple Network Management Protocol). Welche Variablen von einem Agenten angeboten werden und welchen Typ sie besitzen, erfährt der Manager aus der speziellen MIB zu jedem Agenten. Typischerweise kann ein Manager mehrere Agenten abfragen.

Da dieses Verfahren bis hierher recht statisch ist, versucht man das Ganze etwas flexibler zu gestalten. Man führt Subagenten ein. Diese wiederum bieten auch Variablen mit Informationen an. Sie werden aber nicht direkt vom Manager abgefragt, sondern erweitern den Leistungsumfang eines Agenten. Wenn ein Manager also eine Variable eines Subagenten abfragen will, bittet er den Agenten darum. Dieser leitet die Anfrage über DPI an den Subagenten weiter. Es geht dabei sogar soweit, daß der Manager nicht weiß, ob die Variable vom Agenten oder von einem Subagenten stammt. Da sich Subagenten sehr einfach während der Laufzeit an einen Agenten „anhängen" lassen, wird der Leistungsumfang eines Agenten dynamisch erweiterbar [3].

Angenommen der Manager und der Agent laufen beide und stehen schon in Verbindung miteinander und man will einen Subagenten zum Agenten anhängen. Dann muß der Subagent zuerst vom Agenten erfahren, auf welchem Port dieser bereit ist, eine DPI-Verbindung aufzubauen. Dazu schickt er ein spezielles SNMP-Packet an den Agenten. Dieser gibt einen Port zurück, auf dem die weitere Verbindung unter DPI stattfinden soll.
Auf diesem Port stellt nun der Subagent zum Beispiel eine TCP-Verbindung zum Agenten

her. Anschließend verschickt der Subagent ein OPEN-Request unter DPI auf diesem Port. Damit übermittelt er den Wunsch, als Subagent unter dem Agenten „dienen " zu dürfen. Er übermittelt mit diesem Open-Packet auch einige Parameter für die weitere Kommunikation. Dazu gehört zum Beispiel eine Time-out-Grenze, ein Maximum auf einmal anfragbarer Variablen, usw. .
Wenn der Agent das OPEN-Request positiv beantwortet, muß sich der Subagent registrieren, um weitermachen zu können. Registrieren heißt, er gibt an, welche Variablen er zur Verfügung stellt. Dazu meldet der Subagent die Wurzel des Baums der von ihm verwalteten Variablen.

Wurde auch das REGISTER-Request angenommen, so ist der Initialisierungsteil abgeschlossen. Der Subagent wartet nun typischerweise auf Anfragen des Agenten, die dieser wiederum vom Manager gestellt bekommen hat. Das heißt also, daß der Subagent ab jetzt nur noch mit dem Agenten kommuniziert. Der Agent wiederum dient nur als Verteiler der Nachrichten des Managers und des Subagenten. Deshalb wird in der weiteren Ausführung nur die Kommunikation zwischen dem Agenten und dem Subagenten betrachtet.
Anfragen, die auf den Subagenten zukommen können sind:

- ein GET-Request. Das GET-Request bedeutet, daß der Agent den Wert einer Variablen lesen möchte. Wenn der Subagent für diese Variable verantwortlich ist, muß er den Wert in einem Response-Paket zurückgeben. Falls er nicht verantwortlich ist, schickt er ein Error-Paket zurück.

- ein GETNEXT. Beim GETNEXT-Request bezeichnet der Agent, wie beim GET-Request eine Variable. Im Unterschied zum GET-Request interessiert er sich aber nicht für den Wert dieser Variablen, sondern für die Bezeichnung und den Wert, der in der MIB-Hierarchie nächsten Variable. Gibt es in der MIB-Hierarchie keine nächste Variable mehr, so wird kein Fehler, sondern ein EndOfMib als Typ der Variable versand. Diese Methode ist sinvoll, um sich durch eine unbekannte MIB zu „hangeln " oder um um Tabellen zu lesen.

- ein SET-Request. Ein SET-Request gibt an, daß der Agent den Wert einer Varibalen vom Subagenten setzen möchte. Zu diesem Zeitpunkt soll die Variable aber noch nicht gesetzt werden, obwohl ihr Wert schon mitgeliefert wird. Der Subagent soll nur schon alle Vorkehrungen für ein schnelles Setzen treffen. Wenn der Subagent die Vorkehrungen getroffen hat, muß er den Erfolg mittels eines Response-Pakets an den Agenten melden. Anderenfalls muß er ein Error-Paket an den Agenten schicken.

- ein UNDO-Request. Ein UNDO-Request zeigt an, daß der Agent die angekündigte Änderung doch nicht durchführen will. Der Subagent soll alle getroffenen Vorkehrungen rückgängig machen. Hat der Subagent dies erfolgreich durchgeführt, so muss er ein erfolganzeigendes Response-Paket an den Agenten schicken. Anderenfalls sendet der Subagent ein Error-Paket.

- ein COMMIT-Request. Bei einem COMMIT-Request muß der Subagent die Variable setzen. Ist dies nicht erfolgreich, so ist unbedingt ein Error-Paket an den Agenten zu verschicken. Bei Erfolg wird ein dementsprechendes Response-Paket übermittelt.

Der etwas umständlich anmutende Mechanismus zum Setzen einer Variablen hat durchaus seine Berechtigung. Man muß sich vor Augen führen, daß der Manager in einer geschlossenen Transaktion nicht nur eine Variable setzen will, sondern eventuell mehr und dies noch bei verschiedenen Agenten bzw. Subagenten. Wenn nun bei einem Agenten bzw. Subagenten ein Problem auftaucht, die anderen aber alle die Variablen setzen würden, könnte es zu Inkonsistenzen kommen. Dies versucht man, durch Aufteilung in eine Ankündigung und ein Durchführen zu verhindern.

Der Subagent kann nach dem Initialisierungsteil nicht nur Nachrichten vom Agenten empfangen, sondern er muß ihm auch welche schicken. Dazu gehört wie oben gesehen die Response-Meldung. Diese wird, bis auf wenige Ausnahmen, immer als Antwort einer Anfrage des Agenten verschickt. Der Subagent hat aber auch eine Möglichkeit eine Meldung an den Agenten zu schicken, ohne daß eine Anfrage notwendig war. Dies sind sogenannte TRAP's. Mit einem TRAP kann ein Subagent zum Beispiel einen Zustandswechsel an den Agenten, und damit an den Manager melden.

Wird der Subagent beendet, so schickt dieser ein UNREGISTER-Request an den Agenten. Hat dieser das UNREGISTER-Request angenommen, so schickt der Subagent ein CLOSE-Request. Da es auf ein CLOSE-Request keine Antwort gibt, kann sich der Subagent nun ohne zu warten beenden. Der Agent würde es auch mitbekommen, wenn der Subagent anderweitig beendet wird (KILL, Ctrl-C, Absturz) [8]. Im Zuge einer sauberen Programmierung, ist die Variante eines kontrollierten und definierten Ausstiegs aus der Kommunikation aber vorzuziehen.

Die Umgebung, in die der erstellte Subagent eingebunden ist, wird in der Abbildung auf Seite 20 dargestellt.

4.2 Die erstellte Funktionenbibliothek

Nach diesem kurzen Einblick in die DPI-Subagenten-Problematik, möchte ich jetzt, die sich daraus ergebenden, implementierten Funktionen kurz erklären. Vorneweg muß ich erwähnen, daß manche Funktionen keine Parameter besitzen, sondern benötigte Werte aus globalen Variablen beziehen. Dies ist zwar für die meisten Programmierer etwas „unsauber“, entspricht aber mehr dem Wesen von PERL. Falls eine Funktion eine solche globale Variable benötigt, wird dies angegeben.

4.2.1 qDPIport()

qDPIport() (query DPI Port) ermittelt über eine SNMP-Verbindung zum Agenten unter UDP den Port, auf dem der Agent bereit ist, eine DPI-Verbindung zu einem Subagenten aufzunehmen. Parameter besitzt diese Funktion keine. Sie geht davon aus, daß sich der Name des Rechners, auf dem der Agent läuft in der globalen Variablen $AGENT_HOST befindet.
Als Rückgabe liefert sie die Nummer des Ports, auf dem die weitere Kommunikation stattfinden soll.
Sollte der Verbindungsaufbau fehlschlagen, wird das komplette Skript beendet.

4.2.2 TCPConnect()

TCPConnect() baut auf dem angegebenen Port eine TCP-Verbindung zum Rechner, der in der globalen Variablen $AGENT_HOST eingetragen ist auf. Als Parameter wird der Port, auf dem die Verbindung aufgebaut werden soll, benötigt.

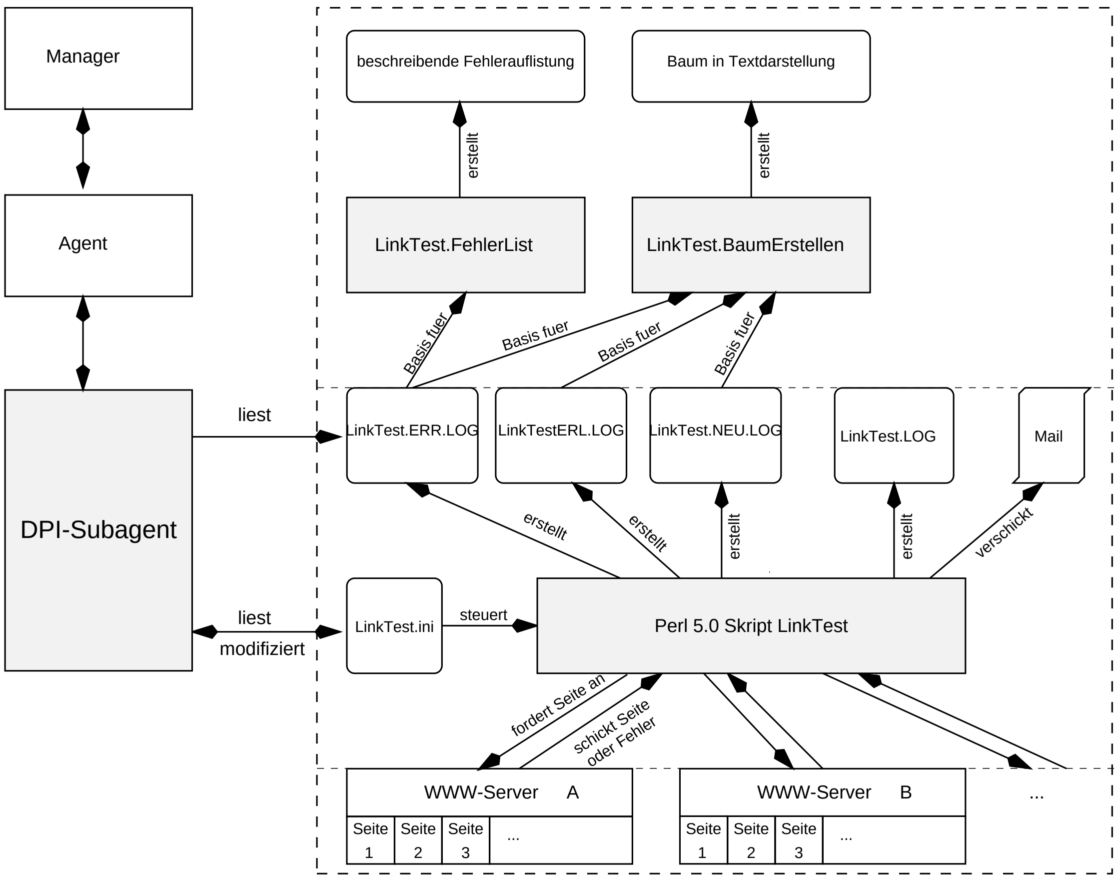

Abbildung 4.1: Beziehungen des Subagenten zu seiner Umgebung

Nachdem die Funktion erfolgreich beendet wurde, existiert ein PERL 5.0 Filehandle namens „DPI_Anschluß " für die weitere Kommunikation.

4.2.3 DPI_Open()

Diese Funktion dient dazu, eine Verbindung mit dem DPI-fähigen SNMP-Agenten aufzubauen. Sie erstellt ein DPI-Open-Request Paket und verschickt dieses an das Filehandle namens DPI_Anschluß. Die Funktion erwartet dabei den OID (Object Identifier), den der Subagent erhalten soll, in der globalen Varibalen $OID. Außerdem wird dem Agenten eine kurze Erklärung zum Subagenten, die sich in der globalen Variablen $DESCRIPTION befindet, übermittelt. Wird die Anfrage durch ein Response-Paket vom Agenten positiv beantwortet, so wird hinter den Aufruf der Funktion zurückgekehrt. Anderenfalls wird das gesamte Skript mit einer Fehlermeldung beendet.

4.2.4 DPI_Register()

Mit dieser Funktion registriert man den Teilbaum, für den der Subagent verantwortlich sein soll. Dazu wird dem Agenten in einem DPI-Register-Request Paket die Wurzel des Teilbaums

übermittelt.
Diese Funktion erwartet die Wurzel (auch Group-ID genannt) in der globalen Variablen $GROUPID. Wird die Registrierung durch ein Response-Paket vom Agenten positiv beantwortet, so wird hinter den Aufruf der Funktion zurückgekehrt. Anderenfalls wird das gesamte Skript mit einer Fehlermeldung beendet. Der Subagent meldet sich dazu vorschriftsmäßig vom Agenten mit einem DPI_Close ab.

4.2.5 DPI_UnRegister()

DPI_UnRegister() meldet den zuvor registrierten Teilbaum beim Agenten ab. Dazu benötigt diese Funktion die Wurzel des abzumeldenden Teilbaums (Group-ID) in der globalen Varibalen $GROUPID. Die Funktion wartet darauf, daß der Agent auf die Abmeldung des Teilbaums mit einem Response-Paket reagiert. Signalisiert dieses Paket Erfolg, so wird die Funktion normal beendet. Trat ein Fehler auf, oder weigert sich der Agent die Abmeldung anzunehmen, wird ein DPI_Close verschickt und das gesamte Skript mit einer Fehlermeldung beendet.

4.2.6 DPI_Close()

Die Funktion DPI_Close() meldet den Subagenten beim Agenten ab. Diese Funktion erwartet keine Parameter und wartet nicht auf eine Antwort des Agenten. Dies entspricht dem RFC 1592.

4.2.7 ResponseError()

ResponseError() verschickt ein Fehler-Paket an den Agenten. Die Funktion erwartet dabei als Parameter die Paket-ID des Pakets, welches den Fehler verursachte und einen Fehlercode nach Tabelle 18 im RFC 1592.

4.2.8 ResponseSuccess()

Hat der Agent den Wert einer Variablen des Subagenten angefordert, so wird dieser mit der Funktion ResponseSuccess() an den Agenten gesendet. Dabei erwartet diese Funktion folgende Parameter:

- Paket-ID. Dies ist die ID des Anfrage-Pakets, welches hiermit beantwortet werden soll.
- Group-ID. Die ID der Group, in der sich die angefragte Varibale befindet.
- Instanz-ID. Die ID der Instanz, also die Bezeichnung, welche die Variable im Teilbaum besitzt. Hängt man die Instanz-ID nahtlos an die Group-ID, so erhält man die von SNMP her bekannte OID der Variable.
- Typ der Variablen. Der Nummerncode des Typs der Variablen nach der Tabelle 17 im RFC 1592. Dabei ist zu beachten, daß dieser im Gegensatz zur Tabelle im RFC 1592 in hexadezimaler Darstellung anzugeben ist.
- Wert der Variablen. Hier wird der Wert der Variablen in der jeweiligen Darstellung übergeben. Wird in einer Reihe von Getnext-Requests das Ende der MIB erreicht, so ist der Funktion ResponseSuccess() als Wert der Variablen „ENDofMIB" zu übergeben.

4.3 Gleichbleibendes Gerüst des Subagenten

Die im vorigen Abschnitt erklärten Funktionen werden in der Regel für alle Subagenten direkt übernommen. Deshalb befinden Sie sich mit noch einigen weiteren Funktionen in der Funktionenbibliothek „subagent_standard.pl “. Diese Bibliothek enthält auch eine Funktion „Subagent() “welche die gesamte Koordination der einzelnen Funktionen übernimmt. Nach einer gewissen Vorarbeit (z.B. Belegung der globalen Variablen) kann diese Funktion direkt die Hauptfunktionen eines Subagenten übernehmen. Dazu zählt der Aufbau der Verbindung, das Abwickeln des Protokolls und das Aufrufen von weiterverarbeitenden Funktionen bei Eingang eines Requests vom Agenten.

4.4 Aufbau eines eigenen Subagenten

Zuerst erstellt man eine Datei, die das PERL 5.0 - Startskript für den Subagenten enthalten soll. Folgender Aufbau wird dringend empfohlen:

```
#!/sw/mnm/perl5/bin/perl

require "sub_gets.pl";
require "sub_sets.pl";
require "sub_undos.pl";
require "sub_coms.pl";
require "mib_gets.pl";
require "mib_sets.pl";
require "subagent_standard.pl";

#######################################################
# Definition einiger Konstanten
#######################################################
# Konstanten fuer den Socket
$AF_INET     = 2;
$SOCK_STREAM = 1;
$SOCK_DGRAM  = 2;
$SOCKADDR    = 'S n a4 x8';

#Konstanten fuer die SNMP-Kommunikation zum Agenten
$SNMP_PORT      = 161 ;
$SNMP_TRAP_PORT = 162 ;
$SNMP_COMMUNITY = "public" ;
$SNMP_TRAP_COMMUNITY    = "public";

#Konstanten fuer die DPI-Kommunikation
$GROUPID        = "1.3.6.1.3.100.7.1.";
$OID            = "1.3.6.1.3.100.7.1";
$DESCRIPTION    = "BeispielSubagent";

########################################################
```

```
# Definition globaler Variablen
########################################################
$AGENT_HOST     = "sunhegering8" ;      # Host des Agenten

########################################################
# Initialisierungsroutine fuer die MIB-Variablen, die
# der Subagent anbietet.
########################################################
sub initial_MIB
        {
        $MIB{'1.0'} = "02";     # Variable 1: Text
        $MIB{'2.0'} = "81";     # Variable 2: Integer
        $MIB{'3.1.1.0'} = "02";        # Tabelle Feld 1: Text
        $MIB{'3.1.2.0'} = "81";        # Tabelle Feld 2: Integer

        # sortierte Liste der MIB-Variablen:
        @MIB = sort by_hierarchy keys(%MIB);

&Subagent;    # Starte den Subagenten
```

Die einzigen Anpassungen, die erforderlich sein sollten, sind das Vorbelegen folgender globaler Variablen:

- $GROUPID: Hier trägt man die Group-ID (entspricht der Wurzel) des verwalteten Teilbaums ein.

