INSTITUT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Implementierung eines Werkzeuges zur Generierung von HTML-Seiten aus bestehenden Dokumenten

Peter Allgeyer

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Alexander Keller
Abgabedatum:	1. Dezember 1997

Zusammenfassung

Die intensive und weiterhin zunehmende Nutzung des *World Wide Web* führt bei den Anbietern von WWW-Dokumenten zu einem hohen Bedarf an Unterstützungswerkzeugen für die Administration ihrer WWW-Server.
Insbesondere die Überführung von Dokumenten, die in unterschiedlichen Textformaten vorliegen, in das HTML-Format gestaltet sich problematisch. Sie muß heute zum überwiegenden Teil von Hand durchgeführt werden, da keine Werkzeuge auf dem Markt erhältlich sind, die die Anforderungen an die Qualität der erzeugten HTML-Seiten erfüllen.

Im Rahmen dieses Fortgeschrittenen-Praktikums wurde daher ein Werkzeug implementiert, das geeignete HTML-Seiten für die Dokumentenformate Postscript und Encapsulated Postscript (EPSF) erzeugt sowie für Dokumente, die in LaTeX- und BibTeX-Formaten vorliegen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

Zum Verständnis der Aufgabenstellung müssen zunächst einige technische Begriffe und Gegebenheiten geklärt werden.

Das Münchener Netzwerk Management Team (im folgenden kurz MNM-Team genannt) stellt seit geraumer Zeit einen Großteil der bei ihm erstellten Dokumente im *World Wide Web* dar. Hierzu gehören u.a.:

- Diplomarbeiten,
- Dissertationen,
- Fortgeschrittenenpraktika,
- Projektberichte sowie
- Publikationen.

Bislang geschieht dies jedoch nicht in der für das *World Wide Web* üblichen Darstellungsweise im Hypertext Markup Language (HTML[1])-Format. Vielmehr werden sämtliche Dokumente im Postscript und Encapsulated Postscript (EPSF)-Format dargestellt, so daß der Betrachter der Seiten auf einen *Previewer*[2] angewiesen ist.

1.1 Aufgabenstellung

Die Aufgabe besteht darin, ein Werkzeug zu implementieren, das die bestehenden Dokumente ins HTML-Format überführt, ein Inhaltsverzeichnis der vorhandenen Dokumente erstellt und zu jedem Dokument eine kleine Zusammenfassung extrahiert.

Insgesamt soll der Generierungsvorgang folgendermaßen ablaufen:

1. Die Quellendokumente werden einem vorher definierten Verzeichnis entnommen.

[1] HTML ist eine Anwendung der Standard Generalized Markup Language (SGML). Sie stellt eine einfache Sprache dar, die zum Aufbau von plattformunabhängigen Hypertext-Dokumenten dient.

[2] Programm, das bestimmte Dateiformate anzeigen kann.

2. Das zu implementierende Script extrahiert den Inhalt der Dokumente und generiert HTML-Seiten. Hierbei ist u.a. darauf zu achten, daß Literaturverweise innerhalb eines Dokuments als HTML-Links auf die nachstehend angegebene Bibliographie angelegt werden.

3. Anschließend wird das Quellenverzeichnis bereinigt.

1.2 Funktionsweise des Scripts

Da ein Großteil der vorhandenen Literatur aus Dokumenten im LaTeX-Format besteht, basiert das implementierte *Perl5* Script auf dem von Nikos Drakos `<nikos@cbl.leeds.ac.uk>` geschriebenen Programm LaTeX2HTML. Wie der Name schon sagt, konvertiert es Dokumente, die im LaTeX-Format geschrieben wurden, ins HTML-Format. Dabei achtet es darauf, Fußnoten, Bilder, Tabellen, Literaturverweise und Verweise als Links innerhalb des HTML-Dokuments darzustellen.

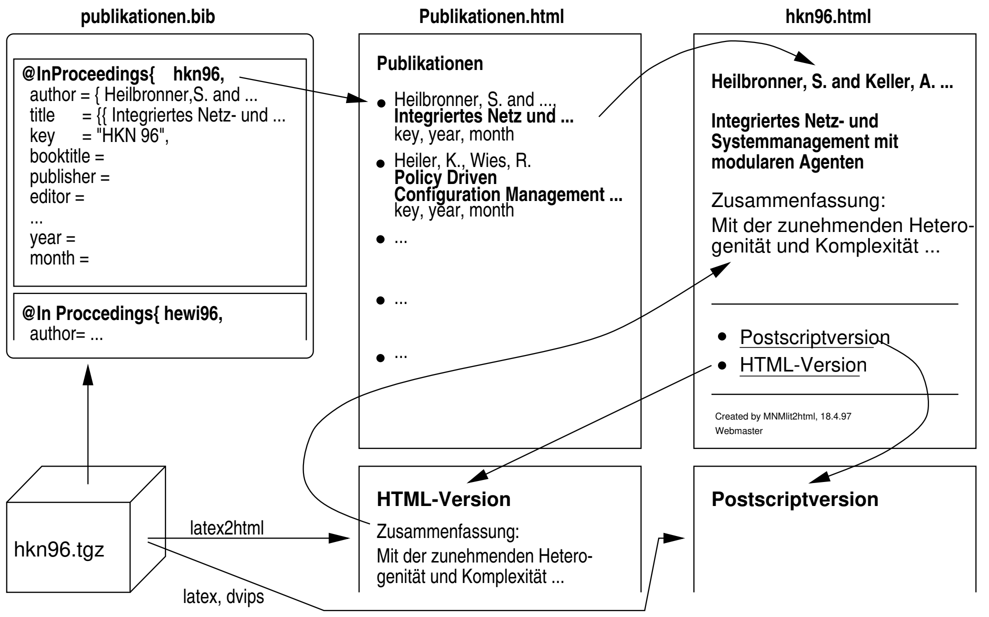

Abbildung 1.1: Ablaufschema

Üblicherweise ruft man das dieser Ausarbeitung zugrundeliegende Script mnmlit2html.pl nicht direkt auf, sondern mit Hilfe eines Wrappers. Der Wrapper, mnmlit2html genannt, ist ein kleines Bourne Shell Script, welches die entsprechenden Umgebungsvariablen setzt, und anschließend mnmlit2html.pl mit den übergebenen Parametern startet.

mnmlit2html.pl erwartet die zu bearbeitenden Dokumente gepackt und komprimiert abgelegt[3] in einem vorher festgelegtem Verzeichnis.

[3] Das eingehende Dokumentenpaket muß mit dem Archivierprogramm `tar` gepackt und anschließend mit `gzip` komprimiert werden.

Das *Perl5*-Script mnmlit2html.pl entpackt die eingehenden Dokumente in ein temporäres Verzeichnis. Anschließend ruft es LaTeX2HTML auf, um die LaTeX-Dokumente nach HTML zu konvertieren.

Die lehrstuhlinternen BibTeX-Dateien (`diplomarbeiten.bib`, `dissertationen.bib`, `fopras.bib`, `projekte.bib`, `publikationen.bib`) legen die thematische Einordnung der Dokumente nach ihrer Art fest. Einen typischen Eintrag dieser Dateien zeigt die linke Hälfte in Abb. 1.1 auf Seite 2.

Diesem Eintrag entnimmt mnmlit2html.pl Verfasser, Titel, Erscheinungsjahr sowie sonstige relevante Informationen. Welche Informationen man als relevant betrachtet, kann man in einer Konfigurationsdatei angeben (siehe Kapitel 3.3 auf Seite 12).