- $OID: Hier trägt man die OID des Subagenten ein.

- $DESCRIPTION: Eine kurze Erklärung bzw. der Name des Subagenten wird hier angegeben.

- $AGENT_HOST: Diese globale Variable belegt man mit dem Namen des Hostrechners, auf dem der Agent läuft.

Alleine mit diesen Angaben würde sich der Subagent schon beim Agenten anmelden und sich registrieren können. Er würde Requests empfangen, könnte aber noch nicht korrekt darauf reagieren. Dazu muß dem Subagenten auch die MIB, die er zur Verfügung stellen soll, bekanntgemacht werden. Dies geschieht in der Funktion

`initial_MIB`.

Ein Beispiel für eine solche Funktion findet man in dem obigen Listing. Zu beachten sind einige Dinge:

- Der globale assoziative Array %MIB nimmt die Information zur MIB auf.

- Die Instance-ID der jeweiligen Variable dient als Zugriffsindex. Dabei ist darauf zu achten, daß die Instance-ID in „' “zu klammern ist. Ansonsten würden die Punkte von PERL 5.0 interpretiert.

- Die Elemente des assoziativen Arrays werden die jeweiligen Typen der MIB-Variablen. Dabei entspricht der Typ dem Nummerncode der Tabelle 17 im RFC 1592, jedoch in hexadezimaler Darstellung. Dieser Typ ist in Anführungszeichen zu setzen.
- Die letzte Zeile bleibt immer gleich. Sie erstellt eine entsprechend dem GetNext-Verhalten sortierte Liste der MIB-Variablen.
- Bei Tabellen reicht es, pro Variable eine Instanz (die Zeile 0) anzugeben.

Mit dieser zusätzlichen Angabe kann der Subagent schon Anfragen des Agenten auf Korrektheit bezüglich des Types und der Verfügbarkeit von angeforderten Varibalen überprüfen.

Nun muß man die MIB in If-then-else Kaskaden abbilden. Dabei werden zwei solcher Kaskaden benötigt. Die eine bildet die Variablen, die mit GET oder GETNEXT gelesen werden können, auf entsprechende Funktionen ab. Die andere bildet Variablen, die gesetzt werden können, auf entsprechende SET-, COMMIT- und UNDO-Funktionen ab. Dabei stehen Variablen, die geschrieben und gelesen werden können natürlich in beiden Kaskaden. Diese Kaskaden sind in den beiden Dateien mib_gets.pl und mib_sets.pl anzugeben. Die zwei Dateien zu unserem obigen kleinen Beipiel befinden sich auf Seite 94 und 95 im Anhang. In diesen Dateien erkennt man schnell die If-then-else-Kaskaden, die abhängig von der gewünschten Instanz eine externe Funktion aufrufen. Zum Anpassen eines eigenen Subagenten ist es nur erforderlich, die jeweils gewünschte Anzahl von else-Zweigen in die Datei einzufügen. Dabei kopiert man am schnellsten einen anderen Zweig. In dem Zweig sind dann die Bedingung (also die Instanz-ID) und die aufzurufende Funktion anzupassen. Die Kaskade in der Datei mib_sets.pl unterscheidet sich von der in der Datei mib_gets.pl noch dadurch, daß jeder Zweig wieder in drei Fälle zerlegt wird. Jeder dieser Zweige repräsentiert dabei eine Set, Undo oder Commit-Funkiton für die jeweilige Variable (oder Instanz).

Die letzte eigentliche Arbeit steckt nun darin, die einzelnen externen Funktionen zu schreiben. Empfehlenswert ist hier, die einzelnen Funktionen nach Typ in den Dateien sub_gets.pl, sub_sets.pl,sub_coms.pl und sub_undos.pl unterzubringen. Hierfür sind einfache Funktionen für das Beispiel im Anhang ab Seite 97 angegeben.
Die Set-Funktion bekommt den gewünschten neuen Wert in einem Array von Bytes in hexadezimaler Darstellung. Für die Umsetzung in den jeweiligen Datentyp ist der Subagenten-Programmierer selbst verantwortlich. Als Beispiele sind die Umwandlung eines Arrays in einen Textstring und die Umwandlung eines vier Byte umfassenden Arrays in eine Integer-Zahl angegeben.

Dies war ein kleines Beispiel für einen einfachen Subagenten. Aus diesen Darstellungen sind hoffentlich die nötigen Schritte zur Erstellung eigener Subagenten unter PERL 5.0 hervorgegangen. Dabei ist zu bedenken, daß die eigentliche Aufgabe nicht in der Erstellung eines DPI-Subagenten lag, der für alle Anwendungen perfekt ist. Es sollte ein DPI-Subagent für das Skript LinkTest entstehen. Falls dieser nun so modular sein sollte, daß er ohne großen Aufwand auch führ andere Anwendungen angepaßt werden kann, so war dies durchaus Absicht, aber nicht erklärtes Ziel der Entwicklung.

Kapitel 5

Zusammenfassung

Zusammenfassend soll hier erwähnt werden, daß mir dieses Fortgeschrittenen-Praktikum einen kleinen Einblick in einige große praktische Bereiche der Netzwerkprogrammierung gab. Es sollen hier nur die Schlagworte WWW, HTML, HTTP, SNMP, MIB's, DPI und Netzwerkmanagement genannt werden.
Ein weiterer sehr interessanter Aspekt dieser Arbeit war sicherlich PERL 5.0. Ich möchte erwähnen, daß sich PERL 5.0 sehr schnell erlernen läßt. PERL ist immer eine Gradwanderung zwischen sauberem, typisiertem Stil und verwaschenen schnellen Lösungen.
Daß der Bereich, in den meine Arbeit fiel, durchaus noch nicht „abgegrast “ist, erkennt man daran, daß uns immer wieder neue weiter Aufgaben begegneten, für die wir aber keine Zeit hatten. So sei zum Beipiel auf bessere Darstellung der Ergebnisse des WWW-Skripts hingewiesen.
Eine Aufgabe, die meiner Meinung nach im Zusammenhang mit einem Gesamtkonzept zwingend einer Lösung bedarf, ist folgendes: Mit der MIB hat man eine so exakt formal festgelegte Definition der Schnittstelle des Subagenten, daß sich eine automatische Generierung des Subagenten hieraus, bis zu einem gewissen Punkt geradezu aufdrängt.
Mit diesem Ausblick möchte ich meine Ausführungen beenden.

Literaturverzeichnis

[1] T. Berners-Lee and D. Connolly. Hypertext markup language - 2.0. draft-ietf-html-spec-04.txt, MIT/W3C, 1995.

[2] R. Fielding, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. draft-ietf-http-v11-spec-00.txt, UC Irvine, MIT/LCS, 1995.

[3] Bernd Hainzinger. Erweiterung eines snmpv2-agenten um eine schnittstelle zur interprozeßkommunikation. Fortgeschrittenenpraktikum, TU München, 1994.

[4] Nedo Haubelt. Entwurf und implementierung der www-anwendung „jahreswagenbörse“ bei der bmw ag. Diplomarbeit, TU München, 1995.

[5] Helmut Kopka and Patrick W. Daly. *A Guide to LaTex*. Addison-Wesley Co. Inc., United States of America, 1993.

[6] Randal L. Schwartz. *Learning PERL*. O'Reilly & Associates, Inc., 1993.

[7] Larry Wall and Randal L. Schwartz. *Programming PERL*. O'Reilly & Associates, Inc., 1992.

[8] B. Wijnen, G. Carpenter, K. Curren, A. Sehgal, and G. Waters. Simple network management protocol distributed protocol interface version 2.0. Rfc1592, T.J. Watson Research Center, IBM Corp., Bell Northern Research, 1994.

Anhang A

Kommentierte Liste der erstellten Dateien

A.1 Dateien bei LinkTest

Folgende Dateien werden in Bezug mit dem Skript LinkTest benötigt oder erstellt:

- LinkTest

  Diese Datei enthält das eigentliche Skript zur Konsistenzprüfung von HTML-Links. Das Ausführen dieser Datei startet die Prüfung. Als Parameter benutzt das Skript die Angaben in der Datei LinkTest.ini.

- LinkTest.ini

  Diese Datei enthält die Konfigurationsparameter des Skripts LinkTest.

- LinkTest.BaumErstellen

  Diese Datei enthält das Skript zur Erstellung eines Baums der Linkstruktur. Dabei holt es Information aus den Dateien LinkTest.ERL.LOG, LinkTest.ERR.LOG, LinkTest.NEU.LOG, die vom Programm LinkTest bei jedem Lauf erstellt werden.

- LinkTest.FehlerList

  Diese Datei enthält das Skript zur Auflistung der Fehler in ausführlicher Form. Information über die fehlerhaften Links holt sich dieses Skript aus der Datei LinkTest.ERR.LOG, die vom Programm LinkTest bei jedem Lauf erstellt wird.

- LinkTest.ERL.LOG

  Diese Datei wird vom Skript LinkTest erstellt. Sie enthält Angaben zu allen erfolgreich gelesenen Links.

- LinkTest.ERR.LOG

  Diese Datei wird vom Skript LinkTest erstellt. Sie enthält Angaben zu allen nicht erfolgreich gelesenen Links.

- LinkTest.NEU.LOG

  Diese Datei wird vom Skript LinkTest erstellt. Sie enthält Angaben zu allen Links, die dem Skript LinkTest bekannt waren, aber noch nicht überprüft wurden.

- LinkTest.log

  Diese Datei wird vom Skript LinkTest erstellt. In ihr werden alle an WWW-Server verschickte Requests mitprotokolliert.

- LinkTest.Nachricht

  Falls in der Initialisierungsdatei LinkTest.ini die Option `[Nachricht]=DEBUG;` gesetzt wurde, wird diese Datei vom Skript LinkTest erstellt. Sie enthält die Mail, die an den WebMaster verschickt worden wäre.

- LinkTest.NachrichtBetreuer

  Falls in der Initialisierungsdatei LinkTest.ini die Option `[Nachricht]=DEBUG;` gesetzt wurde, wird diese Datei vom Skript LinkTest erstellt. Sie enthält die Mails, die an die Betreuer verschickt worden wären.

A.2 Dateien des Subagenten

Folgende Dateien werden in Bezug mit dem Subagenten benötigt:

- subagent_start

  Diese Datei enthält einige, den Subagenten definierende Angaben. Überdies wird der Subagent durch Aufruf dieser Datei gestartet.

- subagent_standard.pl

  Diese Datei enthält das Gerüst des Subagenten und die Funktionsbibliothek für die DPI-Kommunikation.

- mib_gets.pl

  In dieser Datei ist die MIB für die lesbaren Variablen des Subagenten mittels einer If-then-else-Kaskade enthalten.

- mib_sets.pl

  In dieser Datei ist die MIB für die schreibbaren Variablen des Subagenten mittels einer If-then-else-Kaskade enthalten.

- sub_gets.pl

  Diese Datei enthält die speziellen Funktionen, die bei einem Get-Request für die einzelnen Variablen aufgerufen werden. Der Aufruf steht dabei in der Datei mib_gets.pl.

- sub_sets.pl

  Diese Datei enthält die speziellen Funktionen, die bei einem Set-Request für die einzelnen Variablen aufgerufen werden. Der Aufruf steht dabei in der Datei mib_sets.pl.

- sub_coms.pl

  Diese Datei enthält die speziellen Funktionen, die bei einem Commit-Request für die einzelnen Variablen aufgerufen werden. Der Aufruf steht dabei in der Datei mib_sets.pl.

- sub_undos.pl

  Diese Datei enthält die speziellen Funktionen, die bei einem Undo-Request für die einzelnen Variablen aufgerufen werden. Der Aufruf steht dabei in der Datei mib_sets.pl.

Anhang B

Benutzte Hilfsprogramme

Zu Testzwecken war es erforderlich, auf schon vorhandene Software zurückzugreifen. Speziell waren dies folgende Programme:

- tcpdump

  Das Programm tcpdump erlaubt es, den kompletten Verkehr auf dem Netzwerk zu betrachten und mitzuprotokollieren. Dabei kann einschränkend angegeben werden, von welchem Source- und Destination-Host die Nachrichten angezeigt, auf welchem Port mitgelauscht werden soll, usw. . Zu erwähnen ist, das tcpdump die Nachrichten, die von dem Host auf dem es gerade läuft abgeschickt werden, nicht anzeigt.

- snmpget

  Das Programm snmpget wurde benutzt, um einen Manager zu simulieren. Mit diesem Programm wurde ein SNMP-Get-Request an den Agenten verschickt. Dieser gab die Anfrage dann an den Subagenten weiter. Der Subagent beantwortete die Anfrage und snmpget zeigte das Ergebnis der Anfrage an. Es wurden nur Anfragen der Version 1 benutzt. Außerdem ist zu erwähnen, daß das Programm nur ordnungsgemäß lief, wenn es auf einem HP-Rechner gestartet wurde.

- snmpset

  Das Programm snmpset wurde benutzt, um einen Manager zu simulieren. Mit diesem Programm wurde eine komplette SNMP-Set Transaktion mit Set und Commit (ggf. auch Undo) an den Agenten verschickt. Dieser gab die Anfragen dann an den Subagenten weiter. Der Subagent bearbeitete die Anfragen und snmpset zeigte den neu gesetzten Wert an. Es wurden nur Anfragen der Version 1 benutzt. Außerdem ist zu erwähnen, daß das Programm nur ordnungsgemäß lief, wenn es auf einem HP-Rechner gestartet wurde.

- snmpwalk

  Das Programm snmpwalk wurde benutzt, um einen Manager zu simulieren. Mit diesem Programm wurde ein SNMP-GetNext-Request an den Agenten verschickt. Dieser gab die Anfrage dann an den Subagenten weiter. Der Subagent beantwortete die Anfrage und snmpwalk zeigte das Ergebnis der Anfrage an. Anschließend verschickte snmpwalk selbständig soviele Get-Next-Anfragen, bis die entsprechende MIB komplett durchlaufen war. Das Programm diente hauptsächlich zum Testen der Tabellen. Es wurden nur

Anfragen der Version 1 benutzt. Außerdem ist zu erwähnen, daß das Programm nur ordnungsgemäß lief, wenn es auf einem HP-Rechner gestartet wurde.

Anhang C

Listings zum Skript LinkTest

C.1 Listing von LinkTest.ini

```
################################################################################
# Initialisierungsdatei zum PERL-Skript LinkTest zur Kontrolle eines WWW-Servers
# auf inkonsistente Links
################################################################################

    # Die Namen der zu testenden WWW-Servers. Begonnen wird mit dem ersten.
[WWW_SERVER]=www.informatik.tu-muenchen.de;

    # Absoluter Pfad des ersten zu testenden Dokuments
[TOP_LINK]=/;

    # Liste der Endungen, bei denen ein einlesen des Kopfes als Kontrolle
    # reicht. z.B. Bilder (gif,tif) oder anderes, wo nicht nach Links
    # gesucht werden soll.
[ENDUNGEN]="gif","tif","jpg","GIF","TIF","JPG","ps";

    # Maximale Anzahl von zu testenden Links.
[MAX_LINKS]=10000;

    # Mit "Betreuer" ist weiterhin der Verantwortliche fuer eine Seite
    # gemeint.
    # Regulaerer Ausdruck in PERL-Syntax, um den Tag zu identifizieren,
    # indem die Betreueraddresse steht.Insgesamt muss der Ausdruck mit '
    # geklammert werden.
[BETREUER_TAG]='<meta name="Author"';

    # Regulaerer Ausdruck in PERL-Syntax, um die email-Adresse des
    # Betreuers innerhalb des Tags zu erkennen. Dabei befindet sich
    # die Adresse innerhalb der reunden Klammern. Insgesamt muss
    # der Ausdruck mit ' geklammert werden.
[BETREUER_ADRESSE]='value="(.*?)"';

    # email-Adresse, an die ausser den Betreuern eine Benachrichtigung
    # versand werden soll (z.B.Boss). Dieser Eintrag muss immer vor
    # [NACHRICHT] stehen.
[NACHRICHT_AN]=schuetzf@informatik.tu-muenchen.de;
```

```
    # Soll keine Nachricht generiert werden (=KEINE)?
    # Wenn ja soll Sie zu Debug-Zwecken in eine Datei gelegt
    # werden (=DEBUG) oder soll Sie verschickt werden (=SENDE)
[NACHRICHT]=DEBUG;

    # Text der versandten Nachricht.
    # Der Texte ist mit << >> zu klammern
    # Im Anschluss an den Text folgen Angaben zum Dokument
    # mit dem fehlerhaften Link.
    # In der Nachricht k"onnen Sie folgende Marken einfuegen,
    # die dann jeweils durch den aktuellen Wert ersetzt werden:
    # {}Datum{}    : Das Datum mit Zeitangabe, wann der Prueflauf
    #             gestartet wurde.
    # {}Link{}    : Absoluter Pfad mit Domain des fehlerhaften
    #             Links auf der Seite.
    # {}Seite{}    : Betreute Seite auf der fehlerhafte Link steht.
    # {}Zeile{} : Zeile im Source der Seite, in der der
    #           : fehlerhafte Link steht.
    # {}Fehlercode{}: Nummerncode des Fehlers.
    # {}FehlerLang{}: Ausfuehrliche Erklaerung des Fehlers.
    # Zeilenschaltungen und Tabs sind direkt an den gewuenschten Stellen
    # einzugeben.
[NACHRICHT_TEXT]=<<Auf einer von Ihnen betreuten HTML-Seite:
    {}Seite{}
befindet sich ein fehlerhafter Link:
    {}Link{}
in Zeile {}Zeile{}.
Der Fehlercode ist:
    {}Fehlercode{}
Eine Beschreibung des Fehlers ist:
    {}FehlerLang{}
Bitte berichtigen Sie diese Seite.
Prueflauf: {}Datum{}>>;
```

C.2 Listing von LinkTest

```
#!/sw/mnm/perl5/bin/perl
#######################################################################
# Diese Datei enthaelt das Programm LinkTest zum testen der Konsistenz
# von WWW-Links.
# Zu Beginn der Datei steht das Hauptprogramm.
# An diese schliessen sich folgende Funktionen an:
#    NeueLinksaufnehmen   : Nimmt die von HoleLinksZuEinerSeite gef. Links in einen
#                           Array auf
#    HoleLinksZuEinerSeite: Gibt alle Links auf einer angegebenen Seite zurueck
#    HoleKopf             : Testet das Vorhandensein eines Dokuments
#    VerschickeBeiFehler  : Fuer fehlerhafte Links wird ein Mail verschickt
#    FehlerLogAbarbeiten  : Laeuft durch die Fehler-Log-Datei und uebergibt den
#                           Fehlercode FehlerBearbeiten
#    FehlerBearbeiten     : Gibt String mit Fehlererklaerung zurueck
#
```

```
# Fuer den Ablauf wird die Datei LinkTest.ini benoetigt. Von ihr wird der Ablauf des
# Programms gesteuert. Das Programm erzeugt in jedem Fall folgende Dateien:
#    LinkTest.ERR.log     : Datei mit den gefundenen fehlerhaften Links
#    LinkTest.ERL.log     : Datei mit den erfolgreich abgearbeiteten Links
#    LinkTest.NEU.log     : Datei mit den noch nicht bearbeiteten Links (infolge
#                           der Angabe MaxAnzahlLinks)
#    LinkTest.log         : Datei mit den durchgefuehrten HTTP-Anfragen
# Ausserdem werden abhaenigig von der Einstellung NACHRICHT in der LinkTest.ini
# Datei folgende Dateien erstellt:
#    LinkTest.NachrichtBetreuer : Text, der an den Betreuer einer Seite geschickt
#                                 wuerde.
#    LinkTest.NachrichtBoss     : Text, der an den Gesamtverantwortlichen geschickt
#                                 wuerde.
#
# Als zentrale Datenstruktur wird ein sog. Assoziativer Array benutzt. In solchen
# Arrays werden saemtliche relevanten Daten als String abgelegt.
# Als Schluesselelement dafuer dient der absolute Pfad mit Domaine.
# Die Elemente des Arrays sind Strings mit folgendem Format:
# LinkMethode;Vaeter;Betreuer;LinkTyp
# - LinkMethode gibt an, welche Methode fuer den Zugriff zu benutzen ist
#     (z.B. HTTP,FTP,MAILTO,...)
# - Vaeter gibt alle Vaeter zu einer Seite innerhalb der durchsuchten Domaine an.
#    Diese sind dann durch | getrennt. Dabei ist bei jedem Vater der
#    absolute Pfad mit Domaine plus die Zeile, in der der Sohn-Link
#    steht durch :Z getrennt angegeben.
# - Hier steht die Mailadresse des ermittelten Betreuers
# - Hier steht, in welchem Zusammenhang auf das Dokument zugegriffen wird
#     (A,IMG,SRC,LINK)


######################################################################
# Lesen der Initialisierungsdatei, wie sie in der Kommandozeile als
# erstes Argument angegeben wurde. Bei keiner Angabe nimm LinkTest.ini

if ($ARGV[0])
    {
    $IniDatei = $ARGV[0];
    $ProgPfad = substr($IniDatei,0,rindex($IniDatei,"/"));
    print "$ProgPfad\n";
    chdir($ProgPfad);
    }
else
    {
    $IniDatei = 'LinkTest.ini'; # Defaultwert
    }

if(!open(INIDATEI,$IniDatei))
    {
    die "\nKonnte Initialisierungsdatei $IniDatei nicht oeffnen!\n";
    }
######################################################################
# Parsen der Ini-Datei und setzen der entsprechenden Variablen
while(<INIDATEI>)
```

```
    {
    if (/^((\s*#)|(\s+))/)    # Falls Kommentar- oder Leerzeile:
        {
        next;                 # ueberspringen
        }
    elsif (/\[WWW_SERVER\]=/) # Domainen, die kontrolliert werden sollen
        {                     # zu einem Regulaeren Ausdruck zusammensetzen
        /=(.*?);/;
        $HilfsVar = $1;
        $httpserver = '($Domain eq "'.join('") || ($Domain eq "',
                                          split(/,/,$HilfsVar)).'")';
        $HilfsVar=~/(.*?),/;
        $Domain = $1;
        }
    elsif (/\[TOP_LINK\]=/)
        {
        /=(.*?);/;
        $TopLink = "$1";
        }
    elsif (/\[ENDUNGEN\]=/)    # Endungen bei denen nur der Kopf benoetigt wird
        {                      # zu einem Regulaeren Ausdruck zusammensetzen
        /=(.*?);/;
        $Endungen = '($1 eq '.join(') || ($1 eq ',split(/,/,$1)).')';
        }
    elsif (/\[MAX_LINKS\]=/)
        {
        /=(.*?);/;
        $MaxLinks = "$1";
        }
    elsif (/\[BETREUER_TAG\]=/)
        {
        /=(.*?);/;
        $RegBetreuerTag = $1;
        }
    elsif (/\[BETREUER_ADRESSE\]=/)
        {
        /=(.*?);/;
        $RegBetreuerAdresse = $1;
        }
    elsif (/\[NACHRICHT\]=KEINE/)
        {
        /=(.*?);/;
        $Nachricht = $1;
        }
    }
close(INIDATEI);


$Datum = `date`;    # Datum und Zeit des Startes sichern

# Initialisiere assoz. Array mit erstem Link
$LinksNEU{"$Domain$TopLink"}=join(";","HTTP","TOP:Z0","TOP","TOP");
# Oeffne Log-Datei fuer die Sitzung
```

```
open(LOG,">LinkTest.LOG");
print LOG "Prueflauf vom $Datum\n";

# Nimm, solange es einen Eintrag gibt, das erste Element aus dem Neuenarray
# Achte dabei darauf, dass nicht mehr Aufrufe als gewuenscht behandelt werden.
while ( ($DomainLink=(keys(%LinksNEU))[0]) && ($Aufrufe < $MaxLinks))
        {
        $Aufrufe++;    # Anzahl der Aufrufe mitzaehlen.
        $Domain = substr($DomainLink,0,index($DomainLink,"/"));
        $Link = substr($DomainLink,index($DomainLink,"/"));
        # Beschreibung aus Array holen
        $Beschreibung = $LinksNEU{"$DomainLink"};
        ###################################################################
        # Liefere eine Liste von Links auf der Seite mit Betreuer zur Seite oder
# Fehler. Je nach Endung nur Kopf oder ganzes Dokument laden
        $Link=~/\/.*\.(.*)$/;    # Endung des Links herausholen
        if (eval ($Endungen) || !eval($httpserver) )   # Mit Endungen und Servern in
                                                       # der Initialisierungsdatei vgl.
        {    # Falls Kopf reicht
        print "$Aufrufe\tSende HEAD //$Domain$Link\n";# Aufruf anzeigen
        print LOG "Sende HEAD //$Domain$Link\n";# Aufruf mitloggen
        %NeueLinks = &HoleKopf($Link,$Domain);
        }
    else
        {    # Falls Seite noetig mit Linkkontrolle
        print "$Aufrufe\tSende GET  //$Domain$Link\n";# Aufruf anzeigen
        print LOG "Sende GET  //$Domain$Link\n";# Aufruf mitloggen
        %NeueLinks = &HoleLinksZuEinerSeite($Link,$Domain,$RegBetreuerTag,
                                                           $RegBetreuerAdresse);
        }
    ##################################################################################
    # Den Link, den man ueberprueft hat erledigen,das heisst bei einem Fehler in Fehler-
    # array speichern, ansonsten den ermittelten Betreuer dazuspeichern und in den Er-
    # ledigtarray ablegen. Fehler und Betreuer werden in dem Feld "FEHLER" bzw. "BETREUER"
    # des Arrays uebergeben. Sie sind anschliessend zu loeschen.
    if($NeueLinks{"FEHLER"})    # Tauchte bei zu pruefendem Link ein Fehler auf ?
        {
        # Beschreibung aufsplitten
        ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Beschreibung);
        $Betreuer=$NeueLinks{"FEHLER"}; # ins Betreuerfeld den Fehlercode
        # Beschreibung in ErrorArray speichern
        $LinksERR{"$DomainLink"}=join(";",$LinkMethode,$Vaeter,$Betreuer,$LinkTyp);
        }
    else
        {
        # Beschreibung aufsplitten
        ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Beschreibung);
        $Betreuer=$NeueLinks{"BETREUER"};
        # Beschreibung mit jeweiligem Betreuer speichern
        $LinksERL{"$DomainLink"}=join(";",$LinkMethode,$Vaeter,$Betreuer,$LinkTyp);
        delete $NeueLinks{"BETREUER"}; # Da dies reines Uebergabefeld war, loeschen.
        &NeueLinksaufnehmen;
        }
```

```
    delete $LinksNEU{$DomainLink};
    }

close(LOG); # Logfile schliessen



#######################################################################
# Statistik - Ausgabe

open (LOGERR,">LinkTest.ERR.LOG");
print LOGERR "################################################################\n";
print LOGERR "# Log-Datei zum Programm LinkTest (testen von Links auf einem WWW-Server)\n";
print LOGERR "# vom $Datum .\n";
print LOGERR "# HIER: LINKS, AUF DIE NICHT ZUGEGRIIFEN WERDEN KONNTE:\n";
print LOGERR "################################################################\n";
while(($Marke,$Wert)= each(%LinksERR))
    {
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Wert);
    print LOGERR "Url      : $LinkMethode://$Marke\n";
    print LOGERR "Vater    : ",join("\n           ",split(/\|/,$Vaeter)),"\n";
    print LOGERR "Fehler   : $Betreuer\n";
    print LOGERR "Intern --: $Marke;$Wert\n";
    }
close (LOGERR);

open (LOGNEU,">LinkTest.NEU.LOG");
print LOGNEU "################################################################\n";
print LOGNEU "# Log-Datei zum Programm LinkTest (testen von Links auf einem WWW-Server)\n";
print LOGNEU "# vom $Datum .\n";
print LOGNEU "# HIER: LINKS, DIE IN DIESEM LAUF NOCH NICHT GETESTET WURDEN,\n";
print LOGNEU "#       AUF DIE ABER ZUGEGRIFFEN WIRD:\n";
print LOGNEU "################################################################\n";
while(($Marke,$Wert)= each(%LinksNEU))
    {
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Wert);
    print LOGNEU "Url      : $LinkMethode://$Marke\n";
    print LOGNEU "Vater    : ",join("\n           ",split(/\|/,$Vaeter)),"\n";
    print LOGNEU "Intern --: $Marke;$Wert\n\n";
    }
close (LOGNEU);

open (LOGERL,">LinkTest.ERL.LOG");
print LOGERL "################################################################\n";
print LOGERL "# Log-Datei zum Programm LinkTest (testen von Links auf einem WWW-Server)\n";
print LOGERL "# vom $Datum .\n";
print LOGERL "# HIER: LINKS, AUF DIE MIT ERFOLG (OHNE FEHLER) ZUGEGRIFFEN WERDEN\n";
print LOGERL "#       KONNTE:\n";
print LOGERL "################################################################\n";

while(($Marke,$Wert)= each(%LinksERL))
    {
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Wert);
```

```
    print LOGERL "Url      : $LinkMethode://$Marke\n";
    print LOGERL "Vater    : ",join("\n           ",split(/\|/,$Vaeter)),"\n";
    print LOGERL "Betreuer : $Betreuer\n";
    print LOGERL "Intern --: $Marke;$Wert\n\n";
    }
close (LOGERL);

$Anzahl = keys(%LinksERL);
print "Anzahl erledigte Links ohne Fehler: $Anzahl\n";
$Anzahl = keys(%LinksERR);
print "Anzahl fehlerhafte Links: $Anzahl\n";


######################################################################
# Mail-Aufruf falls gewuenscht
if (!($Nachricht eq "KEINE"))
    {
    &VerschickeBeiFehler($IniDatei,%LinksERL,%LinksERR);
    }

######################################################################
# Hinweis auf Hilfsprogramm zur Fehleransicht.

print "Mit \"LinkTest.FehlerList\" koennen Sie sich eine ausfuehrliche Liste \n
       der aufgetretenen Fehler anzeigen lassen\n";
print "Mit \"LinkTest.BaumErstellen\" koennen Sie die Struktur des geprueften \n
       Bereichs als Baum ausgeben.\n";


###################################################################################
#    ENDE Hauptprogramm
###################################################################################


#####################################################################################
#
# Unterprogramm NeueLinksaufnehmen
#
#####################################################################################

# Gefundene Links , falls noch nicht schon wo vorhanden, in den Neuenarray aufnehmen


sub NeueLinksaufnehmen
    {
    # Nimm neue Links in Liste auf
    foreach $NeuerDomainLink (keys(%NeueLinks))
        {
        # Beschreibung vom vermeintlich neuen Link aufsplitten
        ($LinkMethode,$NeuerVater) = split(/;/,$NeueLinks{"$NeuerDomainLink"});

        ######################################################################
        # Pruefe ob Link schon im Neuenarray vorhanden
```

```
if ($LinksNEU{$NeuerDomainLink})
    {    # Wenn vorhanden:
    # Alte Beschreibung aufsplitten
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,
                                    $LinksNEU{"$NeuerDomainLink"});
    # Vaeter erweitern (sind sicher nicht doppelt,
    # da aus Unterprog nur einfach hochgereicht)
    $Vaeter = join("|", ($Vaeter,$NeuerVater));
    # $Beschreibung wieder zusammenbauen und in Liste ablegen
    $LinksNEU{"$NeuerDomainLink"}=join(";",($LinkMethode,$Vaeter,
                                    $Betreuer,$LinkTyp));

    next;    # Naechsten neuen Link
    }
######################################################################
# Pruefe ob Link schon im Erledigtenarray vorhanden
elsif ($LinksERL{$NeuerDomainLink})
    {    # Wenn vorhanden:
    # Alte Beschreibung aufsplitten
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,
                                 $LinksERL{"$NeuerDomainLink"});
    # Vaeter erweitern (sind sicher nicht doppelt,
    # da aus Unterprog nur einfach hochgereicht)
    $Vaeter = join("|", ($Vaeter,$NeuerVater));
    # $Beschreibung wieder zusammenbauen und in Liste ablegen
    $LinksERL{"$NeuerDomainLink"}=join(";",($LinkMethode,$Vaeter,
                                    $Betreuer,$LinkTyp));

    next;    # Naechsten neuen Link
    }
#######################################################################
# Pruefe ob Link schon im Fehlerarray vorhanden
elsif ($LinksERR{$NeuerDomainLink})
    {    # Wenn vorhanden:
    # Alte Beschreibung aufsplitten
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,
                                     $LinksERR{"$NeuerDomainLink"});

    # Vaeter erweitern (sind sicher nicht doppelt,
    # da aus Unterprog nur einfach hochgereicht)
    $Vaeter = join("|", ($Vaeter,$NeuerVater));
    # $Beschreibung wieder zusammenbauen und in Liste ablegen
    $LinksERR{"$NeuerDomainLink"}=join(";",($LinkMethode,$Vaeter,
                                     $Betreuer,$LinkTyp));
    next;    # Naechsten neuen Link
    }
else        ########################################################
    {    # Wenn Link noch nirgends vorhanden
    if ($LinkMethode eq "HTTP")
        {    # Falls ein HTTP-Link, nimm ihn in Neuenarray auf
        $LinksNEU{$NeuerDomainLink} = $NeueLinks{$NeuerDomainLink};
        }
    else
```

```
            {    # Falls andere Methode, nimm ihn in Erledigtenarray auf
            $LinksERL{$NeuerDomainLink} = $NeueLinks{$NeuerDomainLink};
            }
        }
    }
}



###################################################################
#
# Unterprogramm HoleLinksZuEinerSeite
#
###################################################################

sub HoleLinksZuEinerSeite
    {
    # Uebernahme der Parameter
    local($Url,$DomainPort,$RegBetreuerTag,$RegBetreuerAdresse) = @_ ;
    # Definition von lokalen Variablen
    local($Pfad,$Link,$LinkTyp,$LinkMethode,$Seite,@Marken,%Links);

    # Pfad abspalten und merken, um ihn bei relativen Links anfuegen zu koennen
    $Pfad = substr($Url,0,rindex($Url,"/")+1);

    ###################################################################
    # Port fuer die Kommunikation abspalten, falls angegeben. Wenn nicht,
    # mit 80 vorbesetzen.
    if ( $DomainPort=~/:/)
        {
        $Stelle = rindex($DomainPort,":");
        $HttpPort = substr($DomainPort,$Stelle+1);
        $Domain = substr($DomainPort,0,$Stelle);
        }
    else
        {
        $HttpPort = 80;
        }


    ###########################
    # Parameter fuer den Socket
    $AF_INET = 2;
    $SOCK_STREAM = 1;
    $sockaddr = 'S n a4 x8';

    # Name des Rechners, von dem aus geprueft wird ermitteln
    $hostname   = `hostname`;

    # Servername fuer die Anbindung aufbereiten
    ($name,$aliases,$proto) = getprotobyname('tcp');
    ($name,$aliases,$HttpPort) = getservbyname($HttpPort,'tcp')
```

```
unless $HttpPort =~ /^\d+$/;
($name, $aliases, $type, $len, $thisaddr) = gethostbyname($hostname);
$this = pack($sockaddr, $AF_INET, 0, $thisaddr);

# Gegenadresse auf gewuenschte Domaine setzen
($name, $aliases, $type, $len, $thataddr) = gethostbyname($Domain);
$that = pack($sockaddr, $AF_INET, $HttpPort, $thataddr);

    # Create a handle to the socket
  if (!socket(HTTP_Anschluss, $AF_INET, $SOCK_STREAM, $proto))
    {
    #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=600;    # Socket failed
    return %Links ;
      }

  # Assign the socket an address
  if (!bind(HTTP_Anschluss, $this))
    {
      #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=601;  # Bind failed
     return %Links ;
      }

    # an den Server connect'en
  if (!connect(HTTP_Anschluss,$that))
    {
       #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=602;    # Connect failed
    return %Links ;
    }

  select(HTTP_Anschluss);        # Anschluss HTTP_Anschluss anwaehlen
      $| = 1;             # Anschluss auf NICHT PUFFERN stellen
select(STDOUT);            # Anschluss STDOUT anwaehlen
$| = 1;             # Anschluss auf NICHT PUFFERN stellen

# $/ = "";    # in einem Stueck einlesen
$* = 1;        # ungleich 0 => Patternsuche ueber mehrere Zeilen

# Den Server um die Seite $Url bitten
print HTTP_Anschluss "GET $Url HTTP/1.0\n\n";

##############################################
# Seite vom Server uebernehmen
while (<HTTP_Anschluss>)
    {
    $Seite = join(" <Z$.Z> ",($Seite,$_));
    }
close HTTP_Anschluss;

if (!$Seite)
    {
```

```
    #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=603;    # Leere Seite
    return %Links ;
      }

$_ = substr($Seite,16);    # In Perl's Allzweckvariable stecken

###############################################
# Auf Fehlermeldungen des Servers ueberpruefen
if (/^([4,5]\d\d)/)
    {
    $Links{"FEHLER"}=$1;
    return %Links ;
    }

###############################################
# Seite weiterbearbeiten
s/\n/ /g;    # New Line loeschen
s/^.*?</</;    # Bis zum ersten Tag alles loeschen
s/>.*?</>;</g;    # Nur Tags behalten,durch Strichpunkt getrennt
s/#.*?(["\s])/$1/g;    # Directives in der Seite loeschen

@Marken     = split(/;/);    # Array @Marken mit den Tags fuellen

foreach (@Marken)    # jede Marke ueberpruefen:
    {
    if (/<Z(\d*?)Z>/)    # Zeilennummer merken
        {
        $Zeile = $1;
        next;
        }
    ########################################
    # Betreuer heraussichern
    if (/$RegBetreuerTag/i)            # falls Betreuer
        {
        /$RegBetreuerAdresse/i;        # URI suchen, in $1 ablegen
        $Link = join("",($1,$2));    # Fund mit oder ohne " $Link zuweisen
        $Links{"BETREUER"} = $Link ;    # Betreuer sichern
        next;
        }


    ########################################
    # Links herausarbeiten
    if (/<IMG /i)        # falls Image
        {
        /SRC="(.*?)"|SRC=(.*?)[\s>]/i;    # URI suchen, in $1 ablegen
        $Link = join("",($1,$2));    # Fund mit oder ohne " $Link zuweisen
        $LinkTyp = "IMG";    # Link-Typ in dem der Link vorkommt sichern
        }
    elsif (/<A /i)            # falls Anker
        {
        /HREF="(.*?)"|HREF=(.*?)[\s>]/i;    # URI suchen, in $1 ablegen
```

```
    $Link = join("",($1,$2));    # Fund mit oder ohne " $Link zuweisen
    $LinkTyp = "A";        # Link-Typ in dem der Link vorkommt sichern
    }
elsif (/<LINK /i)        # falls Link
    {
    /HREF="(.*?)"|HREF=(.*?)[\s>]/i;    # URI suchen, in $1 ablegen
    $Link = join("",($1,$2));    # Fund mit oder ohne " $Link zuweisen
    $LinkTyp = "LINK";    # Link-Typ in dem der Link vorkommt sichern
    }
elsif (/<INPUT /i)        # falls Input
    {
    /SRC="(.*?)"|SRC=(.*?)[\s>]/i;    # URI suchen, in $1 ablegen
    $Link = join("",($1,$2));    # Fund mit oder ohne " $Link zuweisen
    $LinkTyp = "INPUT";    # Link-Typ in dem der Link vorkommt sichern
    }
else                # falls nichts von alledem,
    {
    next;                # ueberpruefe naechste Marke
    }

########################################
# Link gefunden, also weiterverarbeiten
if ($Link)            # Link ist nicht leer?
    {
    ################################################
    # Methoden abspalten
    if ($Link=~s/http://)        # Methode http? (wenn ja http: weg)
        {
        $LinkMethode = "HTTP";
        }
    elsif ($Link=~s/mailto://)    # Methode mailto? (wenn ja mailto: weg)
        {
        $LinkMethode = "MAILTO";
        }
    elsif ($Link=~s/ftp://)        # Methode ftp? (wenn ja ftp: weg)
        {
        $LinkMethode = "FTP";
        }
    elsif ($Link=~s/news://)    # Methode news? (wenn ja news: weg)
        {
        $LinkMethode = "NEWS";
        }
    elsif ($Link=~s/file://)    # Methode news? (wenn ja news: weg)
        {
        $LinkMethode = "FILE";
        }
    else                # Alles andere auf http. Das wird naemlich
        {            # weitergeprueft und man erkennt so einen Fehler
        $LinkMethode = "HTTP";
        }
    ################################################
    # absoluten Pfad bauen
    if (!($Link=~/^\//))    # Falls relativer Pfad (kein / oder // am Anfang)
```

```
                {
                $Link = "$Pfad$Link";    # Pfad vorne anhaengen
                }
            if ($Link=~s/^\/\/(.*?)//)    # Falls mit Domaine ( // am Anfang)
                {
                $DomainNeu = $1;        # Domaine sichern
                }
            else
                {
                $DomainNeu = $DomainPort;    # sonst Vaterdomain
                }

            unless ($Links{"$DomainNeu$LinkNeu"})    # undef falls noch nicht vorhanden
                {
                # Neuen Link aufnehmen
                $Links{"$DomainNeu$Link"}=join(";",($LinkMethode,
                                               "$DomainPort$Url:Z$Zeile","",$LinkTyp));
                }
            }
        }    # Naechste Marke bearbeiten


    return %Links ;
    }


######################################################################
#
# Unterprogramm HoleKopf
#
######################################################################
sub HoleKopf
    {
    # Uebernahme der Parameter
    local($Url,$DomainPort) = @_ ;
    # Definition von lokalen Variablen
    local(%Links);

    ######################################################################
    # Port fuer die Kommunikation abspalten, falls angegeben. Wenn nicht,
    # mit 80 vorbesetzen.
    if ( $DomainPort=~/:/)
        {
        $Stelle = rindex($DomainPort,":");
        $HttpPort = substr($DomainPort,$Stelle+1);
        $Domain = substr($DomainPort,0,$Stelle);
        }
    else
        {
        $HttpPort = 80;
        }
```

```
###########################
# Parameter fuer den Socket
$AF_INET = 2;
$SOCK_STREAM = 1;
$sockaddr = 'S n a4 x8';

# Name des Rechners, von dem aus geprueft wird ermitteln
$hostname   = `hostname`;

# Servername fuer die Anbindung aufbereiten
($name,$aliases,$proto) = getprotobyname('tcp');
($name,$aliases,$HttpPort) = getservbyname($HttpPort,'tcp')
unless $HttpPort =~ /^\d+$/;
($name, $aliases, $type, $len, $thisaddr) = gethostbyname($hostname);
$this = pack($sockaddr, $AF_INET, 0, $thisaddr);


# Gegenadresse auf gewuenschte Domaine setzen
($name, $aliases, $type, $len, $thataddr) = gethostbyname($Domain);
$that = pack($sockaddr, $AF_INET, $HttpPort, $thataddr);

    # Create a handle to the socket
  if (!socket(HTTP_Anschluss, $AF_INET, $SOCK_STREAM, $proto))
    {
    #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=600;
    return %Links ;
      }

  # Assign the socket an address
  if (!bind(HTTP_Anschluss, $this))
    {
      #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=601;
     return %Links ;
      }

  # Connect to the server
  if (!connect(HTTP_Anschluss,$that))
    {
       #Fehlermeldung in Rueckgabe schreiben und Unterprogramm beenden
     $Links{"FEHLER"}=602;
    return %Links ;
    }


# Eliminate buffering
  select(HTTP_Anschluss);        # Anschluss HTTP_Anschluss anwaehlen
      $| = 1;             # Anschluss auf NICHT PUFFERN stellen
select(STDOUT);             # Anschluss STDOUT anwaehlen
$| = 1;             # Anschluss auf NICHT PUFFERN stellen

# Den Server um den Kopf der Seite $Url bitten
```

```
    print HTTP_Anschluss "HEAD $Url HTTP/1.0\n\n";

    ################################################################
    # Nur erste Headerzeile vom Server uebernehmen
    <HTTP_Anschluss>=~/HTTP\/1\.0 (\d\d\d)/;

    ########################################################
    # Auf Fehlermeldungen des Servers ueberpruefen
    if ($1 >= 400)
        {
        $Links{"FEHLER"}=$1;
        return %Links ;
        }
    }


#######################################################################
#
# Unterprogramm VerschickeBeiFehler
#
#######################################################################
sub VerschickeBeiFehler
    {
    # Uebernahme der Parameter
    local($IniDatei,%Links) = @_ ;
    # Definition von lokalen Variablen
    #local($NachrichtText,$InNachrichtText,$INI);

    $/ = "\n";    # zeilenweise einlesen
    $* = 0;        # ungleich 0 => Patternsuche ueber mehrere Zeilen

    if(!open(INIDATEI,$IniDatei))
        {
        die "\nKonnte Initialisierungsdatei $IniDatei nicht oeffnen!\n";
        }
    while(<INIDATEI>)    # Durchsuche ini.Datei
        {
        $INI=$_;
        if ($InNachrichtText eq "Ja")
            {
            if ($INI=~s/>>;//)    # Falls Endezeichen zu Text in der Zeile enthalten,
                {        # loesche dieses und ...
                $NachrichtText = join("\n",$NachrichtText,$INI);
                $InNachrichtText = "Nein";
                next;
                }
            else
                {
                $NachrichtText = join("\n",$NachrichtText,$INI);
                next;
                }
            }
```

```
    if ($INI=~/^((\s*#)|(\s+))/)       # Falls Kommentar- oder Leerzeile:
        {
        next;            # ueberspringen
        }
    elsif ($INI=~/\[NACHRICHT_AN\]=/)    # Adresse von BOSS speichern
        {
        /=(.*?);/;
        $NachrichtAn  = "$1";
        next;
        }
    elsif ($INI=~/^\[NACHRICHT\]=DEBUG;/)    # Soll in eine Datei abgelegt werden?
        {
        $MailOpenBoss = '>LinkTest.NACHRICHT';    # In diese Datei legen
        next;
        }
    elsif ($INI=~/^\[NACHRICHT\]=SENDE/)    # Soll gesendet werden?
        {
        $MailFlag = "SENDE";
        $MailOpenBoss = "|mail $NachrichtAn";
        next;
        }
    elsif ($INI=~/\[NACHRICHT_TEXT\]=<<(.*)/)    # Hole 1. Zeile des Mailtextes
        {
        $NachrichtText = $1;
        $InNachrichtText = "Ja";
        next;
        }
    }
close(INIDATEI);



####################################################################
# Mail an wegschicken
if (!open(MAIL_BOSS,$MailOpenBoss))
    {
    print "Konnte Filehandle nicht oeffnen";
    }
print MAIL_BOSS "Folgender Text wurde bei einem Fehler verschickt:\n----\n";
print MAIL_BOSS "$NachrichtText\n----\n";
print MAIL_BOSS "Auflistung der Fehler:\n=====================\n";

@Nachricht = split("{}",$NachrichtText);

$AnzahlFehler = 0;        # Fehleranzahl mitzaehlen
while(($Link,$Beschreibung)= each(%Links))    # Laufe durch den Array
    {
    ($LinkMethode,$Vaeter,$Betreuer,$LinkTyp) = split(/;/,$Beschreibung);
    if ($Betreuer=~/^\D\D\D/ || !($Betreuer))
        {
        next;
        }
    $AnzahlFehler++;
```

```
print MAIL_BOSS "\nUrl      : $LinkMethode://$Link\n";
print MAIL_BOSS "Vater    : ",join("\n\t\t",split(/\|/,$Vaeter)),"\n";
print MAIL_BOSS "Fehler   : $Betreuer\n";
print MAIL_BOSS "Betreuer : \n";
@Vaeter=split(/\|/,$Vaeter);    # Array Vaeter mit den Vaetern fuellen
foreach $Vater (@Vaeter)
    {
    $Zeile = substr($Vater,rindex($Vater,":Z")+2);
    $Vater = substr($Vater,0,rindex($Vater,":Z"));
    $VaterBeschreibung=$Links{$Vater};
    ($VaterLinkMethode,$Vaeter,$VaterBetreuer,$LinkTyp)=
                             split(/;/,$VaterBeschreibung);
    if ($VaterBetreuer)
        {
        print MAIL_BOSS "$VaterBetreuer\n";
        if ($MailFlag)
            {
            $MailOpenBetreuer = "|mail $VaterBetreuer";
            }
        else
            {
            $MailOpenBetreuer = '>LinkTest.NACHRICHT.BETREUER';
            }
        open(MAIL_BETREUER,$MailOpenBetreuer);
        foreach $Text (@Nachricht)
            {
            if ($Text eq 'Link')
                {
                print MAIL_BETREUER $Link;
                }
            elsif ($Text eq 'Seite')
                {
                print MAIL_BETREUER $Vater;
                }
            elsif ($Text eq 'Zeile')
                {
                print MAIL_BETREUER $Zeile;
                }
            elsif ($Text eq 'Datum')
                {
                print MAIL_BETREUER `date`;
                }
            elsif ($Text eq 'Fehlercode')
                {
                print MAIL_BETREUER $Betreuer;
                }
            elsif ($Text eq 'FehlerLang')
                {
                $FehlerLang = &FehlerBearbeiten($Betreuer);
                print MAIL_BETREUER $FehlerLang;
                }
            else
                {
```

```
                        print MAIL_BETREUER $Text;
                        }

                    }
                close(MAIL_BETREUER);
                }
            }
        }

    print MAIL_BOSS "\n=========\nInsgesamt traten $AnzahlFehler Fehler auf\n";
    close(MAIL_BOSS);
    }
```