Alle zu einer Sparte (Diplomarbeiten, Dissertationen, Fopras etc.) gehörigen Arbeiten legt mnmlit2html.pl im nächsten Schritt auf einer eigenen Seite ab (siehe mittlere Hälfte der Abb. 1.1 auf Seite 2 oder Abb. 1.2 auf Seite 4), so daß das Script fünf HTML-Seiten generiert, die die aus den fünf BibTeX-Dateien gewonnenen Informationen chronologisch nach Verfasser geordnet auflisten. Voraussetzung für das Anlegen dieser Datei ist, daß mindestens ein Dokument dieser Sparte erfolgreich mit mnmlit2html.pl eingebunden wurde.

Darauffolgend erzeugt mnmlit2html.pl eine „Vorabtitelseite“ (siehe rechte Hälfte der Abb. 1.1 auf Seite 2 oder Abb. 1.3 auf Seite 5). Der Aufbau der Seite besteht aus dem Verfasser, dem Titel des verfaßten Dokuments und einer kleinen Zusammenfassung. Diese Zusammenfassung wird aus dem vorliegenden LaTeX-Quellcode des Dokuments exstrahiert, nachdem dieses mittels LaTeX2HTML nach HTML konvertiert wurde. Voraussetzung für die Erzeugung einer Zusammenfassung ist das Vorhandensein von \abstract im LaTeX-Source. Ist dieses Konstrukt nicht Teil des LaTeX-Source, so kann trotzdem eine Zusammenfassung extrahiert werden, sofern man vorgeht, wie in Kapitel 3.3.2 auf Seite 14 beschrieben ist.

Das Vorabtitelblatt weist drei Links auf. Der eine verweist auf das im ersten Schritt nach HTML übersetzte Dokument, die anderen beiden auf das Dokument im Postscript bzw. PDF-Format, sofern diese im eingehenden Dokumentenpaket enthalten sind. Sinn und Zweck dieser Seite ist es, eine kleine Zusammenfassung des angewählten Dokuments übersichtlich darzustellen und als Ausgangspunkt für die Postscript-, PDF- sowie HTML-Versionen zu dienen.

Abbildung 1.2: Inhaltsübersicht

Abbildung 1.3: Vorabtitelseite

Kapitel 2

Aufruf und Arbeitsweise

2.1 Voraussetzung

Das Programm mnmlit2html.pl setzt das Vorhandensein des *Perl5*-Scripts LaTeX2HTML von Nikos Drakos `<nikos@cbl.leeds.ac.uk>` in der Version 97.1 oder höher voraus.

Vor dem Starten des Programms ist sicherzustellen, daß im Quellverzeichnis[1] ein oder mehrere Dokumentenarchive liegen. Darüberhinaus müssen die BibTeX-Dateien lesbar und das Zielverzeichnis mit den notwendigen Schreibrechten versehen sein.

Fehlt eine dieser Prämissen, so endet das Programm mit einer Fehlermeldung.

2.2 Der Aufruf des Programms

Das Programm mnmlit2html.pl versteht sieben optionale Aufrufparameter:

-latex2htmldir <directory>
: Gibt den Pfad zu dem Verzeichnis an, in dem LaTeX2HTML zu finden ist.

-bibdir <directory>
: Bezeichnet den Pfad zu dem Verzeichnis, in dem die BibTeX-Dateien liegen.

-inputdir <directory>
: In diesem Verzeichnis liegen die Quelldokumente im gezippten Tararchiv-Format.

-htmldir <directory>
: In dieses Verzeichnis werden die fertig formatierten Dokumente nach ihrem Lauf durch mnmlit2html.pl gelegt.

-tmpdir <directory>
: Zeigt das temporäre Verzeichnis an. In dieses Verzeichnis wird das komprimierte Tar-Archiv ausgepackt und verschiedene Hilfsdateien von LaTeX2HTML und mnmlit2html.pl gelegt.

[1] Standardmäßig: /proj/Literatur/MNM/Upload

-childline <reg-expr>
Gibt die Trennung zwischen Hauptteil und Fuß eines erstellten HTML-Dokuments an. Mit Hilfe des übergebenen regulären Ausdrucks erkennt mnmlit2html.pl zuverlässig den Schluß einer Zusammenfassung, wenn es keinen *Abstract* im vorliegenden LaTeX-Dokument gibt.

Dieser Parameter muß der Variable `$CHILDLINE` aus LaTeX2HTML angepaßt werden. Siehe hierzu das Beispiel in Kapitel 3.3.2 auf Seite 13.

-debug
Schaltet in den Debug-Modus. Es werden ausführliche Meldungen über den Programmverlauf auf `<stdout>` ausgegeben (default: kein Debugging).

-h(elp)
Gibt folgende kleine Übersicht über die Programmparameter aus:

```
Usage: mnmlit2html.pl
     [-latex2htmldir <directory>]
     [-bibdir <directory>]
     [-inputdir <directory>]
     [-htmldir <directory>]
     [-childline <reg-expr>]
     [-background]
     [-debug]
     [-h(elp)]
```

2.3 Die Arbeitsweise von mnmlit2html.pl

Das Programm mnmlit2html.pl ist dafür verantwortlich, die im Archivformat vorliegenden Dokumente zu scannen, deren Übersetzung nach HTML anzustoßen und das Ergebnis in Form einer geordneten Verzeichnisstruktur (siehe Abb. 3.1 auf Seite 11) geeignet abzulegen.

Nach dem Auslesen einiger Pfadkonstanten wird zunächst die Konfigurationsdatei eingelesen. Daraufhin entpackt das Script den Sourcecode eines oder mehrerer Dokumente, die in einem definierten Verzeichnis abgelegt wurden. Das Programm sucht nach Dateien mit den Endungen *.tgz, .tar.gz* und *.tar.Z* und entpackt diese in einem temporären Verzeichnis.

Alsdann durchsucht es das angegebene Verzeichnis, in dem sich die BibTeX-Dateien befinden, scannt die gefundenen Dateien nach den in der Initialisierungsdatei `.mnmlit2html.ini` spezifizierten Einträgen und speichert diese in einem Hash. Der Hash ordnet den Schlüsseln der BibTeX-Datei (allg97, kene97, ...) die einzelnen Felder (author, title, key, ...) zu. Dazu war es nötig, den assoziativen Array, wie der Hash unter *Perl* auch genannt wird, zu verschachteln. Die dem Schlüssel zugeordnete Information bestand ihrerseits wieder aus einem assoziativen Array. Da die Informationen, die der Hash speichert, öfters benötigt werden, bot sich diese Methode besonders an, da auf einen Hash schneller zugegriffen werden kann als auf ein einfaches Array oder mehrere Stringvariablen.

Im folgenden Schritt werden pro entpackter Datei die dazugehörigen Vorabtitelseiten erstellt. Hierzu werden die Dateien mittels des Hilfsprogramms LaTeX2HTML nach HTML konvertiert und anschließend nach Schlüsselwörtern durchsucht, anhand derer eine Zusammenfassung des vorliegenden Dokuments erzeugt wird. Schlüsselwörter sind etwa Zusammenfassung, Einführung, Abstract o.ä.

Die Schlüsselwörter können in der Konfigurationsdatei angegeben werden. Zusammen mit den im Hash erfaßten Daten über Autor, Titel, Erscheinungsjahr und -monat erscheint die Zusammenfassung auf der Vorabtitelseite. Diese ist in HTML abgefaßt und kann in Abbildung 1.3 auf Seite 5 beispielhaft betrachtet werden.