C.3 Listing von LinkTest.FehlerList

```
#!/usr/local/dist/bin/perl
######################################################################
#
# Programm     : LinkTest.FehlerList
# Parameter    : Name der zugrundeliegenden Fehler-Log-Datei von LinkTest
# Beschreibung : Gibt eine detailiertere Fehlerbeschreibung zu den
#          bei einem Lauf von LinkTest festgestellten Fehlern aus.
#
######################################################################
# Uebernahme der Parameter

if ($ARGV[0])
    {
    $FehlerLogDatei = $ARGV[0] ;
    }
else
    {
    $FehlerLogDatei = 'f.ERR.LOG';
    }

####################################################################
# Die Fehler-Log-Datei "offnen, die interne Darstellung herausziehen
# und damit die Variablen belegen

if(!open(LOGERR,"$FehlerLogDatei"))
    {
    die "\nKonnte Fehler-Log-Datei $FehlerLogDatei nicht "offnen!\n";
    }

while (<LOGERR>)
    {
    if (/^Intern --:/)
        {
        $Beschreibung = substr($_,11);
        ($DomainPortLink,$LinkMethode,$VaeterZeile,$Fehler) = split(";",$Beschreibung);
```

```
        print "\nLink      : $DomainPortLink\n";
        print "Vaeter    : ",join("\n           ",split(/\|/,$VaeterZeile)),"\n";
        print &FehlerBearbeiten($Fehler);
        }
    else
        {
        next;
        }
    }

close (LOGERR);

######################################################################
#
# Unterprogramm FehlerBearbeiten
#
######################################################################

sub FehlerBearbeiten
    {
    # Uebernahme der Parameter
    local($Fehler) = @_ ;
    # Definition von lokalen Variablen
    # local($FehlerLang);

    if ($Fehler == 400)
        {
        $FehlerLang = "Fehlercode: 400 Bad Request\nKategorie : Client-Fehler\n
        Klartext  : Die Syntax der angegebenen Url war falsch. Ein weiterer Versuch\n
        wird auch fehlschlagen. Bessern Sie die Url aus.\n";
        }
    elsif ($Fehler == 401)
        {
        $FehlerLang = "Fehlercode: 401 Unauthorized\nKategorie : Client-Fehler\n
        Klartext  : Die Seite, auf die Sie zugreifen wollen erfordert eine
        Authentification.\nUebermitteln Sie diese im Header ihrer Anfrage.\n
        Wenn Sie eine Authentifikation mitgeschickt haben, zeigt dieser Fehler an,\n
        dass diese nicht akzeptiert wurde.";
        }
    elsif ($Fehler == 402)
        {
        $FehlerLang = "Fehlercode: 402 Payment Required\nKategorie : Client-Fehler\n
        Klartext  : Wird noch nicht benutzt\n";
        }
    elsif ($Fehler == 403)
        {
        $FehlerLang = "Fehlercode: 403 Forbidden\nKategorie : Client-Fehler\n
        Klartext  : Der Server hat die Anfrage verstanden, weigert sich aber diese zu
        erfuellen.\nEine Wiederholung oder Authentifikation hat keinen Sinn.\n
        Diese Meldung wird oft dazu benutzt, eine Seite ohne Begruendung zu
        verweigern.\n";
        }
    elsif ($Fehler == 404)
```

```
    {
    $FehlerLang = "Fehlercode: 404 Not Found\nKategorie : Client-Fehler\n
    Klartext  : Seite konnte unter dieser Adresse nicht gefunden werden.\n";
    }
elsif ($Fehler == 405)
    {
    $FehlerLang = "Fehlercode: 405 Method Not Allowed\nKategorie : Client-Fehler\n
    Klartext  : Die fuer dieses Objekt erforderliche Zugriffsmethode ist nicht
    explizit im Kopf angegeben.\nEs wurde deshalb nichts geschickt.\n";
    }
elsif ($Fehler == 406)
    {
    $FehlerLang = "Fehlercode: 406 None Acceptable\nKategorie : Client-Fehler\n
    Klartext  : Der Server hat die Seite gefunnden, schickt sie aber nicht,\n
    da Sie der vom Clienten angegebenen Accept-Klausel nicht entspricht.\n";
    }
elsif ($Fehler == 407)
    {
    $FehlerLang = "Fehlercode: 407 Proxy Authentication Required\nKategorie :
    Client-Fehler\nKlartext  : Fuer die Seite, auf die Sie zugreifen wollen,
    fordert der Proxy eine Authentification.\nUebermitteln Sie diese im Header
    ihrer Anfrage.\nWenn Sie eine Authentifikation mitgeschickt haben, zeigt
    dieser Fehler an,\n dass diese nicht akzeptiert wurde.";
    }
elsif ($Fehler == 408)
    {
    $FehlerLang = "Fehlercode: 408 Request Timeout\nKategorie : Client-Fehler\n
    Klartext  : Der Server hat auf eine Anfrage gewartet, aber der Client hat
    diese nach dem Erstellen der Verbindung\nnicht schnell genug gesendet.";
    }
elsif ($Fehler == 409)
    {
    $FehlerLang = "Fehlercode: 409 Conflict\nKategorie : Client-Fehler\n
    Klartext  : Dieser Fehler wird tritt wenn, nur bei den Methoden PUT und
    PATCH auf.\nEr gibt zumeist an, dass zwei Parteien eine Datei gleichzeitig
    modifiziert haben.\nEs laegen also beim speichern zwei unterschiedliche
    Versionen vor, was den Konflikt ausl"ost.\n";
    }
elsif ($Fehler == 410)
    {
    $FehlerLang = "Fehlercode: 410 Gone\nKategorie : Client-Fehler\n
    Klartext  : Die angef"orderte Seite gibt es nicht mehr.\n";
    }
elsif ($Fehler == 411)
    {
    $FehlerLang = "Fehlercode: 411 Length Required\nKategorie : Client-Fehler\n
    Klartext  : Der Server erfuellt die Anfrage erst, wenn der Client ein
    gueltiges Content-Length Feld mitschickt.\n";
    }
elsif ($Fehler == 412)
    {
    $FehlerLang = "Fehlercode: 412 Unless True\nKategorie : Client-Fehler\n
    Klartext  : Die Bedingung im request-header Feld Unless ist wahr.
```

```
        Der Client kann jetzt entscheiden wie er fortfaehrt.\n";
        }
    elsif ($Fehler == 500)
        {
        $FehlerLang = "Fehlercode: 500 Internal Server Error\nKategorie : Server-Fehler\n
        Klartext  : Es trat ein unerwarteter Zustand beim Server ein, weshalb er
        die Anfrage nicht erfuellen konnte.\n";
        }
    elsif ($Fehler == 501)
        {
        $FehlerLang = "Fehlercode: 501 Not Implemented\nKategorie : Server-Fehler\n
        Klartext  : Der Server beherrscht die benutzte Methode nicht.\n";
        }
    elsif ($Fehler == 502)
        {
        $FehlerLang = "Fehlercode: 502 Bad Gateway\nKategorie : Server-Fehler\n
        Klartext  : Der Proxy erhielt eine Fehlermeldung des angesprochenen Servers.\n";
        }
    elsif ($Fehler == 503)
        {
        $FehlerLang = "Fehlercode: 503 Service Unavailable\nKategorie : Server-Fehler\n
        Klartext  : Der Server ist momentan ueberlastet oder nicht verfuegbar.\n
        Zu einem spaeteren Zeitpunkt kann wieder zugegriffen werden.\n";
        }
    elsif ($Fehler == 504)
        {
        $FehlerLang = "Fehlercode: 504 Gateway Timeout\nKategorie : Server-Fehler\n
        Klartext  : Der Proxy hat innerhalb der Zeitspanne keine Antwort vom
        angeforderten Server erhalten.\n";
        }
    elsif ($Fehler == 600)
        {
        $FehlerLang = "Fehlercode: 600 (intern), Socket failed\nKategorie : Fehler
        waehrend Skriptausfuehrung\nKlartext  : Es war nicht m"oglich, einen Socket
        fuer die Kommunikation zu erstellen.\n";
        }
    elsif ($Fehler == 601)
        {
        $FehlerLang = "Fehlercode: 601 (intern), Bind failed\nKategorie : Fehler
        waehrend Skriptausfuehrung\nKlartext  : Es war nicht m"oglich, eine
        Verbindung zwischen den Rechnern einzurichten.\n";
        }
    elsif ($Fehler == 602)
        {
        $FehlerLang = "Fehlercode: 602 (intern), Connect failed\nKategorie : Fehler
        waehrend Skriptausfuehrung\nKlartext  : Es war nicht m"oglich, von diesem
        Rechner aus eine Verbindung zum der Domain\nentsprechenden Server aufzubauen.\n";
        }
    elsif ($Fehler == 603)
        {
        $FehlerLang = "Fehlercode: 603, Leere Seite\nKategorie : Warnung\n
        Klartext  : Es wurde eine leere Seite empfangen. Dies kann durchaus so
        gewuenscht sein. Trotzdem wird hier darauf hingewiesen.\n";
```

```
        }
     else
        {
        $FehlerLang = "Keine ausfuehrliche Beschreibung fuer den Fehler mit Fehlercode:
        \n$Fehler\n vorhanden";
        }
    return  ($FehlerLang);
    }
```