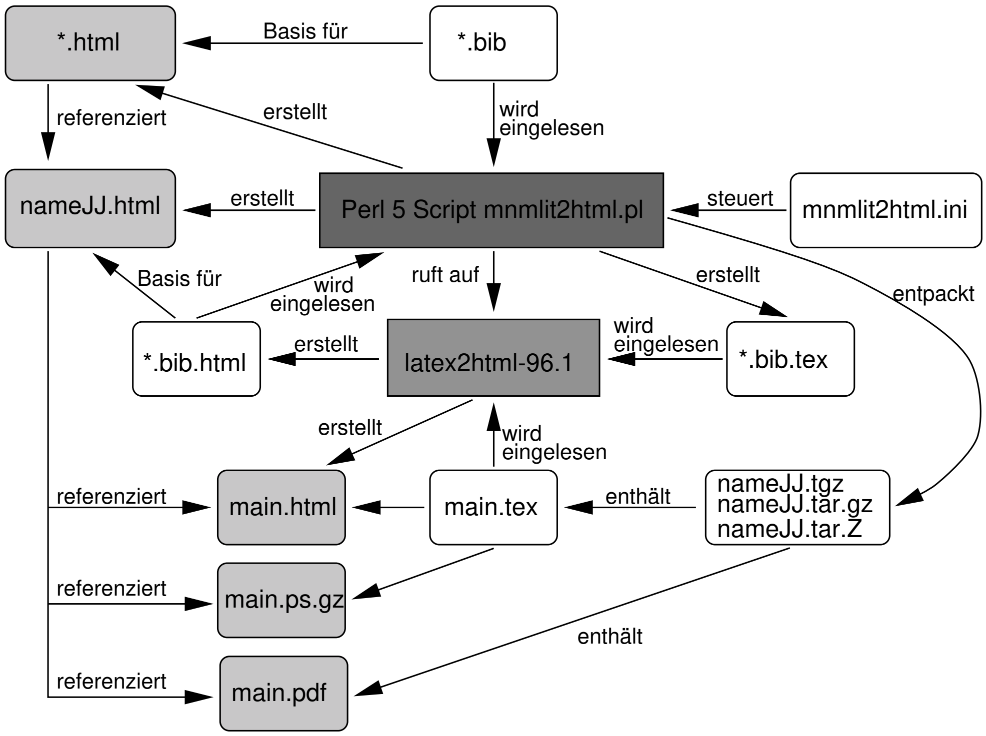

Abbildung 2.1: Beziehungsgeflecht

Im darauffolgenden Schritt werden die generierten HTML-Seiten sowie die zum Dokumentationsbaum gehörige Postscript- sowie PDF-Datei an ihren entgültigen Bestimmungsort kopiert. Falls die Postscript-Datei unkomprimiert vorliegt, wird sie mit GNU-Zip komprimiert, um Plattenplatz und Bandbreite bei der Übertragung zu sparen. Zusätzlich gibt eine in Klammern angefügte Zahl die Größe der komprimierten Datei in Kilobyte an, damit jeder für sich entscheiden kann, ob es sich lohnt die Datei herunterzuladen. PDF-Dateien sind von Haus aus schon recht kompakt, so daß auf eine derartige Behandlung verzichtet wurde.

Um auf die einzelnen Arbeiten gezielt zugreifen zu können, wird am Schluß noch eine thematische Inhaltsübersicht erstellt, die in Abbildung 1.2 auf Seite 4 betrachtet werden kann. Im vorliegenden Fall stimmen diese Inhaltsübersichten mit den Namen der einzelnen

BibTeX-Dateien überein. Die Inhaltsübersicht enthält eine Auflistung sämtlicher gefundener Dokumententitel, deren Autor, Jahr und Monat sowie sämtlicher Einträge, die in der Konfigurationsdatei als „zu Veröffentlichen“ gekennzeichnet sind. Ausgehend vom Titel des jeweiligen Dokuments kann per *Hyperlink* die Vorabtitelseite des Dokuments erreicht werden, sofern der Quellbaum des Dokuments schon übersetzt wurde.

Nach einem Hinweis über den Erfolg der übersetzten Seiten ist das Programm beendet. Die Beziehungen der einzelnen Skripten zueinander und zur Umgebung stellt die Abbildung 2.1 auf Seite 8 dar.

Kapitel 3

Implementierung

Zur Implementierung wurde die Skriptsprache *Perl5* verwendet, da diese ein geeignetes Werkzeug im Umgang mit der Textanalyse ist. Dabei fanden Routinen aus dem Buch *Programming Perl* [WCS92, WCS96] Anwendung.

3.1 Allgemeiner Aufbau des Programms

Das Programm gliedert sich im wesentlichen in drei Teile:

- Dem Hauptprogramm mnmlit2html.pl, welches die BibTeX-Dateien scannt und für die Erstellung der HTML-Seiten verantwortlich ist,
- dem Hilfsprogramm LaTeX2HTML, das die Umsetzung des LaTeX-Quellcodes nach HTML bewerkstelligt und
- der Konfigurationsdatei `.mnmlit2html.ini`, in der die Pfade zu den einzelnen Programmen sowie die Regeln zur Erstellung des Vorabtitelblattes festgehalten werden.

3.2 Die Bereitstellung der Daten

Die Archivdateien müssen folgendem Layout genügen:

Im Verzeichnis `Dokumentation` befinden sich die Unterverzeichnisse `Postscript`, `PDF` und `Latex`.

Das Verzeichnis `Postscript` enthält das Dokument im Postscript-Format, das Verzeichnis `PDF` die PDF-Variante. Sollte das Dokument nicht der Form *nameJJ.ps* entsprechen oder *main.ps* heißen, so wird die erste gefundene Datei mit der Endung *.ps* als die maßgebende Postscript-Datei angesehen, auf die später auf dem Vorabtitelblatt mittels Hyperlink bezug genommen wird. Analoges gilt für die PDF-Datei.

nameJJ bezeichnet den Schlüssel aus der BibTeX-Datei, der sich aus den ersten vier Buchstaben des Nachnamens zuzüglich der Jahreszahl der erstellten Arbeit ergibt. So wäre der Eintrag dieses Dokuments bspw. **allg97** für den Autor Allgeyer und die Jahreszahl 97, dem Jahr, in dem das Dokument erstellt wurde. Das MNM-Team trägt alle erstellten

Dokumente in dieser Weise in einer der fünf BibTeX-Dateien ein. Die Schlüssel müssen *eindeutig* vergeben werden! mnmlit2html.pl wurde so geschrieben, daß es auch Einträge mit mehr als vier Buchstaben findet. So ist es in der Lage auch Einträge der Form *kene97a*, *kene97b* und *kene97c* zu erkennen, so daß weiterhin die Eindeutigkeit gewahrt bleibt und gleichzeitig das vorliegende Schriftstück chronologisch korrekt eingeordnet werden kann.

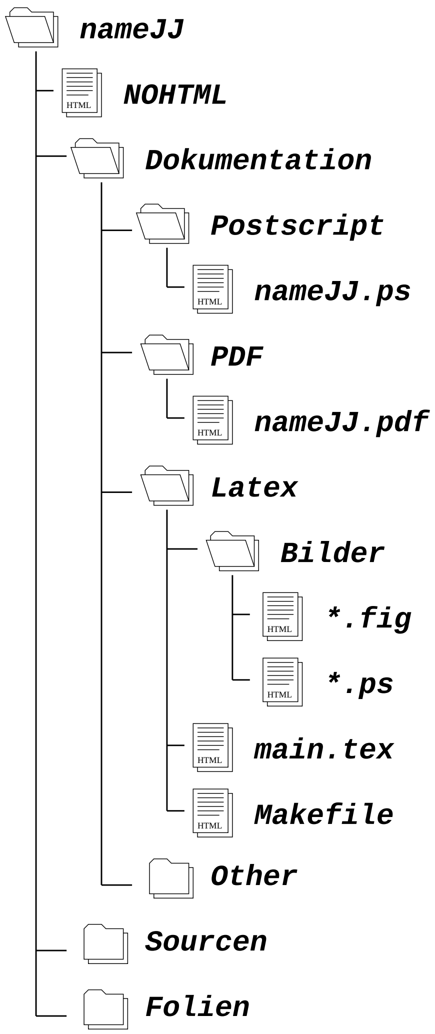

Abbildung 3.1: Verzeichnisstruktur

Im Verzeichnis `Latex` befinden sich die LaTeX-Quelldateien, die an der Kennung *.tex* zu erkennen sind. Besteht der Text aus mehreren LaTeX-Quelldateien, so sucht mnmlit2html.pl zunächst nach einer Datei mit dem Namen `main.tex` bzw. nach einer Datei der Form *nameJJ.tex*. Erst wenn es keine solche LaTeX-Quelldatei findet, beginnt mnmlit2html.pl die vorliegenden *.tex*-Dateien nach `\begin{document}` zu parsen. Durch diese Vorgehensweise findet mnmlit2html.pl recht zuverlässig das Hauptdokument zu einer Ausarbeitung, auch wenn im Quellverzeichnis mehrere unterschiedliche LaTeX-Dokumente abgelegt werden.

Bilder, die im Dokument eingebunden werden, gehören einschließlich der Bildquelldatei in ein Unterverzeichnis `Bilder`[1]. Bei komplizierteren LaTeX-Bearbeitungen ist noch Platz für ein `Makefile` vorgesehen, welches den Durchlauf von `latex`, `bibtex`, `makeindex` etc. automatisiert, aber *nicht* automatisch von mnmlit2html.pl angestoßen wird!

Eine besondere Funktion hat die Datei `NOHTML`. Existiert sie, so legt mnmlit2html.pl keinen HTML-Teil an. `NOHTML` ist also ein Schalter, der es ermöglicht, die Umwandlung des vorliegenden Dokuments von LaTeX nach HTML zu untersagen.

3.3 Die Konfigurationsdatei `.mnmlit2html.ini`

Das Hauptprogramm mnmlit2html.pl benötigt eine Konfigurationsdatei, in der die wichtigsten Programmpfade und Angaben über die Art und Weise der Darstellung der HTML-Seiten festgehalten werden.

3.3.1 Das Auffinden der Konfigurationsdatei

Die Konfigurationsdatei bekommt wie schon erwähnt den Namen `.mnmlit2html.ini`. Diese Datei wird in einem der folgenden drei Verzeichnisse gesucht, wobei die Reihenfolge bestimmt, welche Parameter letztendlich gültig sind.

Suchreihenfolge:

1. Verzeichnis, in dem mnmlit2html.pl liegt[2],
2. aktuelles Verzeichnis,
3. Homeverzeichnis.

Wird keine Konfigurationsdatei gefunden, so gibt das Programm einen Hinweis aus und verwendet die im Hauptprogramm festgelegten Standardeinstellungen.

3.3.2 Der Aufbau der Konfigurationsdatei

Die Konfigurationsdatei gliedert sich in drei Teile. Im ersten Teil werden benötigte Pfade und Programmschalter gesetzt. Der zweite Teil bestimmt die Begriffe, nach denen das Programm das vorliegende Dokument durchsuchen soll. Im dritten Teil schließlich werden Konvertierungsregeln angegeben.

Die Pfadangaben und Programmschalter

Sie werden in *Perl5*-Syntax durch ein vorangestelltes Dollar-Zeichen eingeleitet, dem der Bezeichner der Variable folgt. Jeder Zeile folgt ein abschließender Strichpunkt.

[1] Zur Kennzeichnung dieses Verzeichnisses sind auch andere Namen wie bspw. Pictures, pictures, bilder etc. denkbar.

[2] Wird das Hauptprogramm über Symlinks referenziert, muß die Variable $MNMLIT2HTMLDIR per Hand im Hauptprogramm gesetzt werden!

```
$INI_DIR = dirname( &getcwd . "/" . "$0" );
```

Diese Variable gibt den Pfad zum Verzeichnis mit der Initialisierungsdatei an. Wird das Hauptprogramm über Symlinks referenziert, so muß diese Variable explizit gesetzt werden, damit das Hauptprogramm die Initialisierungsdatei auch in dem Verzeichnis finden kann, in dem das ausführbare Programm liegt. Denkbar wäre es auch, ein Verzeichnis `lib` oder `etc` zu erstellen und `$INI_DIR` darauf verweisen zu lassen. Da es wenig Sinn macht, diese Variable erst in der Initialisierungsdatei zu definieren, gehört sie natürlich ins Hauptprogramm mnmlit2html.pl und soll hier nur der Vollständigkeit halber aufgeführt werden.

```
$LATEX2HTML_DIR = "/usr/local/dist/bin";
```

bezeichnet das Verzeichnis, in dem sich das Tool LaTeX2HTML befindet. Voraussetzung ist mindestens die Version 97.1, da sich der Aufruf der Parameter früherer Versionen teilweise anders verhält.

```
$HTML_DATA_DIR = "/proj/Literatur/public-htdocs/MNMPub";
```

gibt den Ort an, an dem die fertigen HTML-Dokumente abgelegt werden. Dieses Verzeichnis sollte natürlich auch dem WWW-Server bekannt sein, damit auf diese Seiten später zugegriffen werden kann.

```
$INPUT_DIR = "/proj/Literatur/MNM/Upload";
```

bezeichnet das Quellverzeichnis. Hier werden die zu bearbeitenden LaTeX-Dokumente im gezippten Tar-Format abgelegt. Den Aufbau des Tar-Files beschreibt das Schriftstück [HK96].

```
$BIB_DIR = "/proj/Literatur/MNM/Bib";
```

Das Bibliographie-Verzeichnis, es enthält Informationen über die einzelnen Dokumente und deren Verfasser. Im einzelnen sind dies die fünf lehrstuhlinternen BibTeX-Dateien (`diplomarbeiten.bib`, `dissertationen.bib`, `fopras.bib`, `projekte.bib`, `publikationen.bib`).

```
$TMP = "/tmp";
```

Temporäres Verzeichnis, dieses sollte ausreichend Platz zur Verfügung stellen, um das entpackte Dokument aufnehmen zu können. Desweiteren legt hier das Hauptprogramm einige Hilfsdateien an. Die angelegten Dateien werden nach Abarbeiten von mnmlit2html.pl wieder gelöscht.

```
$CHILDLINE = "<BR>.*<HR>";
```

Dieser Eintrag bezeichnet den HTML-Befehl, der Haupteil und Fuß voneinander trennt. Er wird normalerweise in der Datei `latex2html.config` gesetzt und sollte entsprechend übernommen werden, damit trotz fehlendem Abstract eine Zusammenfassung korrekt gefunden wird. Anhand dieser Variable kann mnmlit2html.pl exakt den Schluß der Zusammenfassung erkennen. Der Default-Eintrag ist sowohl für die Version 96.1 als auch die Version 97.1 von LaTeX2HTML gültig.

Die Suchbegriffe

Der zweite Teil der Konfigurationsdatei ist ein wenig komplexer. Hier können Begriffe angegeben werden, nach denen das Programm das vorliegende Dokument durchsuchen soll. Die Syntax gleicht der von *Perl5*-Arrays: Einen Klammeraffen gefolgt vom Arraynamen folgt eine in runden Klammern gefaßte Liste, deren Elemente durch Kommata getrennt werden. Es ist dringend anzuraten, die Listenelemente in doppelte Hochkommata zu fassen, damit auch gequotete Zeichen richtig von Perl übernommen werden! Auch der abschließende Strichpunkt einer jeden Zeile darf nicht vergessen werden.

```
@ZUSAMMENFASSUNG = ("Einf\"uhrung", "Zusammenfassung");
```

Standardmäßig sucht das Programm nach einem Abschnitt, der mit `\begin{abstract}` beginnt und mit `\end{abstract}` endet. Leider enthält nicht jedes Dokument diese Konvention. Damit trotzdem auf dem „Vorabtitelblatt" ein kleiner *abstract*, also eine kleine Zusammenfassung des vorliegenden Dokuments erscheint, muß dem Hauptprogramm bekanntgemacht werden, nach welchen Schlüsselwörtern es den vorliegenden Text durchsuchen soll. Zweckmäßig sind alle Suchbegriffe, die in einer Überschrift (section) des vorliegenden LaTeX-Dokuments erscheinen. Wird trotz allem keine Zusammenfassung extrahiert, so erscheint ein Hinweis beim Erzeugen auf der Standardausgabe, daß keine Zusammenfassung gefunden wurde.