C.4 Listing von LinkTest.BaumErstellen

```
#!/usr/local/dist/bin/perl
########################################################################
# Programm     : LinkTest.BaumErstellen
# Parameter    : Maximale Zeilenlaenge (Standard: 80 Zeichen)
#                Gewuenschte Wurzel (Standard: Wurzel bei Aufruf von LinkTest)
#                    In diesem Fall muss Zeilenlaenge expl. angeg. werden
# Beschreibung : Erstellt aus den letzten erzeugten Log-Dateien von LinkTest
#                einen Baum der Aufrufstruktur. Das Programm setzt in den
#                Baum "-->", falls die max. Zeilenlaenge ueberschritten
#                wuerde und "-s.o." falls ein Zweig in einen Zyklus
#                fuehren wuerde
########################################################################
# Uebernahme der Parameter
if ($ARGV[0])
        {
        $MaxZeile = $ARGV[0] ;
        }
else
        {
        $MaxZeile = 80;
        }

if ($ARGV[1])
        {
        $Wurzel = $ARGV[1] ;
        }
else
        {
        $Wurzel = undef;
        }

# Lokale Groessen
local(%Baum);
# Initialisieren des Log-File-Arrays mit den Standardnamen
$LogDatei[0] = 'f.ERL.LOG';
$LogDatei[1] = 'f.NEU.LOG';
$LogDatei[2] = 'f.ERR.LOG';

#######################################################################
# Die Log-Dateien oeffnen, die interne Darstellung herausziehen
# und damit die Variablen belegen
```

```
for ($i=0;$i<=2;$i++)
        {
        if(!open(LOG,$LogDatei[$i]))
                {
                die "\nKonnte Log-Datei $LogDatei nicht oeffnen!\n";
                }
        while (<LOG>)
                {
                if (/^Intern --:/)
                        {
                        $Beschreibung = substr($_,11);
                        ($Link,$LinkMethode,$Vaeter) = split(";",$Beschreibung);
                        &InBaumAufnehmen($Link,$Vaeter);
                        }
                else
                        {
                        next;
                        }
                }
        close (LOG);
        }

$|=1;           # Ohne zu puffern ausgeben
if (!$Wurzel)   # evtl. mit Standardwert belegen
        {
        ($Marke,$AnzahlSoehne,$Wurzel) = split(/;/,$Baum{"TOP"});
        }
&SoehneAusgeben($Wurzel,0);

######################################################################
#
# Unterprogramm InBaumAufnehmen
#
######################################################################
sub InBaumAufnehmen
        {
        # Uebernahme der Parameter
        local($Sohn,$Vaeter) = @_ ;
        # Definition von lokalen Variablen
        local($Soehne,@Soehne,$Vater,@Vaeter,$AnzahlSoehne,$Marke);

        if ( !$Baum{$Sohn} )              # Falls Sohn noch nicht im Baum:
                {                        # Nimm ihn schon auf
                $Baum{$Sohn} = "Frei;0;";# vorbelegt mit 0 Soehnen
                }
        @Vaeter = split(/\|/,$Vaeter);   # Lege Vaeter in Array
        foreach $Vater (@Vaeter)         # Fuer jeden Vater:
                {
                $Vater = substr($Vater,0,rindex($Vater,":Z")); # Zeilenzahl entf.
                if ( $Baum{$Vater} )     # Falls Vater schon vorhanden:
                        {                # Haenge neuen Sohn zusaetzlich hin
                        ($Marke,$AnzahlSoehne,$Soehne) = split(/;/,$Baum{$Vater});
                        $AnzahlSoehne++;
```

```
                        if ($Soehne)
                                {
                                $Soehne = join("|",($Soehne,$Sohn));
                                }
                        else
                                {
                                $Soehne = $Sohn;
                                }
                        $Baum{$Vater} = join(";",($Marke,$AnzahlSoehne,$Soehne));
                        }
                else                        # Erster Sohn zu diesem Vater
                        {
                        $Baum{$Vater} = "Frei;1;$Sohn"; # Eintragen:1 Sohn,Name
                        }
                }
        }

########################################################################
#
#  Unterprogramm SoehneAusgeben
#
########################################################################
sub SoehneAusgeben
        {
        # Uebernahme der Parameter
        local($Vater,$Stelle) = @_;
        # Definition von lokalen Variablen
        local($Sohn,$Soehne,@Soehne,$AnzahlSoehne,$Marke,$i);

        for ($i=0;$i<$Stelle;$i++)
                {
                print '|   ';
                }

        if (($Stelle*4+length($Vater)) < $MaxZeile)
                {
                print "$Vater";
                $Stelle++;
                }
        else
                {
                print "-->\n";
                return;
                }

        ($Marke,$AnzahlSoehne,$Soehne) = split(/;/,$Baum{$Vater});
        @Soehne = split(/\|/,$Soehne);

        if ($Marke eq "Frei")
                {
                print "\n";
                $Marke = "Belegt";      # Zyklus verhindern
                $Baum{$Vater} = join(";",($Marke,$AnzahlSoehne,$Soehne));
```

```
        foreach $Sohn   (@Soehne)
                {
                &SoehneAusgeben($Sohn,$Stelle);
                }
        $Baum{$Vater} = join(";",($Marke,$AnzahlSoehne,$Soehne));
        }
else
        {
        print "-s.o.\n";
        }
}
```

Anhang D

Listings zum Subagenten

D.1 Listing von subagent_start

```
#!/sw/mnm/perl5/bin/perl

require "sub_gets.pl";
require "sub_sets.pl";
require "sub_undos.pl";
require "sub_coms.pl";
require "mib_gets.pl";
require "mib_sets.pl";
require "subagent_standard.pl";

#######################################################
# Definition einiger Konstanten
#######################################################
# Konstanten fuer den Socket
$AF_INET = 2;
$SOCK_STREAM = 1;
$SOCK_DGRAM  = 2;
$SOCKADDR = 'S n a4 x8';

#Konstanten fuer die SNMP-Kommunikation zum Agenten
$SNMP_PORT     = 161 ;
$SNMP_TRAP_PORT    = 162 ;
$SNMP_COMMUNITY    = "public" ;
$SNMP_TRAP_COMMUNITY    = "public";

#Konstanten fuer die DPI-Kommunikation
$GROUPID    = "1.3.6.1.3.100.7.1.";
$OID        = "1.3.6.1.3.100.7.1";
$DESCRIPTION    = "wwwlinkSubagent";

########################################################
# Definition globaler Variablen
########################################################
$AGENT_HOST    = "sunhegering8" ;    # Host des Agenten

########################################################
```

```
# Initialisierungsroutine fuer die MIB-Variablen, die
# der Subagent anbietet.
#######################################################
sub initial_MIB
    {
    $MIB{'1.0'} = "02";    # Nachricht an
    $MIB{'2.0'} = "02";    # zu durchsuchende Domaine
    $MIB{'3.0'} = "02";    # Toplink
    $MIB{'4.0'} = "81";    # Maximum der zu testenden Links
    $MIB{'5.0'} = "02";    # Reg. Ausdruck fuer Betreuer-Tag
    $MIB{'6.0'} = "02";    # Reg. Ausdruck fuer Betreuer-Adresse
    $MIB{'7.0'} = "02";    # Endungen, bei denen Head reicht
    $MIB{'8.0'} = "02";    # Nachricht (Keine,Sende,Debug)
    $MIB{'9.0'} = "02";    # Nachricht Text
    $MIB{'10.1.1.0'} = "02";    # Tabelle Feld 1: Link
    $MIB{'10.1.2.0'} = "02";    # Tabelle Feld 2: Methode
    $MIB{'10.1.3.0'} = "02";    # Tabelle Feld 3: LinkVaeter
    $MIB{'10.1.4.0'} = "02";    # Tabelle Feld 4: Betreuer
    $MIB{'99.0'} = "81";    # Kontroll-Variable fuer den Subagenten

    @MIB = sort by_hierarchy keys(%MIB);    # sortierte Liste der MIB-Variablen
    }

#######################################################
# Vorbelegung der Steuer-Variable
#######################################################
# Variable zum ausklinken des Subagenten (0 = beenden)
$Wert{'99.0'} = 1;    # Wert mit 1 vorbelegen
$WertTemp{'99.0'} = $Wert{'99.0'};    # Temporaeren Wert identisch belegen

######################################################
# Zusaetzliche Variablen
######################################################
$ProgPfad  = '/home/usr/stud/schuetzf/fopra/LinkTest/src';
$IniDatei  = $ProgPfad.'/LinkTest.ini'; # Ini-datei fuers Linktesten
$ProgDatei = $ProgPfad.'/LinkTest '.$IniDatei;    # Linktest-Programm
$ErrLog = $ProgPfad.'/LinkTest.ERR.LOG';

&Subagent;
```