Nachfolgend erscheint eine Liste aller der bibliographischen Gattungen, die auch in den schon angesprochenen lehrstuhlinternen BibTeX-Dateien (`diplomarbeiten.bib, publikationen.bib, fopras.bib, projekte.bib, dissertationen.bib`) erscheinen. Es wird nicht zwischen Groß- und Kleinschreibung unterschieden!

```
@article            = ("author","title","journal",
                       "year","type","month","note","key");
@article_to_publish = ("author","title","journal",
                       "month","year");
```

In der ersten Zeile steht für den Variablennamen die bibliographische Gattung. Die nachfolgende Liste gibt an, nach welchen Einzelfeldern[3] der Text geparst werden soll. Die der Konfigurationsdatei mitgelieferte Liste erhält keinen Anspruch auf Vollständigkeit! Sie ist nach eigenem Ermessen zu erweitern oder zu kürzen. Die Felder `author`, `title` und `year` sollten im Hinblick auf die Gestaltung des Vorabtitelblatts jedoch auf jeden Fall vorhanden sein.

Die zweite Zeile enthält diejenigen Einzelfelder, die später im Inhaltsverzeichnis erscheinen sollen. Daher auch das Namensanhängsel *_to_publish*.

Hinweis 1 *Wird nach dem Durchlauf von* mnmlit2html.pl *ein Eintrag im Inhaltsverzeichnis vermißt, so deutet das in den meisten Fällen daraufhin, daß die zum BibTeX-Eintrag gehörige bibliographische Gattung nicht in der Konfigurationsdatei aufgeführt wird.*

[3] Die Begriffe *bibliographische Gattung* sowie *Einzelfeld* stammen aus [Won93].

Die Konvertierungsregeln

Im dritten Teil der Initialisierungsdatei finden sich einige wenige Konvertierungsregeln, die das Umwandeln von Mehrzahl in Einzahl, Klein- in Großgeschriebenes oder das Expandieren von Abkürzungen beschreiben sollen.

```
%months = (
        jan     => Januar,
        feb     => Februar,
        mar     => "M\&auml\;rz",
        apr     => April,
        may     => Mai,
        jun     => Juni,
        jul     => Juli,
        aug     => August,
        sep     => September,
        oct     => Oktober,
        nov     => November,
        dec     => Dezember,
);
```

Der Aufbau gliedert sich in drei Teile:

- Dem Variablennamen, durch ein Prozentzeichen eingeleitet,
- dem Schlüsselwort, gefolgt von einem *Doppelpfeil* und
- dessen Wert, dem Ergebnis der Konvertierung.

Die Monatsangaben in den Einzelfeldern der BibTeX-Dateien sollen nach Konvention[4] nicht ausgeschrieben werden, sondern nur aus den ersten drei Buchstaben (der englischen Schreibweise und klein) bestehen – z.B. *may* oder *oct*. Damit diese Angaben korrekt ausgeschrieben auf der Inhaltsübersicht erscheinen, muß eine Umsetzungstabelle geschrieben werden. Diese kann man ändern, so daß auch eine englischsprachige Version in Betracht käme.

Hinweis 2 *Hier sollte nur die rechte Seite des assoziativen Arrays geändert werden, da sich die Abkürzungen auf der linken Seite i.A. nicht ändern. Ein Beispiel für englische Monatsnamen findet sich in* mnmlit2html.pl.

```
%types = (
   "diplomarbeiten"  => "Diplomarbeiten / Master's Theses",
   "dissertationen"  => "Dissertationen / Ph.D. Theses",
   "fopras"          => "Systementwicklungsprojekte / Advanced Practicals",
   "projekte"        => "Projektberichte / Project Reports",
   "publikationen"   => "Publikationen / Publications"
);
```

[4] Siehe bspw. auch die Hinweise im Kopf der Datei `publikationen.bib`.

Dieses assoziative Array beschreibt eine Namensumsetzung. Die auf der rechten Seite stehenden Begriffe werden den Begriffen auf der linken Seite zugeordnet.

Die Begriffe auf der linken Seite entsprechen den Namen der BibTeX-Dateien ohne Endung. Klein- und Großschreibung bleiben hierbei unberücksichtigt. Die Begriffe auf der rechten Seite entsprechen den Überschriften, die die zu erzeugenden HTML-Seiten bekommen werden.

Eine solche Umsetzung ist nötig, damit man die Überschriften auch in Zukunft frei gestalten kann.

3.4 Die erzeugte Verzeichnisstruktur

Unterhalb von `$HTML_DATA_DIR` legt das Programm die Verzeichnisstruktur an, auf die der Webserver zugreift (siehe Seite 16, Abbildung 3.2).

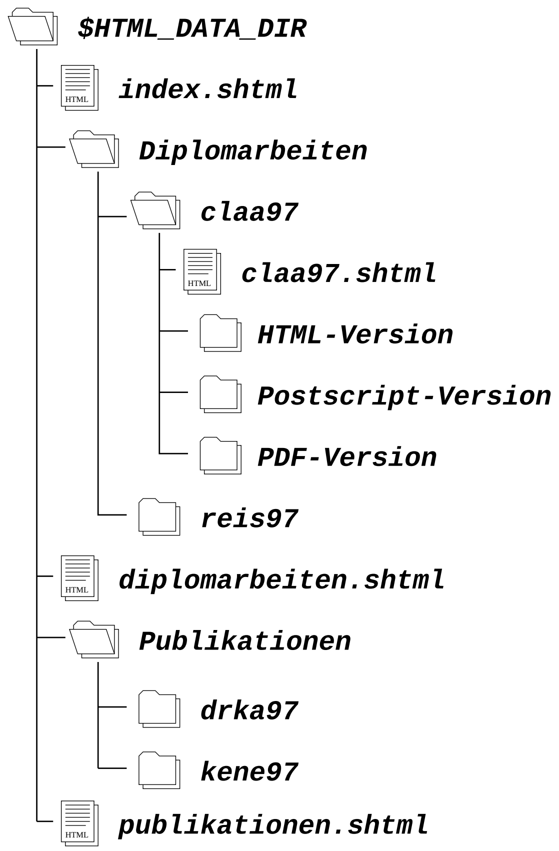

Abbildung 3.2: Verzeichnisstruktur unterhalb `$HTML_DATA_DIR`

Für jede Arbeit, die einer gefundenen BibTeX-Datei angehört, wird ein Unterverzeichnis mit dem Namen der BibTeX-Datei angelegt (Diplomarbeiten, Publikationen, etc.). Unterhalb dieser Verzeichnisse werden pro eingehendem Dokument weitere Verzeichnisse mit dem Namen des Dokuments angelegt (claa97, reis97, drka97 usw.). Parallel dazu gibt es für jede

BibTeX-Datei eine Datei mit der chronologischen Inhaltsübersicht.

Unterhalb der Dokumentenverzeichnisse schließlich sind die drei Verzeichnisse `HTML-Version` `PDF-Version` und `Postscript-Version` lokalisiert, die die jeweiligen Dokumente im betreffenden Format enthalten.

3.5 Performance

Anders als eine richtige Interpretersprache wie z.B. die *Bourne Shell*, die ein Skript Zeile für Zeile bearbeitet, übersetzt und ausführt, kompiliert die Programmiersprache *Perl* das Programm zunächst in eine Zwischenform, dem sogenannten Bytecode, den es dann vom Interpreter auszuführen gilt. Bereits beim Kompilieren werden Fehler gefunden und Optimierungen durchgeführt. Daher ist *Perl* signifikant schneller als ein ähnliches auf der Shell basierendes Programm. Es ist jedoch langsamer als ein in C oder C++ geschriebenes Programm, da der erzeugte Bytecode interpretiert werden muß. Der Vorteil von *Perl* gegenüber Programmiersprachen wie C oder C++ liegt in der Fähigkeit, ausgezeichnet mit Strings umgehen zu können.

Es läßt sich schlecht absolut sagen, wie lange das Script zur Bearbeitung eines Dokuments braucht. Viel zu viele Faktoren beeinflussen die Abarbeitungszeit, wie z.B. die Hardware des Rechners und die vorhandenen Systemressourcen sowie die Länge des Dokuments und deren Anzahl an Bildern. Für die Abarbeitung dieses Schriftstücks brauchte ein Intel 486 DX 100 mit 32 MB Hauptspeicher 2:32 Minuten. Eine SparcStation 20 mit zwei SPARC Prozessoren und 128 MB Hauptspeicher benötigte dagegen nur 2:00 Minuten. Die dabei erzeugte Last stieg weit über den Wert 1.00[5] an. Weitere Messungen an mehreren Dokumenten ergaben teilweise drastisch längere Laufzeiten, die v.a. von der Anzahl und Größe eingebetteter Bilder abhängig waren.

Wegen der relativ hohen Systembelastung über einen längere Zeitraum hinweg, kommt eine Online-Abarbeitung nicht in Frage. Überlicherweise soll die Konvertierung als Cron-Job über Nacht durchgeführt werden, so daß die Ergebnisse am nächsten Tag vorliegen. Daher ist die Frage der Geschwindigkeit und Systembelastung zu vernachlässigen.

[5] Von *top* erzeugte Ausgabe.

Kapitel 4

Das Hilfsprogramm makebib.pl

Bestehende Literatur verweist oftmals auf Dokumente, deren Bibliographie-Einträge zentral gewartet und abgelegt werden. Mittels der Environment-Variable `$BIBINPUTS` kann auf das Verzeichnis hingewiesen werden, welches die BibTeX-Dateien enthält. Nachteil dieser Methode ist es, daß Programme wie LaTeX2HTML die Literaturverweise u.U. nicht mehr richtig anzeigen können, wenn ein anderer Benutzer als der ursprüngliche Autor BibTeX startet.

Um dieses Problem zu vermeiden, wurde zusätzlich ein kleines *Perl*-Script geschrieben.

4.1 Funktionsweise

Das Script durchsucht alle vorhandenen Dateien mit der Endung *.tex* nach evtl. vorhandenen Literaturverweisen[1]. Im weiteren Verlauf durchsucht es alle BibTeX-Dateien, die es anhand der gesetzten Environment-Variablen `$BIBINPUTS` findet, nach den zuvor gefundenen Literaturverweisen und speichert die entsprechenden Einträge in einer neuen BibTeX-Datei. Der Name der neuen Datei kann frei gewählt werden.

4.2 Programmaufruf

Das Programm makebib.pl versteht drei Aufrufparameter:

-newbibfile <file>
: Mit diesem Parameter gibt man den Namen der zukünftigen BibTeX-Datei an. Dieser Name sollte in der LaTeX-Hauptdatei als `\bibliography{<file>}` eingetragen werden.

-debug
: Schaltet in den Debug-Modus. Es werden ausführliche Meldungen über den Programmverlauf auf `<stdout>` ausgegeben (default: kein Debugging).

-h(elp)
: Gibt folgende kleine Übersicht über die Programmparameter aus:

[1] Literaturverweise werden unter LaTeX mittels `\cite{<Verweis>}` eingeleitet.

```
This is Makebib Version 97.1a (Sep 29, 1997) by Peter Allgeyer,
Institut fuer Informatik, Technical University of Munich.

Usage: makebib.pl
   [-newbibfile <file>]
   [-debug]
   [-h(elp)]
```

Standardmäßig versucht makebib.pl eine Datei mit dem Namen `main.bib` zu erstellen. Gibt es jedoch schon eine Datei mit diesem Namen, so fragt das Script sicherheitshalber nochmal nach, ob diese überschrieben werden soll. das Programm bricht an dieser Stelle ab, wenn man die Frage mit „n“ oder „N“ beantwortet. Bei Bestätigung mit „y“ oder „Y“ folgt eine Aufstellung sämtlicher gefundener BibTeX-Einträge, die der neuen BibTeX-Datei zugefügt werden.

Zusätzlich zu dieser Beschreibung gibt es noch eine Manual-Seite namens `makebib.1`.

Kapitel 5

Hinweise zum Umgang mit LaTeX2HTML

LaTeX2HTML kann mittels Programmschalter, Initialisierungsdatei oder einer zentralen Konfigurationsdatei gesteuert werden. Die zentrale Konfigurationsdatei findet sich meist im gleichen Verzeichnis wie das *Perl*-Script `latex2html` und heißt `latex2html.config`. Sind die hier vorgenommenen Defaulteinstellungen für das vorliegende Dokument unzupassend, so muß nachgebessert werden.

Im Homeverzeichnis desjenigen Benuzers, der `mnmlit2html.pl` aufruft, sollte der betreffende Benutzer eine Datei mit Namen `.latex2html-init` anlegen, die von LaTeX2HTML ausgewertet wird. Leider wertet LaTeX2HTML seine Konfigurationsdateien in genau umgekehrter Reihenfolge aus wie mnmlit2html.pl. Dadurch ist es nicht möglich, die Konfigurationsdatei an einen neutralen Ort zu stellen, da hierdurch `.latex2html-init` im Homeverzeichnis des Benutzers der LaTeX2HTML aufruft, überschrieben würde.

Auf einige wenige in dieser Datei setzbare Parameter soll im Folgenden eingegangen werden, da sie das Aussehen der von mnmlit2html.pl erzeugten Dokumente entscheidend beeinflussen. Die angegebenen Werte sind als Defaultwerte zu verstehen, die für die meisten Schriftstücke die geeignetsten Ergebnisse liefern.

`$SHOW_SECTION_NUMBERS = 1;`
: Mit dieser Option kann die Numerierung der einzelnen Kapitel an- oder ausgeschaltet werden.

`$NETSCAPE_HTML = 1;`
: Der von LaTeX2HTML erzeugte Code ist für den Browser der Firma Netscape optimiert worden.

`$FIGURE_SCALE_FACTOR = "1.4";`
: Hiermit bestimmt man den Skalierfaktor der Grafikumwandlung. Größere Werte ergeben größere Bilder, kleinere Werte kleinere Bilder. Der in `latex2html.config` vorgeschlagene Wert von „1.6“ ist für viele in den Publikationen des MNM-Teams verwendete Grafiken zu groß. Der obige Wert eignet sich besser.

`$HTML_VERSION = "3.2";`

Unterstützt werden HTML-Versionen von 2.0 bis 3.2. Je nach verwendeter HTML-Version können entsprechende Features wie Tabellen erkannt und übersetzt werden. Ab Version 97.1 von LaTeX2HTML sollte der Wert unbedingt auf „3.2“ gesetzt werden. Damit erhöht sich die Qualität des erzeugten HTML-Codes.

`$ICONSERVER = "http://www.nm.informatik.....de/common/icons/latex2html-97.1";`

Das MNM-Team muß sich darauf verlassen können, daß auf seine Dokumente korrekt zugegriffen werden kann. Da das Programm LaTeX2HTML u.U. von einer anderen Institution zur Verfügung gestellt wird, wird es so sein, daß deren HTTP-Server als Ursprung sämtlicher Icons eingetragen ist. Damit nun nicht die Dokumete, welche die Icons referenzieren und die Icons selber an getrennten Orten zu finden sind, kann obige Variable gesetzt werden, die sinnigerweise auf den HTTP-Server des MNM-Teams zeigt.