D.2 Listing von subagent_standard.pl

```
# snmp - Subagent in Perl implementiert

sub Subagent {

#######################################################
# Beginn des Hauptprogramms
#######################################################

&initial_MIB;    # Intialisiere die MIB fuer den Subagenten
```

```
# Ermitteln des TCP-Port's, der fuer die DPI-Kommunikation
# mit dem Agenten benutzt werden soll.
$Port = &qDPIport;
print "Port:$Port\n";

# Ueber den ermittelten Port eine TCP-Verbindung zum Agenten
# aufbauen (fuer die DPI-Kommunikation) => Handle: DPI_Anschluss
&TCPConnect($Port);

# Oeffnen der DPI-Verbindung
&DPI_Open;
sleep 1;

# Registrieren des Teilbaums
&DPI_Register;

#--------------------------------------------------------------
# Warten auf eine Anfrage
while(!($Wert{'99.0'} == 0))    # Warten auf DPI-Anfragen des Agenten
    {
    if ($Wert{'99.0'} == 2)    # Falls www-Skript starten
        {        # zuerst Tabelle zuruecksetzen
        @Links = keys(%Links);
        foreach (@Links)
            {
            delete $Links{$_};
            }
        foreach (@MIB)
            {
            delete $MIB{$_};
            }
        &initial_MIB;
        $Wert{'99.0'} = 1;
        $WertTemp{'99.0'}=1;
        unless (fork)    # Parallel f starten
            {
            exec($ProgDatei);
            exit;
            }
        }

    recv(DPI_Anschluss,$Laenge,2,0);
    $Laenge = unpack("n",$Laenge);
    recv(DPI_Anschluss,$dpi_packet,$Laenge,0);
    @dpi_header = unpack("H2H2H2nH2",$dpi_packet);    # Header auslesen

    if ($dpi_header[4] eq "01")    # Falls get-Request
        {
        print "Get-Request empfangen\n";
        &answer_get;        # Beantworte get-Request
        }
    elsif ($dpi_header[4] eq "02")    # falls getnext-Request
```

```
        {
        print "Getnext-Request empfangen\n";
        &answer_getnext;    # Beantworte getnext-Request
        }
    elsif ($dpi_header[4] eq "03" || $dpi_header[4] eq "0a" || $dpi_header[4] eq "0b")
        {    # falls set-Request Gruppe
        &answer_setgrp($dpi_header[4]);
        }
    elsif ($dpi_header[4] eq "05")    # falls Response
        {
        print "Response empfangen\n";
        next;
        }
    else
        {
        print "Undefinierbare Nachricht empfangen\n";
        print "gemeldeter Packettyp: $dpi_header[5]\n";
        }
    }
#---------------------------------------------------------------
# Unregistrieren des Teilbaums
&DPI_UnRegister;

# Schliessen der DPI-Verbindung
&DPI_Close;

# Schliessen des Handles fuer die TCP-Verbindung zum Agenten
close(DPI_Anschluss);
}
########################################################
# Ende des Hauptprogramms
########################################################

############################################################
# answer_setgrp(Art)
# Zerlegen des jeweiligen Packets und wenn moeglich Beantwortung
# Parameter: Art,    Set ="03", Commit="0a", Undo="0b"
############################################################
sub answer_setgrp
    {
    # Uebernahme der Parameter
    local($Art)=@_;
    # Lokale Variablen
    local($GID,$GID_Laenge,$InstID,$InstID_Laenge,$WertTemp);

    @dpi_get = unpack("H2H2H2nH2n",$dpi_packet);
    $dpi_get_packet_id = $dpi_get[3]; # Fuer Antwort merken
    $ComNameLaenge = $dpi_get[5];
    if ($ComNameLaenge)    # Falls ein Community-Name vorhanden
        {
        $Position = 7;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2nA$dpi_get[5]"."H2" x ($Laenge-8-$ComNameLaenge);
        }
```

```
    else            # Falls kein Community-Name vorhanden
        {
        $Position = 6;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2n"."H2" x ($Laenge-8);
        }

    @dpi_get = unpack($UnpackString,$dpi_packet);
    while (!($dpi_get[$Position] eq "00")) # GID ermitteln (vorerst hex)
        {
        $GID_Laenge++;
        $NewGID = unpack("A",pack("H2",$dpi_get[$Position]));
        $GID = $GID.$NewGID;
        $Position++;
        }
    $Position++;
    while (!($dpi_get[$Position] eq "00"))
        {
        $InstID_Laenge++;
        $NewInstID = unpack("A",pack("H2",$dpi_get[$Position]));
        $InstID = $InstID.$NewInstID;
        $Position++;
        }
    $Position++;
    $Set_Typ = $dpi_get[$Position];
    $Position++;
    $Set_Laenge = hex($dpi_get[$Position])*16 + hex($dpi_get[$Position+1]);
    $Position = $Position + 2;
    for ($i = $Position; $i <= @dpi_get; $i++)
        {
        $Set_Wert[$i-$Position] = $dpi_get[$i];
        }

    &answer_set2($GID,$InstID,$Art,$dpi_get_packet_id,$Set_Typ,$Set_Laenge,
                                                                  @Set_Wert);
    }

############################################################
# answer_get
# Zerlegen des get-Packets und wenn moeglich Beantwortung
############################################################
sub answer_get
    {
    # Lokale Variablen
    local($GID,$GID_Laenge,$InstID,$InstID_Laenge);

    @dpi_get = unpack("H2H2H2nH2n",$dpi_packet);
    $dpi_get_packet_id = $dpi_get[3]; # Fuer Antwort merken
    $ComNameLaenge = $dpi_get[5];
    if ($ComNameLaenge)    # Falls ein Community-Name vorhanden
        {
        $Position = 7;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2nA$dpi_get[5]"."H2" x ($Laenge-8-
                                                          $ComNameLaenge);
```

```
        }
    else            # Falls kein Community-Name vorhanden
        {
        $Position = 6;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2n"."H2" x ($Laenge-8);
        }

    @dpi_get = unpack($UnpackString,$dpi_packet);
    while (!($dpi_get[$Position] eq "00")) # GID ermitteln (vorerst hex)
        {
        $GID_Laenge++;
        $NewGID = unpack("A",pack("H2",$dpi_get[$Position]));
        $GID = $GID.$NewGID;
        $Position++;
        }
    $Position++;
    while (!($dpi_get[$Position] eq "00"))
        {
        $InstID_Laenge++;
        $NewInstID = unpack("A",pack("H2",$dpi_get[$Position]));
        $InstID = $InstID.$NewInstID;
        $Position++;
        }
    print "$GID   ";
    print "$InstID\n";
    &answer_get2($GID,$InstID,$dpi_get_packet_id); # Auf ermittelte Anfrage antw.
    }

#############################################################
# answer_getnext
# Zerlegen des getnext-Packets und wenn moeglich Beantwortung
#############################################################
sub answer_getnext
    {
    # Lokale Variablen
    local($GID,$GID_Laenge,$InstID,$InstID_Laenge,$MIBInst);

    @dpi_get = unpack("H2H2H2nH2n",$dpi_packet);
    $dpi_get_packet_id = $dpi_get[3]; # Fuer Antwort merken
    $ComNameLaenge = $dpi_get[5];
    if ($ComNameLaenge)    # Falls ein Community-Name vorhanden
        {
        $Position = 7;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2nA$dpi_get[5]"."H2" x ($Laenge-8-$ComNameLaenge);
        }
    else            # Falls kein Community-Name vorhanden
        {
        $Position = 6;    # Beginn der GID im Array
        $UnpackString = "H2H2H2nH2n"."H2" x ($Laenge-8);
        }

    @dpi_get = unpack($UnpackString,$dpi_packet);
    while (!($dpi_get[$Position] eq "00")) # GID ermitteln (vorerst hex)
```

```
        {
        $GID_Laenge++;
        $NewGID = unpack("A",pack("H2",$dpi_get[$Position]));
        $GID = $GID.$NewGID;
        $Position++;
        }
    $Position++;
    while (!($dpi_get[$Position] eq "00"))
        {
        $InstID_Laenge++;
        $NewInstID = unpack("A",pack("H2",$dpi_get[$Position]));
        $InstID = $InstID.$NewInstID;
        $Position++;
        }
    print "$GID";
    print "   $InstID\n";

    ##################################################################
    # Gesuchten Wert ermitteln
    if (!($GID eq $GROUPID))    # Falls der Subagent fuer diese Group-ID
        {             # nicht zustaendig
        &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
        }

    $NextID = "";

    foreach $MIBInst (@MIB)
        {
        if (&less_by_hierarchy($MIBInst,$InstID) <= 0)
            {
            next;
            }

        $NextID = $MIBInst;
        last;
        }
    if ($NextID eq "")
        {
        $Next = "11";    # SNMP_TYPE_endOfMibView
        }
    else
        {
        $Next = undef;
        }
    &answer_get2($GID,$NextID,$dpi_get_packet_id,$Next);#Auf ermittelte Anfr. antw.
    }

######################################################################
# ResponseSuccess(Paket-Id,GID,InstID,Typ,Wert)
# Aufbauen eines Response-Pakets als Antwort auf eine Anfrage.
# Parameter:    Paket-Id, Paket-ID der Anfrage
#        GID,      Group-ID nach der gefragt wurde
#        InstID,   Instanz-ID nach der gefragt wurde
```

```
#       Typ,      Typ des Rueckgabewerts, vgl. RFC1592,Tab.17 aber hex
#       Wert,      Rueckgabewert
########################################################################
sub ResponseSuccess
    {
    # Uebernahme der Parameter
    local($dpi_Packet_ID,$GID,$InstID,$Wert) = @_;

    if(!($Wert eq "ENDofMIB"))
        {
        $Typ = $MIB{$InstID};
        }
    else
        {
        $Typ = "11";
        }

    # Bestimmen der Laengen der Strings (auch abhaengig vom Typ)
    $GID_Laenge = length($GID);
    $Inst_Laenge = length($InstID);
    if ($Typ eq "02")    # Falls Typ = SNMP_TYPE_OCTET_STRING
        {
        $Wert_Laenge = length($Wert);
        $Wert_PackString = "A".$Wert_Laenge;
        }
    elsif ($Typ eq "03")    # Falls Typ = SNMP_TYPE_OBJECT_IDENTIFIER
        {
        $Wert_Laenge = length($Wert);
        $Wert_PackString = "A".$Wert_Laenge;
        }
    elsif ($Typ eq "11")    # Falls Typ = SNMP_TYPE_endOfMibView
        {
        print "EndofMib\n";
        $Wert_Laenge = 0;
        $Wert_PackString = "";
        }
    elsif ($Typ eq "81")    # Falls Typ = SNMP_TYPE_Integer32
        {
        $Wert_Laenge = 4;
        $Wert_PackString = "N";
        }
    else            # Falls unbekannter Typ
        {
        print "Unbekannter Typ: $Typ\n";
        return;
        }

    # Aufbauen des Pack-Strings
    $PackString = "H2H2H2nH2H2H8A".$GID_Laenge."xA".$Inst_Laenge."xH2n".
                                                         $Wert_PackString;

    #####################################################
    # Paket ohne Feld Paketlaenge zusammensetzen
```

```
    $dpi_Packet = pack($PackString,
                # Head-Beginn
                "02",    # Protokol Haupt Version
                "02",    # Protokol Niedere Version
                "00",    # Protokol Release
                $dpi_Packet_ID,
                "05",    # Typ: Response-Packet
                # Head-Ende
                "00",    # Fehlercode
                "00000000",    # Fehlerindex
                $GID,    # Group-ID
                $InstID,# Instanz-ID
                $Typ,    # Typ der Variablen
                $Wert_Laenge,    # Laenge des Werts
                $Wert    # Rueckgabewert
               );
    #####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet);        # Laenge des Packets ermitteln
    $dpi_Packet_Length = pack("n",$Laenge);# Laenge des Packets in
                                           # eigenes Paket legen

    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Success-Response verschickt\n";
    }

######################################################################
# SendTrap(GenTrapCode,SpecTrapCode,Enterprise,GID,InstID,Typ,Wert)
# Aufbauen eines Trap-Pakets und senden desselben.
# Parameter:    GenTrapCode,     Allgemeiner Trap Code
#        SpecTrapCode,    Spezieller Trap Code
#        Enterprise,    Optionale Enterprise-ID
#        GID,          Group-ID der Trap-Variablen
#        InstID,       Instanz-ID der Trap-Variablen
#        Typ,          Typ des Rueckgabewerts, vgl. RFC1592,Tab.17 aber hex
#        Wert,          Rueckgabewert
######################################################################
sub SendTrap
    {
    # Uebernahme der Parameter
    local($GenTrapCode,$SpecTrapCode,$EnterpID,$GID,$InstID,$Typ,$Wert) = @_;

    $dpi_Packet_ID++;    # Packet-ID hochzaehlen
    # Bestimmen der Laengen der Strings (auch abhaengig vom Typ)
    $EnterpID_Laenge = length($EnterpID);
    $GID_Laenge = length($GID);
    $Inst_Laenge = length($InstID);
    if ($Typ eq "02")    # Falls Typ = SNMP_TYPE_OCTET_STRING
        {
        $Wert_Laenge = length($Wert);
        $Wert_PackString = "A".$Wert_Laenge;
        }
```

```
elsif ($Typ eq "03")    # Falls Typ = SNMP_TYPE_OBJECT_IDENTIFIER
    {
    $Wert_Laenge = length($Wert);
    $Wert_PackString = "A".$Wert_Laenge;
    }
elsif ($Typ eq "11")    # Falls Typ = SNMP_TYPE_endOfMibView
    {
    print "EndofMib\n";
    $Wert_Laenge = 0;
    $Wert_PackString = "";
    }
elsif ($Typ eq "81")    # Falls Typ = SNMP_TYPE_Integer32
    {
    $Wert_Laenge = 4;
    $Wert_PackString = "N";
    }
else            # Falls unbekannter Typ
    {
    print "Unbekannter Typ: $Typ\n";
    return;
    }
if ($EnterpID)
    {
    # Aufbauen des Pack-Strings mit EnterpriseID
    $PackString = "H2H2H2nH2nnA".$EnterpID_Laenge."xA".$GID_Laenge."xA".
                                   $Inst_Laenge."xH2n".$Wert_PackString;
    }
else
    {
    # Aufbauen des Pack-Strings ohne EnterpriseID
    $PackString = "H2H2H2nH2nnA1A".$GID_Laenge."xA".$Inst_Laenge."xH2n".
                                   $Wert_PackString;
    $EnterpID = "\0";
    }
#####################################################
# Paket ohne Feld Paketlaenge zusammensetzen
$dpi_Packet = pack($PackString,
            # Head-Beginn
            "02",    # Protokol Haupt Version
            "02",    # Protokol Niedere Version
            "00",    # Protokol Release
            $dpi_Packet_ID,
            "04",    # Typ: Trap-Packet
            # Head-Ende
            $GenTrapCode,    # Allgemeiner Trap-Code
            $SpecTrapCode,    # Spezieller Trap-Code
            $EnterpID,    # optional Enterprise ID
            $GID,    # Group-ID
            $InstID,# Instanz-ID
            $Typ,    # Typ der Variablen
            $Wert_Laenge,    # Laenge des Werts
            $Wert    # Rueckgabewert
           );
```

```
    ####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet);        # Laenge des Packets ermitteln
    $dpi_Packet_Length = pack("n",$Laenge); # Laenge des Packets in
                                            # eigenes Paket legen

    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Success-Response verschickt\n";
    }

#############################################################
# ResponseError(Paket-Id,Fehlercode in hex)
# Aufbauen eines Response wegen Fehler und verschicken dieses.
# Parameter:    Paket-Id, Paket-id der Anfrage, die fehlerhaft
#              war
#       Fehlercode in hex, vgl. rfc1592 Tab. 18
#############################################################
sub ResponseError
    {
    # Uebernahme der Parameter
    local($dpi_Packet_ID,$Error) = @_;

    ####################################################
    # Paket ohne Feld Paketlaenge zusammensetzen
    $dpi_Packet = pack("H2H2H2nH2H2n",
                # Head-Beginn
                "02",    # Protokol Haupt Version
                "02",    # Protokol Niedere Version
                "00",    # Protokol Release
                $dpi_Packet_ID,
                "05",    # Typ: Response-Packet
                # Head-Ende
                $Error,    # Fehlercode
                1,    # Fehler bei VarBind 1
               );
    ####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet);        # Laenge des Packets ermitteln
    $dpi_Packet_Length = pack("n",$Laenge); # Laenge des Packets in
                                            # eigenes Paket legen

    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Error-Response verschickt\n";
    }

############################################################
# DPI_UnRegister
# Aufbauen eines SNMP_DPI_UnRegister Request Packets und
# verschicken desselben.
############################################################
sub DPI_UnRegister
```

```
{
$dpi_Packet_ID++;     # Packet-Identifier hochzaehlen
$GID_Laenge = length($GROUPID);
$PackString = "H2H2H2nH2H2A".$GID_Laenge."x";

####################################################
# Paket ohne Feld Paketlaenge zusammensetzen
$dpi_Packet = pack($PackString,
            # Head-Beginn
            "02",    # Protokol Haupt Version
            "02",    # Protokol Niedere Version
            "00",    # Protokol Release
            $dpi_Packet_ID,
            "07",    # Typ: Unregister-Packet
            # Head-Ende
            "02",    # Grund: Programmende
            $GROUPID    # GID ab der der Subagent zustaendig
           );
####################################################
# Paketlaenge ermitteln
$Laenge = length($dpi_Packet);    # Laenge des Packets ermitteln
         # (+1 wegen des zusaetzl. Feldes Paketlaenge)
$dpi_Packet_Length = pack("n",$Laenge);    # Laenge des Packets in
                                           # eigenes Paket legen

if($child = fork)
    {
    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Unregister-Request: ";
    }
else
    {
    # SNMP-DPI Response entgegennehmen und auswerten
    &DPI_OpenResponse;
    }
waitpid($child,0);
$Return = $?>>8 ;
if (!($Return == 0))     # Falls die Unregister-Request nicht angenommen
    {
    print "nicht angenommen!\n";
    print "Fehlercode: $Return\n";
    &DPI_Close;    # Schliessen der DPI-Verbindung
    close(DPI_Anschluss); # Schliessen des Handles fuer die TCP-
                          # Verbindung zum Agenten
    die "\n";
    }
else
    {
    print "angenommen\n";
    }
}
```

```
############################################################
# DPI_Register
# Aufbauen eines SNMP_DPI_Register Request Packets und
# verschicken desselben.
############################################################
sub DPI_Register
    {
    $dpi_Packet_ID++;     # Packet-Identifier hochzaehlen
    $GID_Laenge = length($GROUPID);
    $PackString = "H2H2H2nH2nnH2H2H2H2A".$GID_Laenge."x";

    ####################################################
    # Paket ohne Feld Paketlaenge zusammensetzen
    $dpi_Packet = pack($PackString,
                # Head-Beginn
                "02",    # Protokol Haupt Version
                "02",    # Protokol Niedere Version
                "00",    # Protokol Release
                $dpi_Packet_ID,
                "06",    # Typ: Register-Packet
                # Head-Ende
                0,    # Prioritaet
                5,    # Time-Out in sek.
                "00",    # Authentifikation von Anfragen:Agent
                "0a",
                "00",    # GetBulk wird zu GetNext
                "00",
                $GROUPID    # GID ab der der Subagent zustaendig,
               );
    ####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet);    # Laenge des Packets ermitteln
                                      # (+1 wegen des zusaetzl.
                                      # Feldes Paketlaenge)
    $dpi_Packet_Length = pack("n",$Laenge);    # Laenge des Packets
                                               # in eigenes Paket legen

    if($child = fork)
        {
        # Pakete verschicken
        print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
        print "Register-Request: ";
        }
    else
        {
        # SNMP-DPI Response entgegennehmen und auswerten
        &DPI_OpenResponse;
        }
    waitpid($child,0);
    $Return = $?>>8 ;
    if (!($Return == 0))     # Falls die Unregister-Request nicht angenommen
        {
        print "nicht angenommen!\n";
```

```
        print "Fehlercode: $Return\n";
        &DPI_Close;    # Schliessen der DPI-Verbindung
        close(DPI_Anschluss); # Schliessen des Handles fuer die
                              # TCP-Verbindung zum Agenten
        die "\n";
        }
    else
        {
        print "angenommen\n";
        }
    }

############################################################
# DPI_AreYouThere
# Aufbauen eines SNMP_DPI_Are_You_There Request Packets und
# verschicken desselben.
############################################################
sub DPI_AreYouThere
    {
    $dpi_Packet_ID++;     # Packet-Identifier hochzaehlen

    ####################################################
    # Paket ohne Feld Paketlaenge zusammensetzen
    $dpi_Packet = pack("H2H2H2nH2",
                # Head-Beginn
                "02",    # Protokol Haupt Version
                "02",    # Protokol Niedere Version
                "00",    # Protokol Release
                $dpi_Packet_ID,
                "15",    # Typ: AreYouThere-Packet
                # Head-Ende
               );
    ####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet);    # Laenge des Packets ermitteln
                                      # (+1 wegen des zusaetzl.
                                      # Feldes Paketlaenge)
    $dpi_Packet_Length = pack("n",$Laenge); # Laenge des Packets
                                            # in eigenes Paket legen

    if($child = fork)
        {
        # Pakete verschicken
        print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
        print "Are-You_There_Request verschickt\n";
        }
    else
        {
        # SNMP-DPI Response entgegennehmen und auswerten
        &DPI_OpenResponse;
        }
    waitpid($child,0);
    $Return = $?>>8 ;
```

```
    if (!($Return == 0))     # Falls die Open-Request nicht angenommen
        {
        print "Verbindung zum Agenten nicht in Ordnung!\n";
        print "Fehlercode: $Return\n";
        die "\n";
        }
    else
        {
        print "Verbindung zum Agenten in Ordnung!\n";
        }
    }

############################################################
# DPI_Close
# Aufbauen eines SNMP_DPI_Close Request Packets und
# verschicken desselben.
############################################################
sub DPI_Close
    {
    $dpi_Packet_ID++;     # Packet-Identifier hochzaehlen

    ####################################################
    # Paket ohne Feld Paketlaenge zusammensetzen
    $dpi_Packet = pack("H2H2H2nH2H2",
                # Head-Beginn
                "02",    # Protokol Haupt Version
                "02",    # Protokol Niedere Version
                "00",    # Protokol Release
                $dpi_Packet_ID,
                "09",    # Typ: Close-Packet
                # Head-Ende
                "02"    # Grund: Programmende
               );
    ####################################################
    # Paketlaenge ermitteln
    $Laenge = length($dpi_Packet)+1;    # Laenge des Packets ermitteln
                                        # (+1 wegen des zusaetzl.
                                        # Feldes Paketlaenge)
    $dpi_Packet_Length = pack("n",$Laenge); # Laenge des Packets in
                                            # eigenes Paket legen
    ####################################################
    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Close-Request verschickt\n";
    }

############################################################
# DPI_Open
# Aufbauen eines SNMP_DPI_Open Request Packets und
# verschicken desselben.
############################################################
sub DPI_Open
    {
```

```
$dpi_Packet_ID++;     # Packet-Identifier hochzaehlen
$OID_Laenge = length($OID);
$DESCRIPTION_Laenge = length($DESCRIPTION);
$PackString = "H2H2H2nH2nnH2A".$OID_Laenge."xA".$DESCRIPTION_Laenge."xn";

####################################################
# Paket ohne Feld Paketlaenge zusammensetzen
$dpi_Packet = pack($PackString,
            # Head-Beginn
            "02",    # Protokol Haupt Version
            "02",    # Protokol Niedere Version
            "00",    # Protokol Release
            $dpi_Packet_ID,
            "08",    # Typ: Open-Packet
            # Head-Ende
            5,    # Time-Out in Sekunden
            1,    # Maximum VarBinds pro Packet
            "01",    # Schriftsatz: eigener
            $OID,    # OID ab der der Subagent zustaendig,
            $DESCRIPTION, # Erklaerung zum Subagenten
            0    # Passwort-Laenge ist Null
           );
####################################################
# Paketlaenge ermitteln
$Laenge = length($dpi_Packet)+1; # Laenge des Packets ermitteln
                                 # (+1 wegen des zusaetzl.
                                 # Feldes Paketlaenge)
$dpi_Packet_Length = pack("n",$Laenge);# Laenge des Packets
                                       # in eigenes Paket legen

if($child = fork)
    {
    # Pakete verschicken
    print DPI_Anschluss $dpi_Packet_Length.$dpi_Packet;
    print "Open-Request: ";
    }
else
    {
    # SNMP-DPI Response entgegennehmen und auswerten
    &DPI_OpenResponse;
    }
waitpid($child,0);
$Return = $?>>8 ;
if (!($Return == 0))     # Falls die Open-Request nicht angenommen
    {
    print "nicht angenommen!\n";
    print "Fehlercode: $Return\n";
    die "\n";
    }
else
    {
    print "angenommen\n";
    }
```

```
    }

############################################################
# DPI_OpenRespone
# Auf Response-Paket auf Open- bzw. AreYouThere-Request warten
# und auf Erfolg des Requests ueberpruefen.
############################################################
sub DPI_OpenResponse
    {
    # Lokale Variablen
    local($Reponse,@Response);

    recv(DPI_Anschluss,$Response,15,0);

    @Response = unpack("nH2H2H2nH2H2H2H2H2H2",$Response);

    if($Response[5] eq "05" && $Response[6] eq "00" )
        {
        exit;
        }
    else
        {
        exit(hex($Response[6]));
        }
    }


############################################################
# TCPConnect($Port)
# Aufbauen einer TCP-Verbindung zum, durch die globale Var.
# $AGENT_HOST gegebenen Rechner ueber Port $Port.
# Bei Erfolg steht DPI_Anschluss als Handle zur Verfuegung.
############################################################
sub TCPConnect
    {
    # Uebernahme der Parameter
    local($Port) = @_ ;

    print "TCPConnect():";
    # Name des Rechners, von dem aus die Verbindung
    # aufgebaut wird
    $hostname   = `hostname`;

    # Service fuer die Anbindung aufbereiten
    ($name,$aliases,$proto) = getprotobyname('tcp');
    ($name,$aliases,$Port) = getservbyname($Port,'tcp')
    unless $Port =~ /^\d+$/;

    ($name, $aliases, $type, $len, $thisaddr) = gethostbyname($hostname);
    $this = pack($SOCKADDR, $AF_INET, 0, $thisaddr);

    # Gegenadresse auf gewuenschte Domaine setzen
    ($name, $aliases, $type, $len, $thataddr) = gethostbyname($AGENT_HOST);
```

```
    $that = pack($SOCKADDR, $AF_INET, $Port, $thataddr);

        # Create a handle to the socket
      socket(DPI_Anschluss, $AF_INET, $SOCK_STREAM, $proto) ||
                                          die "Socket fehlgeschlagen\n";

      # Assign the socket an address
      bind(DPI_Anschluss, $this) || die "Bind fehlgeschlagen\n";

      # Connect to the server
      connect(DPI_Anschluss,$that) || die "Connect fehlgeschlagen\n";

    select(DPI_Anschluss);
    $| = 1;     # Auf nicht puffern stellen
    select(STDOUT);
    print " ok\n";
    }

#######################################################
# qDPIport
# Unterprogramm zum Ermitteln des Port's, ueber den der
# Subagent mit dem Agenten kommunizieren soll.
# Benoetigt den Hostnamen des Agenten in der globalen
# Variablen $AGENT_HOST
# Rueckgabe: TCP-Port der fuer die weitere DPI-Verbindung
#         vorgeschlagen wurde.
#######################################################
sub qDPIport
    {
    # Definition von lokalen Variablen
    local($snmpPacket,@snmpPacket);

    print "qDPIport: ";
    # Paket zusammensetzen
    $snmpPacket = pack( "H2H2H6H2H2A6H4H6H6H6H4H4H4H22H4",
                "30",     # ASN.1 Header in Hex
                "29",     # PDU-length
                "020100",# snmp Version
                "04",     # Feld:Community-Name
                "06",     # Laenge Community-Name
                "public",# Community-Name
                "a01c",     # snmp-Get-Request
                "020101",# snmp-Request-Id
                "020100",# snmp-error-Status
                "020100",# snmp-Index
                "3011",     # varBind list
                "300f",     # varBind
                "060b",     # Feld:Object-Id
                "2b060104010202010100", # Object-Id
                "0500"     # null-Value
                  );
```

```
    ######################################
    # UDP-Verbindung zu Agenten aufbauen

    # Name des Rechners, auf dem der Subagent gestartet wird
    $hostname   = `hostname`;

    ($name, $aliases, $type, $len, $addr) = gethostbyname($hostname);
    $this = pack($SOCKADDR, $AF_INET, 0, $addr);
    ($name, $aliases, $type, $len, $thataddr) = gethostbyname($AGENT_HOST);
    $that = pack($SOCKADDR, $AF_INET, $SNMP_PORT, $thataddr);

        # Create a handle to the socket
      socket(SNMP_Anschluss, $AF_INET, $SOCK_DGRAM, 17) ||
                                              die "Socket fehlgeschlagen\n";

      # Assign the socket an address
      bind(SNMP_Anschluss, $this) || die "Bind fehlgeschlagen\n";

    ###########################################
    # SNMP-Get-Request-Paket schicken
    send(SNMP_Anschluss,$snmpPacket,0,$that);

    ################################################################
    # SNMP-Response-Paket empfangen die noetigen Informationen
    # ermitteln und zurueckgeben.
    recv(SNMP_Anschluss,$snmp_Packet,54,0);
    close SNMP_Anschluss;
    @snmp_Packet = unpack("H2H6H6H2H2A6H8H6H6H6H8H8H4H22H4n",$snmp_Packet);

    print "ok\n";
    return($snmp_Packet[15]);
    }

##################################################
# Vergleichsfunktion zum Sortieren
##################################################
sub by_hierarchy
    {
    local($i,@a,@b);

    @a = split(/\./,$a);
    @b = split(/\./,$b);

    if (@a < @b)
        {
        for ($i=0;$i<@a;$i++)
            {
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
```

```
                return($a[$i]-$b[$i]);
                }
            }
        return(-1);
        }
    elsif (@a > @b)
        {
        for ($i=0;$i<@b;$i++)
            {
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
                return($a[$i]-$b[$i]);
                }
            }
        return(1);
        }
    else
        {
        for ($i=0;$i<@b;$i++)
            {
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
                return($a[$i]-$b[$i]);
                }
            }
        return(0);
        }
    }

###################################################
# Vergleichsfunktion
###################################################
sub less_by_hierarchy
    {
    # Uebernahme der Parameter
    local($a,$b)=@_;
    local($i,@a,@b);

    @a = split(/\./,$a);
    @b = split(/\./,$b);

    if (@a < @b)
        {
        for ($i=0;$i<@a;$i++)
            {
```

```
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
                return($a[$i]-$b[$i]);
                }
            }
        return(-1);
        }
    elsif (@a > @b)
        {
        for ($i=0;$i<@b;$i++)
            {
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
                return($a[$i]-$b[$i]);
                }
            }
        return(1);
        }
    else
        {
        for ($i=0;$i<@b;$i++)
            {
            if ($a[$i] == $b[$i])
                {
                next;
                }
            else
                {
                return($a[$i]-$b[$i]);
                }
            }
        return(-1);
        }
    }


1;
```