`$ADDRESS = "Copyright Munich Network Management Team";`

LaTeX2HTML generiert normalerweise im Fuß einer jeden erstellten Seite das Datum und den Namen des Benutzers, der das Progarmm `latex2html` gestartet hat. Mit dieser Variable kann der Eintrag des Namens geändert werden.

`1; # This must be the last line`

Der Kommentar sagt es eigentlich schon. LaTeX2HTML verlangt eine eins gefolgt von einem Strichpunkt als letzte Zeile seiner Konfigurationsdatei.

Wer weitere Parameter verändern will, der sollte einen Blick in `latex2html.config` oder die Dokumentation zu LaTeX2HTML[1] werfen.

[1] http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/

Kapitel 6

Troubleshooting

In diesem Kapitel sollen die häufigsten Fehlerursachen und deren Behebung aufgelistet werden.

mnmlit2html.pl erzeugt keinen Listeneintrag
: In den meisten Fällen liegt ein Fehler in der BibTeX-Datei vor. Entweder fehlt der Key oder er ist verkehrt, es fehlt das @ vor der entsprechenden Gattung oder es gibt keinen entsprechenden Eintrag für die Gattung in der Initialisierungsdatei .mnmit2html.ini.

An error occured, no Documents added to WWW!
: Mit großer Wahrscheinlichkeit gibt es einen Fehler in der BibTeX-Datei. Näheres siehe voriger Punkt.

Input directory empty?
: Das Programm stoppt, wenn `$INPUT_DIR` leer ist. Sollte dieses Verzeichnis nicht leer sein, so kann ein Blick auf die Rechte der einzelnen Dateien im Verzeichnis nicht schaden. Zumindest die Leserechte der einzelnen Dateien müssen für die entsprechende Gruppe bzw. für alle anderen gesetzt sein!

/tmp/publikationen.bib.tex not writeable?
: Vermutlich wurde ein vorheriger Durchlauf von mnmlit2html.pl mit einem Fehler abgebrochen oder nicht vollständig durchgeführt. Als Folge davon werden einige temporäre Dateien nicht gelöscht. Abhilfe schafft hier, ein anderes temporäres Verzeichnis zu wählen oder den Dateibesitzer zu bitten, seine Dateien zu löschen.

Bilder werden nicht korrekt übersetzt
: LaTeX2HTML erstellt eine Datei `images.log`, die durch einen LaTeX-Lauf von `images.tex` entstanden ist. Finden sich in `images.log` ungewöhnliche Fehlermeldungen wie nicht gefundene Styles, so kann es sein, daß LaTeX2HTML die Bilder nicht von Encapsulated Postscript nach GIF übersetzen konnte. Fehlerursache ist häufig ein nicht korrekter `$TEXINPUTS`-Pfad. Abhilfe schafft u.U. ein Zurücksetzen von `$TEXINPUTS` mittels `TEXINPUTS=::`. Siehe hierzu auch die Beschreibung zu LaTeX2HTML[1].

  Sollten trotz dieser Maßnahmen die Bilder nicht korrekt oder gar nicht dargestellt werden, so ist zu prüfen, ob das Dokument mit einer LaTeX-Version 2.09 oder kleiner übersetzt wurde. Speziell LaTeX2HTML-97.1 verhält sich da nicht besonders tolerant. Abhilfe

[1] http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/

schafft ein Löschen der Dateien mit den Endungen *.aux* und *.toc* und anschließendes Neuübersetzen mit LaTeX2e im Kompatibilitätsmodus.

Bilder erscheinen zu klein oder zu groß.

`$FIGURE_SCALE_FACTOR` entsprechend setzen, um die Größe der Bilder zu skalieren. Siehe hierzu Kapitel 5 auf Seite 20.

Es erscheint keine Zusammenfassung

obwohl ein Eintrag in `@ZUSAMMENFASSUNG`[2] vorhanden ist und es ein Kapitel mit der Überschrift des Eintrages in `@ZUSAMMENFASSUNG` gibt. Zunächst ist zu überprüfen, ob Umlaute im LaTeX-Format, also mit doppelten Anführungszeichen gefolgt von einem Vokal, eingegeben wurden. Dabei sind die Anführungszeichen zu quoten, sprich: mit einem vorgestellten Backslash zu versehen und der Begriff selber in Anführungszeichen zu setzten.

Es erscheint eine „falsche" Zusammenfassung

Möglicherweise ist die Variable `$CHILDLINE` falsch gesetzt, so daß das Programm mnmlit2html.pl nicht das Ende der Zusammenfassung findet. Für die Versionen 96.1 und 97.1 von LaTeX2HTML sollte die im Sourcecode gesetzte Default-Einstellung richtig sein. Den Anfang der Zusammenfassung findet mnmlit2html.pl anhand dem Eintrag in `@ZUSAMMENFASSUNG` und der darauffolgenden Zeile. Beginnt diese mit einem Abschnittsbeginn (`<P>`) oder einem Überschriftsende (`<\H1>`), so gehören alle folgenden Zeilen bis zu `$CHILDLINE` zur Zusammenfassung und werden in einer Variablen abgelegt. Sollte sich daran in zukünftigen Versionen von LaTeX2HTML etwas ändern, so muß der Sourcecode entsprechend gepatcht oder die Variable `$CHILDLINE` anders gesetzt werden.

Die Numerierung der einzelnen Abschnitte fehlt teilweise

Normalerweise reicht es unter LaTeX, Sonderzeichen wie die deutschen Umlaute durch doppelte Hochkommata und dem anschließenden Vokal zu schreiben, wenn der Style `german.sty` eingebunden ist. LaTeX2HTML hingegen verlangt die ausführliche Schreibweise in Überschriften. Also:

ä ö ü ß	`\"a \"o \"u \ss{}`
Ä Ö Ü	`\"A \"O \"U`

statt

ä ö ü ß	`"a "o "u "s`
Ä Ö Ü	`"A "O "U`

Sollten trotz dieser Maßnahmen die Abschnitte nicht nummeriert werden, so ist zu prüfen, ob das Dokument mit einer LaTeX-Version 2.09 oder kleiner übersetzt wurde. Speziell LaTeX2HTML-97.1 verhält sich da nicht besonders tolerant. Abhilfe schafft ein Löschen der Dateien mit den Endungen *.aux* und *.toc* und anschließendes Neuübersetzen mit LaTeX2e im Kompatibilitätsmodus.

LaTeX2HTML dreht Schleifen

Dieses unangenehmen „Loopen" des Programms kann von nicht korrekt verwendeten

[2] siehe hierzu Abschnitt 3.3.2 auf Seite 14

LaTeX-Befehlen herrühren. Im Kapitel *Troubleshooting* der Dokumentation zu LaTeX2HTML[3] finden sich nähere Hinweise hierzu.

Was LaTeX2HTML bspw. gar nicht bekommt, sind im \author-Feld eingebundene Grafiken! Dann (und nur dann, soweit das bei den Tests von LaTeX2HTML auftrat) gibt es Loops von LaTeX2HTML.

In den folgenden Passagen muß die input-Zeile **unbedingt** auskommentiert werden:

```
\author{
%\centerline{\input{logo.tex}} \\   <--- ohne % => Loops!
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
\centerline{{\small Munich Network Management Team}}
\and Alexander Keller \\
  Faculty of Computer Science, Munich University of Technology \\
  Oettingenstr.\ 67, 80538 Munich, Germany \\
  E-Mail: keller@informatik.uni-muenchen.de}
\date{}
\maketitle
```

Das MNM-Team verwendet gelegentlich solche Konstruktionen, um z.B. das MNM-Logo in die oberste Zeile zu setzen. LaTeX2HTML neigt dabei jedoch zum Loopen.

Zur Vermeidung solcher Loops wäre auch die Einbettung der entsprechenden Zeile in folgende Umgebung denkbar:

```
   [...]
\begin{latexonly}
\centerline{\input{logo.tex}}
\end{latexonly}
   [...]
```

main.tex is newer than main.bbl: Please rerun latex and bibtex

Diese Meldung kann getrost übersehen werden. LaTeX2HTML meckert an dieser Stelle, wenn die *.tex*-Datei neuer ist als die dazugehörige *.bbl*-Datei.