D.3 Listing von mib_gets.pl

```
#############################################################
# answer_get2
# Beantwortung eines get's oder getnext's
```

```
#############################################################
sub answer_get2
    {
    # Parameter uebernehmen
    local($GID,$InstID,$dpi_get_packet_id,$Next)=@_;
    # Lokale Variablen
    local($GID_Laenge,$InstID_Laenge,@Wert);
    #################################################################
    # Gesuchten Wert ermitteln
    if (!($GID eq $GROUPID))    # Falls der Subagent fuer diese Group-ID
        {           # nicht zustaendig
        &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
        }

    if ($Next)    # End of MIB bei getnext
        {
        print "EndofMIB\n";
        $Wert = "ENDofMIB";
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        $Next = undef;
        }
    elsif ($InstID eq '99.0')
        {
        $Wert = &get_99;    # Funktion fuer Instanz 99 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '1.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 1 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '2.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 2 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '3.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 3 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '4.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 4 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '5.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 5 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '6.0')
        {
```

```
        $Wert = &get($InstID);    # Funktion fuer Instanz 6 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '7.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 7 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '8.0')
        {
        $Wert = &get($InstID);    # Funktion fuer Instanz 8 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID eq '9.0')
        {
        $Wert = &get_9;    # Funktion fuer Instanz 9 aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID =~ /^10\.1\.1/)
        {
        $Wert = &get_Tabelle($InstID);    # Funktion fuer Tabelle aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID =~ /^10\.1\.2/)
        {
        $Wert = &get_Tabelle($InstID);    # Funktion fuer Tabelle aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID =~ /^10\.1\.3/)
        {
        $Wert = &get_Tabelle($InstID);    # Funktion fuer Tabelle aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }
    elsif ($InstID =~ /^10\.1\.4/)
        {
        $Wert = &get_Tabelle($InstID);    # Funktion fuer Tabelle aufrufen
        &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
        }

    else        # Variable nicht vorhanden
        {
        &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
        }
    }

1;
```

D.4 Listing von mib_sets.pl

```
#############################################################
# answer_set2
```

```
# Beantwortung eines Set-, Commit-, Undo-Requests
#############################################################
sub answer_set2
    {
    # Uebernahme der Parameter
    local($GID,$InstID,$Art,$dpi_get_packet_id,$Set_Typ,$Set_Laenge,@Set_Wert)=@_;

    if (!$GID eq $GROUPID)    # Falls der Subagent fuer diese Group-ID
        {        # nicht zustaendig
        &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
        }
    # Set-Variablen
    if ($InstID eq "99.0")
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 99 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com_99;    # Commit-Funktion fuer Instanz 99 aufrufen
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion fuer Instanz 99 aufrufen
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '1.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
```

```
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '2.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '3.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 3 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '4.0')
        {
```

```
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 4 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '5.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '6.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
```

```
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '7.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '8.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
```

```
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    elsif  ($InstID eq '9.0')
        {
        if ($Art eq "03")    # Set - Request
            {    # Funktion fuer Instanz 2 aufrufen
            $Fehler = &set($InstID,$Set_Typ,$Set_Laenge,@Set_Wert);
            }
        elsif ( $Art eq "0a")    # Commit - Request
            {
            $Fehler = &com($InstID);    # Commit-Funktion
            }
        elsif ( $Art eq "0b")    # Undo - Request
            {
            $Fehler = &undo($InstID);    # Undo-Funktion
            }
        if(!$Fehler)
            {
            &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Set_Typ,$Wert{$InstID});
            }
        else
            {
            &ResponseError($dpi_get_packet_id,$Fehler) # nicht Schreibbar
            }
        }
    else        # Keine Set-Variable vorhanden
        {
        &ResponseError($dpi_get_packet_id,"11") # nicht Schreibbar
        }
    }
1;
```

D.5 Listing von sub_gets.pl

```
#################################################################
# get_Tabelle Funktion fuer Uebergabe der Tabelle;
# Parameter: InstID
# Rueckgabe: Wert einer Variablen aus der Tabelle
#################################################################
sub get_Tabelle
    {
    # Uebernahme der Parameter
    local($InstID)=@_;
    # Lokale Variablen
    local($Wert,@Wert,$ZeilenNr,$VarNr);
```

```
##################################################
# Welche Tabellenspalte(=VarNr) und welche Zeile
$ZeilenNr= substr($InstID,rindex($InstID,'.')+1);
$InstID = substr($InstID,0,rindex($InstID,'.'));
$VarNr= substr($InstID,rindex($InstID,'.')+1);
$InstID = substr($InstID,0,rindex($InstID,'.'));

#################################################
# Falls noetig, Tabelle in den Speicher holen
if (!keys(%Links))
    {
    print "Erstelle Tabelle\n";
    # Falls Tabelle noch nicht im Speicher ex.
    if(!open(LOGERR,$ErrLog))
        {
        return("Konnte Fehler-Log-Datei nicht oeffnen!");
        }
    while (<LOGERR>)
        {
        if (/^Intern --:/)
            {
            $Beschreibung = substr($_,11);
            $Zeile =substr($Beschreibung,index($Beschreibung,";")+1);
            $Index =substr($Beschreibung,0,index($Beschreibung,";"));
            $Links{$Index}=$Zeile;
            }
        else
            {
            next;
            }
        }
    close (LOGERR);
    @Links = keys(%Links);
    }
#################################################
# Ausgeben des gewuenschten Tabelleneintrags
if ($VarNr == 1)
    {    # Falls Index
    $Wert = $Links[$ZeilenNr];
    }
else
    {    # Falls kein Indexfeld
    $Zeile = $Links{$Links[$ZeilenNr]};
    @Wert = split(/;/,$Zeile);
    $Wert = $Wert[$VarNr-2];    # -2, da von 0 gezaehlt und Index extra
    }
##################################################
# Falls es eine naechste Zeile in der Tabelle gibt,
# die noch nicht in der internen MIB aufgenommen ist,
# aufnehmen
if ($Links[$ZeilenNr+1] && !($MIB{$InstID.'.'.$VarNr.'.'.($ZeilenNr+1)}))
    {    # Falls Folgezeile ex. nimm Sie in MIB auf
    $MIB{$InstID.'.'.$VarNr.'.'.($ZeilenNr+1)}=$MIB{$InstID.'.'.$VarNr.'.'."0"};
```

```
        @MIB = sort by_hierarchy keys(%MIB);    # sortierte Liste der MIB-Variablen
        }

    return($Wert);    # wert zurueckgeben
    }

#######################################################################
# get Funktion zum Bestimmen der jeweiligen Variablen aus der ini.Datei
# Parameter : Nummer der Variablen
# Rueckgabe : Typ der Variablen, vgl. rfc1592 Tab. 17, aber hex
#          Wert der Variablen
#######################################################################
sub get
    {
    # Uebernahme des Parameters
    local($InstID) = @_;
    # Lokale Variablen
    local($Fehler);
    $Fehler = undef;

    # Oeffnen der Ini-Datei
    if(!open(INIDATEI,$IniDatei))
        {
        $Fehler="Konnte Initialisierungsdatei $IniDatei nicht oeffnen!";
        }
    if ($InstID eq '1.0')
        {
        $Ausdruck = '\[NACHRICHT_AN\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '2.0')
        {
        $Ausdruck = '\[WWW_SERVER\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '3.0')
        {
        $Ausdruck = '\[TOP_LINK\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '4.0')
        {
        $Ausdruck = '\[MAX_LINKS\]=';
        $Wert = "0";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '5.0')
        {
        $Ausdruck = '\[BETREUER_TAG\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '6.0')
        {
        $Ausdruck = '\[BETREUER_ADRESSE\]=';
```

```
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '7.0')
        {
        $Ausdruck = '\[ENDUNGEN\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    elsif ($InstID eq '8.0')
        {
        $Ausdruck = '\[NACHRICHT\]=';
        $Wert = "Fehler";    # Um Fehler zu erkennen
        }
    else
        {
        $Fehler = 0;
        }

    #########################################################
    # Parsen der Ini-Datei nach dem gesuchten Eintrag
    while(<INIDATEI>)
        {
        if (/^((\s*#)|(\s+))/)        # Falls Kommentar- oder Leerzeile:
            {
            next;            # ueberspringen
            }
        elsif (/$Ausdruck/)
            {
            /=(.*?);/;
            $Wert = $1;
            last;
            }
        }
    close(INIDATEI);

    if (!$Fehler)
        {
        return($Wert);
        }
    else
        {
        return($Fehler);
        }
    }

#################################################################
# get_9 Funktion den Nachricht-Text an Instanz 9;
# Rueckgabe : Wert der Variablen
#################################################################
sub get_9
    {

    if(!open(INIDATEI,$IniDatei))
        {
```

```
        $Fehler="Konnte Initialisierungsdatei $IniDatei nicht oeffnen!";
        }
    while(<INIDATEI>)    # Durchsuche ini.Datei
        {
        $INI=$_;
        if ($InNachrichtText eq "Ja")
            {
            if ($INI=~s/>>;//)    # Falls Endezeichen zu Text in der Zeile enthalten,
                {        # loesche dieses und ...
                $Wert9 = join("\n",$Wert9,$INI);
                $InNachrichtText = "Nein";
                next;
                }
            else
                {
                $Wert9 = join("\n",$Wert9,$INI);
                next;
                }
            }
        if ($INI=~/^((\s*#)|(\s+))/)        # Falls Kommentar- oder Leerzeile:
            {
            next;            # ueberspringen
            }
        elsif ($INI=~/\[NACHRICHT_TEXT\]=<<(.*)/)    # Hole 1. Zeile des Mailtextes
            {
            $Wert9 = $1;
            $InNachrichtText = "Ja";
            next;
            }
        }
    close(INIDATEI);
    return($Wert9);
    }

#################################################################
# get_99 Funktion fuer die Kontroll-Variable an Instanz 99;
# Rueckgabe: Wert der Variablen
#################################################################
sub get_99
    {
    return($Wert{'99.0'});
    }

1;
```

D.6 Listing von sub_sets.pl

```
#################################################################
# set(Typ,Laenge,Wert,InstID) Set-Funktion fuer den WWW-Server
# Parameter : Typ,    gibt den Typ des zu setzenden Werts an
#          Laenge,    gibt die Laenge des Werts an
```

```
#           Wert,    der Wert selbst, als hex-Array
#          InstID    welches Set?
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
################################################################
sub set
    {
    # Uebernahme der Parameter
    local($InstID,$SetTyp,$SetLaenge,@SetWert)=@_;
    # Lokale Variablen
    local($Temp);

    if (!($SetTyp eq $MIB{$InstID}))    # Typ ueberpruefen
        {
        return("07");
        }
    elsif ($SetTyp eq "02")
        {
        foreach (@SetWert)
            {
            $NewWert = unpack("A",pack("H2",$_));
            $Temp = $Temp.$NewWert;
            }
        $WertTemp{$InstID} = $Temp;
        }
    elsif ($SetTyp eq "81")
        {
        $WertTemp{$InstID} = unpack("N",pack("H2H2H2H2",$SetWert[0],$SetWert[1],
                                                 $SetWert[2],$SetWert[3]));
        }
    print "Wert vorlaeufig: $WertTemp{$InstID}\n";
    return(undef);
    }

1;
```

D.7 Listing von sub_coms.pl

```
################################################################
# com Funktion um WWW-Server zu setzen
# Parameter : Instant-ID um Variable zu identifizieren
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
################################################################
sub com
    {
    # Uebernahme der Parameter
    local($InstID)=@_;

    # Oeffnen der Ini-Datei
    if(!open(INIDATEI,$IniDatei))
        {
        print "Fehler beim Oeffnen von $IniDatei\n";
```

```
        return("05");
        }
    # Oeffnen einer temporaeren Datei
    if(!open(TEMPDATEI,">temp.tmp"))
        {
        print "Fehler beim Oeffnen der Temp-Datei\n";
        return("05");
        }
    if ($InstID eq '1.0')
        {
        $Ausdruck = '\[NACHRICHT_AN\]=';
        $Ausdruck2 ='[NACHRICHT_AN]=';
        }
    elsif ($InstID eq '2.0')
        {
        $Ausdruck = '\[WWW_SERVER\]=';
        $Ausdruck2 = '[WWW_SERVER]=';
        }
    elsif ($InstID eq '3.0')
        {
        $Ausdruck = '\[TOP_LINK\]=';
        $Ausdruck2 = '[TOP_LINK]=';
        }
    elsif ($InstID eq '4.0')
        {
        $Ausdruck = '\[MAX_LINKS\]=';
        $Ausdruck2 = '[MAX_LINKS]=';
        }
    elsif ($InstID eq '5.0')
        {
        $Ausdruck = '\[BETREUER_TAG\]=';
        $Ausdruck2 = '[BETREUER_TAG]=';
        }
    elsif ($InstID eq '6.0')
        {
        $Ausdruck = '\[BETREUER_ADRESSE\]=';
        $Ausdruck2 = '[BETREUER_ADRESSE]=';
        }
    elsif ($InstID eq '7.0')
        {
        $Ausdruck = '\[ENDUNGEN\]=';
        $Ausdruck2 = '[ENDUNGEN]=';
        }
    elsif ($InstID eq '8.0')
        {
        $Ausdruck = '\[NACHRICHT\]=';
        $Ausdruck2 = '[NACHRICHT]=';
        }
    else
        {
        return("14");
        }
    ##########################################################
```

```
    # Ersetzen der Variablen in der Ini-Datei
    while(<INIDATEI>)
        {
        if (/$Ausdruck/)
            {    # Wert ersetzen
            $Wert{$InstID}=$WertTemp{$InstID};
            print TEMPDATEI $Ausdruck2.$Wert{$InstID}.";\n";
            }
        else
            {    # gelesenes 1 zu 1 in temporaere Datei
            print TEMPDATEI $_;
            }
        }
    close(INIDATEI);
    close(TEMPDATEI);
    unlink($IniDatei);
    rename("temp.tmp",$IniDatei);
    return(undef);
    }


#################################################################
# com_99 Pseudofunktion fuer die Variable an Instanz 99
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub com_99
    {
    $Wert{'99.0'} = $WertTemp{'99.0'};
    return(undef);
    }

1;
```

D.8 Listing von sub_undos.pl

```
#################################################################
# undo Funktion zum Abbrechen einer Variablen-Aenderung
# Parameter : Instance-ID zum Identifizieren der Variablen
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub undo
    {
    # Uebernahme der Parameter
    local($InstID)=@_;
    $WertTemp{$InstID} = $Wert{$InstID};
    return(undef);
    }

1;
```

Anhang E

Listings der Beispiel-Dateien

E.1 Listing von subagent_start

```
#!/sw/mnm/perl5/bin/perl

require "sub_gets.pl";
require "sub_sets.pl";
require "sub_undos.pl";
require "sub_coms.pl";
require "mib_gets.pl";
require "mib_sets.pl";
require "subagent_standard.pl";

#######################################################
# Definition einiger Konstanten
#######################################################
# Konstanten fuer den Socket
$AF_INET = 2;
$SOCK_STREAM = 1;
$SOCK_DGRAM  = 2;
$SOCKADDR = 'S n a4 x8';

#Konstanten fuer die SNMP-Kommunikation zum Agenten
$SNMP_PORT     = 161 ;
$SNMP_TRAP_PORT    = 162 ;
$SNMP_COMMUNITY    = "public" ;
$SNMP_TRAP_COMMUNITY    = "public";

#Konstanten fuer die DPI-Kommunikation
$GROUPID    = "1.3.6.1.3.100.7.";
$OID        = "1.3.6.1.3.100.7";
$DESCRIPTION    = "Beispiel-Subagent";

########################################################
# Definition globaler Variablen
########################################################
$AGENT_HOST    = "sunhegering8" ;    # Host des Agenten

########################################################
```

```
# Initialisierungsroutine fuer die MIB-Variablen, die
# der Subagent anbietet.
########################################################
sub initial_MIB
    {
        $MIB{'1.0'} = "02";     # Variable 1: Text
        $MIB{'2.0'} = "81";     # Variable 2: Integer
        $MIB{'3.1.1.0'} = "02";        # Tabelle Feld 1: Text
        $MIB{'3.1.2.0'} = "81";        # Tabelle Feld 2: Integer

    @MIB = sort by_hierarchy keys(%MIB);    # sortierte Liste der MIB-Variablen
    }

&Subagent;
```

E.2 Listing von subagent_standard.pl

```
# snmp - Subagent in Perl implementiert

sub Subagent {

########################################################
# Beginn des Hauptprogramms
########################################################

&initial_MIB;    # Intialisiere die MIB fuer den Subagenten

# Ermitteln des TCP-Port's, der fuer die DPI-Kommunikation
# mit dem Agenten benutzt werden soll.
$Port = &qDPIport;
print "Port:$Port\n";

# Ueber den ermittelten Port eine TCP-Verbindung zum Agenten
# aufbauen (fuer die DPI-Kommunikation) => Handle: DPI_Anschluss
&TCPConnect($Port);

# Oeffnen der DPI-Verbindung
&DPI_Open;
sleep 1;

# Registrieren des Teilbaums
&DPI_Register;

#---------------------------------------------------------------
# Warten auf eine Anfrage
while(TRUE)    # Warten auf DPI-Anfragen des Agenten
    {
    recv(DPI_Anschluss,$Laenge,2,0);
    $Laenge = unpack("n",$Laenge);
    recv(DPI_Anschluss,$dpi_packet,$Laenge,0);
    @dpi_header = unpack("H2H2H2nH2",$dpi_packet);    # Header auslesen
```

```
    if ($dpi_header[4] eq "01")    # Falls get-Request
        {
        print "Get-Request empfangen\n";
        &answer_get;        # Beantworte get-Request
        }
    elsif ($dpi_header[4] eq "02")    # falls getnext-Request
        {
        print "Getnext-Request empfangen\n";
        &answer_getnext;    # Beantworte getnext-Request
        }
    elsif ($dpi_header[4] eq "03" || $dpi_header[4] eq "0a" || $dpi_header[4] eq "0b")
        {    # falls set-Request Gruppe
        &answer_setgrp($dpi_header[4]);
        }
    elsif ($dpi_header[4] eq "05")    # falls Response
        {
        print "Response empfangen\n";
        next;
        }
    else
        {
        print "Undefinierbare Nachricht empfangen\n";
        print "gemeldeter Packettyp: $dpi_header[5]\n";
        }
    }
#---------------------------------------------------------------
# Unregistrieren des Teilbaums
&DPI_UnRegister;

# Schliessen der DPI-Verbindung
&DPI_Close;

# Schliessen des Handles fuer die TCP-Verbindung zum Agenten
close(DPI_Anschluss);
}
########################################################
# Ende des Hauptprogramms
########################################################
```

E.3 Listing der Beispiel-Kaskadendatei mib_gets.pl

```
############################################################
# answer_get2
# Beantwortung eines get's oder getnext's
############################################################
sub answer_get2
        {
        # Parameter uebernehmen
        local($GID,$InstID,$dpi_get_packet_id,$Next)=@_;
        # Lokale Variablen
```

```
        local($GID_Laenge,$InstID_Laenge,@Wert);
        ##################################################################
        # Gesuchten Wert ermitteln
        if (!($GID eq $GROUPID)) # Falls der Subagent fuer diese Group-ID
                {                # nicht zustaendig
                &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
                }

        if ($Next)      # End of MIB bei getnext
                {
                print "EndofMIB\n";
                $Wert = "ENDofMIB";
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
                $Next = undef;
                }
        elsif ($InstID eq '1.0')
                {
                $Wert = &get_1;  # Funktion fuer Instanz 1 aufrufen
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
                }
        elsif ($InstID eq '2.0')
                {
                $Wert = &get_2;  # Funktion fuer Instanz 2 aufrufen
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
                }
        elsif ($InstID =~ /^3\.1\.1/)
                {
                $Wert = &get_Tabelle($InstID);  # Funkt. fuer Tab. aufrufen
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
                }
        elsif ($InstID =~ /^3\.1\.2/)
                {
                $Wert = &get_Tabelle($InstID);  # Funkt. fuer Tab. aufrufen
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,$Wert);
                }
        else            # Variable nicht vorhanden
                {
                &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
                }
        }

1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```

E.4 Listing der Beispiel-Kaskadendatei mib_sets.pl

```
############################################################
# answer_set2
# Beantwortung eines Set-, Commit-, Undo-Requests
############################################################
sub answer_set2
        {
        # Uebernahme der Parameter
```

```
local($GID,$InstID,$Art,$dpi_get_packet_id,$Set_Typ,$Set_Laenge,
                        @Set_Wert)=@_;

if (!$GID eq $GROUPID)  # Falls der Subagent fuer diese Group-ID
        {               # nicht zustaendig
        &ResponseError($dpi_get_packet_id,"05") # Allgemeiner Fehler
        }
# Set-Variablen
if ($InstID eq "1.0")
        {
        if ($Art eq "03")       # Set - Request
                {
                $Fehler = &set_1($Set_Laenge,@Set_Wert);
                }
        elsif ( $Art eq "0a")   # Commit - Request
                {
                $Fehler = &com_1;      # Commit-Funk.
                }
        elsif ( $Art eq "0b")   # Undo - Request
                {
                $Fehler = &undo_1;       # Undo-Funktion
                }
        if(!$Fehler)
                {
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,
            $Set_Typ,$Wert{$InstID});
                }
        else
                {# nicht Schreibbar
                &ResponseError($dpi_get_packet_id,$Fehler)
                }
        }
elsif  ($InstID eq '2.0')
        {
        if ($Art eq "03")       # Set - Request
                {       # Funktion fuer Instanz 2 aufrufen
                $Fehler = &set_2(@Set_Wert);
                }
        elsif ( $Art eq "0a")   # Commit - Request
                {
                $Fehler = &com_2;        # Commit-Funktion
                }
        elsif ( $Art eq "0b")   # Undo - Request
                {
                $Fehler = &undo_2;       # Undo-Funktion
                }
        if(!$Fehler)
                {
                &ResponseSuccess($dpi_get_packet_id,$GID,$InstID,
            $Set_Typ,$Wert{$InstID});
                }
        else
                {# nicht Schreibbar
```

```
                        &ResponseError($dpi_get_packet_id,$Fehler)
                        }
                }
        else            # Keine Set-Variable vorhanden
                {
                &ResponseError($dpi_get_packet_id,"11") # nicht Schreibbar
                }
        }

1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```

E.5 Listing der Beispiel-Datei sub_gets.pl

```
########################################################################
# get_1 Funktion zum Bestimmen des Werts der Variablen 1.0
# Rueckgabe : Wert der Variablen
# Annahme: der Wert befindet sich in der Variablen $A1
########################################################################
sub get_1
        {
    return($A1);
    }

########################################################################
# get_2 Funktion zum Bestimmen des Werts der Variablen 2.0
# Rueckgabe : Wert der Variablen
# Annahme: der Wert befindet sich in der Variablen $A2
########################################################################
sub get_1
        {
    return($A2);
    }

#################################################################
# get_Tabelle Funktion fuer Uebergabe der Tabelle;
# Annahme dabei, die erste Spalte ist der Index des assoziativen
# Arrays %Links und die zweite Spalte ist das jeweilige Element
# des Arrays. Ausserdem befinden sich alle Indizes sortiert in der
# Liste @Links.
# Parameter: InstID
# Rueckgabe: Wert einer Variablen aus der Tabelle
#################################################################
sub get_Tabelle
        {
        # Uebernahme der Parameter
        local($InstID)=@_;
        # Lokale Variablen
        local($Wert,@Wert,$ZeilenNr,$VarNr);

        ###################################################
        # Welche Tabellenspalte(=VarNr) und welche Zeile
        $ZeilenNr= substr($InstID,rindex($InstID,'.')+1);
```

```
        $InstID = substr($InstID,0,rindex($InstID,'.'));
        $VarNr= substr($InstID,rindex($InstID,'.')+1);
        $InstID = substr($InstID,0,rindex($InstID,'.'));

        #################################################
        # Ausgeben des gewuenschten Tabelleneintrags
        if ($VarNr == 1)
                {       # Falls Index
                $Wert = $Links[$ZeilenNr];
                }
        else
                {       # Falls kein Indexfeld
                $Wert = $Links{$Links[$ZeilenNr]};
                }
        ##################################################
        # Falls es eine naechste Zeile in der Tabelle gibt,
        # die noch nicht in der internen MIB aufgenommen ist,
        # aufnehmen
        if ($Links[$ZeilenNr+1] && !($MIB{$InstID.'.'.$VarNr.'.'.
                        ($ZeilenNr+1)}))
                {       # Falls Folgezeile ex. nimm Sie in MIB auf
        # Kopiere Typ von nullter Zeile
                $MIB{$InstID.'.'.$VarNr.'.'.($ZeilenNr+1)}=
                    $MIB{$InstID.'.'.
                        $VarNr.'.'."0"};
                @MIB = sort by_hierarchy keys(%MIB);#sort. Liste der MIB-Var.
                }

        return($Wert);  # wert zurueckgeben
        }

1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```

E.6 Listing der Beispiel-Datei sub_sets.pl

```
################################################################
# set_1(InstID,Typ,Laenge,Wert) Set-Funktion fuer die erste Variable
# Parameter : InstID,    Instanz- ID der gesuchten Variable
#          Typ,    Typ des Wertes
#          Laenge,   gibt die Laenge des Werts an
#             Wert,     der Wert selbst, als hex-Array
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
# Annahme: Wert befindet sich in Variable $A1; besitze eine
# Variable $A1_Temp als Zwischenspeicher bis zum Commit.
################################################################
sub set_1
        {
        # Uebernahme der Parameter
        local($InstID,$SetTyp,$SetLaenge,@SetWert)=@_;
        # Lokale Variablen
        local($Temp);
```

```
        if (!($SetTyp eq $MIB{$InstID}))        # Typ ueberpruefen
                {
                return("07");
                }
    foreach (@SetWert)    # Typ war Text, deshalb:
        {
        $NewWert = unpack("A",pack("H2",$_));
        $A1_Temp = $A1_Temp.$NewWert;
        }

        return(undef);
        }

#################################################################
# set_2(Typ,Laenge,Wert) Set-Funktion fuer den WWW-Server
# Parameter : InstID,    Instanz- ID der gesuchten Variable
#          Typ,    Typ des Wertes
#          Laenge,   gibt die Laenge des Werts an
#             Wert,     der Wert selbst, als hex-Array
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
# Annahme: Wert befindet sich in Variable $A2; besitze eine
# Variable $A2_Temp als Zwischenspeicher bis zum Commit.
#################################################################
sub set_2
        {
        # Uebernahme der Parameter
        local(($InstID,$SetTyp,$SetLaenge,@SetWert)=@_;
        # Lokale Variablen
        local($Temp);

        if (!($SetTyp eq $MIB{$InstID}))        # Typ ueberpruefen
                {
                return("07");
                }
    $A2_Temp = unpack("N",pack("H2H2H2H2",$SetWert[0],$SetWert[1],
                           $SetWert[2],$SetWert[3]));

        return(undef);
        }
1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```

E.7 Listing der Beispiel-Datei sub_coms.pl

```
#################################################################
# com_1 Funktion fuer die Variable an Instanz 1.0
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub com_1
        {
        $A1 = $A1_Temp;
        return(undef);
        }
```

```
#################################################################
# com_2 Funktion fuer die Variable an Instanz 2.0
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub com_2
        {
        $A2 = $A2_Temp;
        return(undef);
        }
1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```

E.8 Listing der Beispiel-Datei sub_undos.pl

```
#################################################################
# undo_1 Funktion fuer die Variable an Instanz 1.0
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub undo_1
        {
        $A1_Temp = $A1;
        return(undef);
        }

#################################################################
# undo_2 Funktion fuer die Variable an Instanz 2.0
# Rueckgabe : undef bei Erfolg, sonst Fehlercode
#################################################################
sub undo_2
        {
        $A2_Temp = $A2;
        return(undef);
        }
1;# PERL - spez. Eine Dat., die mit require eingeb. wird muss mit True enden.
```