[3] http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/

Kapitel 7

Zusammenfassung und Ausblick

Dieses Fortgeschrittenen-Praktikum eignete sich hervorragend, um sich eingehender mit BibTeX zu befassen, HTML in Grundzügen zu lernen sowie einen Einblick in die Programmiersprache *Perl5* zu bekommen.

Perl5 läßt sich schnell erlernen, da der Aufbau dieser Programmiersprache der natürlichen Sprache angelehnt ist. So ist es nahezu ein Genuß, mit *Perl5* nach regulären Ausdrücken zu suchen und diese weiterzuverarbeiten. Diese regulären Ausdrücke sind eine wesentliche Erleichterung im Umgang mit Strings. *Perl5* handhabt diese in einer Art und Weise, die es v.a. für Personen, die unter UNIX schon Erfahrung mit Programmen wie sed oder awk gesammelt haben, erleichtert damit umzugehen.

Perl5 im Allgemeinen ist aber auch eine Kompromißlösung zwischen sauberem, typisiertem Stil und unsauberen, schnellen Lösungen, da es sich schnell und unkompliziert handhaben läßt wie eine Shell, gleichzeitig aber auch so leistungsfähig und performant wie eine moderne Programmiersprache ist.

Das Programm selber läßt sich natürlich noch manigfaltig erweitern. So wäre z.B. eine Erweiterung dahingehend denkbar, daß auch Dokumente im Word- oder RTF-Format berücksichtigt werden, statt nur Postscript- und PDF-Dokumente. Bei einer vernünftigen Umsetzung des Word-Formats nach HTML, könnte auch an eine Erweiterung derart gedacht werden, daß man den von Word erzeugten HTML-Teil an die geeignete Stelle im Dokumentationsbaum setzt, die momentan der von LaTeX2HTML erzeugte Teil innehat.

Auch eine eingehende Prüfung des Quelltextes auf mögliche Verweise, wie z.B. HTML-Adressen, die noch nicht mit `\htmladdnormallink` eingeleitet werden, dergleichen mit e-mail Adressen, Literaturverweisen etc. ist überlegenswert. Kurzum, die vorliegende Lösung ist sicher noch ausbaufähig, wenn auch zukünftige Versionen von LaTeX2HTML einige neue Features bringen werden.

Literaturverzeichnis

[HK96] Stephen Heilbronner and Alexander Keller. *Regeln zur Erstellung von Dokumentation zu Diplomarbeiten, Fortgeschrittenenpraktika, Seminarvorträgen etc.* Institut für Informatik, Universität München, Oettingenstr. 67, 80538 München, Germany, Oktober 1996.

[WCS92] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472, 1st edition, March 1992.

[WCS96] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472, 2nd edition, September 1996.

[Won93] Reinhard Wonneberger. *Kompaktführer LaTeX.* Addison-Wesley, Bonn, 3rd edition, 1993.

Anhang A

Listing zur Inidatei `.mnmlit2html.ini`

```
# ----------------------------------------------------
#           TEIL 1
# ----------------------------------------------------

## Latex2html findet sich hier:
$LATEX2HTML_DIR = "/usr/local/dist/bin";

## Ort, an dem  die fertigen HTML-Dokumente
## abgelegt werden.
$HTML_DATA_DIR = "/proj/Literatur/public-htdocs/MNMPub";

## Literaturverzeichnis, hier werden die Dokumente abgelegt,
## die zur Bearbeitung durch mnnlit2html anstehen.
$INPUT_DIR = "/proj/Literatur/MNM/Upload";

## Bibliographie-Verzeichnis,
## enthaelt Informationen ueber die
## einzelnen Dokumente und deren Verfassern
$BIB_DIR = "/proj/Literatur/MNM/Bib";

## Temporaeres Verzeichnis, dieses sollte
## ausreichend Platz zur Verfuegung stellen
$TMP = "/tmp";

## Bezeichnet den HTML-Befehl, der Haupteil und
## Fuss voneinander trennt. Wird in latex2html.config
## gesetzt und sollte entsprechend uebernommen werden,
## damit trotz fehlendem Abstract eine Zusammenfassung
## korrekt gefunden wird.
## Default-Eintrag ist fuer latex2html-{96.1,97.1} gueltig
$CHILDLINE = "<BR>.*<HR>";


# -----------------------------------------------------
#         Teil 2
```

```
# ------------------------------------------------------

## ZUSAMMENFASSUNG
@ZUSAMMENFASSUNG = ("Einf\"uhrung", "Zusammenfassung");

##
## Die erste Zeile gibt an, nach welchen Feldern
## in den einzelnen Bibliographie-Eintragen gesucht wird.
## Die zweite Zeile muss Eintraege aus der ersten enthalten
## und gibt an, welche Eintraege publiziert werden
##
## ARTICLE
@article =
   ("author","title","journal","year","type","month","note","key");
@article_to_publish =
   ("author","title","journal","month","year");

## BOOK
@book =
   ("author","editor","title","publisher","year","volume",
    "series","address","edition","month","note","key");
@book_to_publish =
   ("author","editor","title","publisher","year","volume",
    "series","address","edition","month","note","key");

## BOOKLET
@booklet =
   ("title","author","howpublished","address","month",
    "year","note","key");
@booklet_to_publish =
   ("title","author","howpublished","address","month",
    "year","note","key");

## CONFERENCE
@conference =
   ("author","title","book","title","year","editor","pages",
    "organization","publisher","address","month","note","key");
@conference_to_publish =
   ("author","title","book","title","year","editor","pages",
    "organization","publisher","address","month","note","key");

## INBOOK
@inbook =
   ("author","editor","title","chapter","pages","publisher",
    "year","volume","series","address","edition","month","note","key");
@inbook_to_publish =
   ("author","editor","title","chapter","pages","publisher",
    "year","volume","series","address","edition","month","note","key");

## INCOLLECTION
@incollection =
   ("author","title","booktitle","year","publisher","editor",
    "chapter","pages","address","month","note","key");
```

```
@incollection_to_publish =
   ("author","title","booktitle","year","publisher","editor",
    "chapter","pages","month");

## INPROCEEDINGS
@inproceedings =
   ("author","title","booktitle","publisher","school",
    "year","type","month","key");
@inproceedings_to_publish =
   ("author","title","booktitle","publisher","school",
    "month","year");

## MANUAL
@manual =
   ("title","author","organization","address","edition",
    "month","year","note","key");
@manual_to_publish =
   ("title","author","organization","address","edition",
    "month","year","note","key");

## MASTERSTHESIS
@mastersthesis =
   ("author","title","school","year","type","month","key");
@mastersthesis_to_publish =
   ("author","title","school","month","year");

## MISC
@misc =
   ("author","title","howpublished","month",
    "year","note","key");
@misc_to_publish =
   ("author","title","howpublished","month",
    "year","note","key");

## PHDTHESIS
@phdthesis =
   ("author","title","school","year","type","address",
    "month","note","key");
@phdthesis_to_publish =
   ("note","author","title","school","year","address");

## PROCEEDINGS
@proceedings =
   ("title","year","editor","publisher","organization",
    "address","month","note","key");
@proceedings_to_publish =
   ("title","year","editor","publisher","organization",
    "address","month","note","key");

## TECHREPORT
@techreport =
   ("author","title","institution","year","type",
    "month","key");
```

```
@techreport_to_publish =
   ("author","title","institution","month","year");

## UNPUBLISHED
@unpublished =
   ("author","title","note","month","year","key");
@unpublished_to_publish =
   ("author","title","note","month","year","key");


# --------------------------------------------------------
#            Teil 3
# --------------------------------------------------------

## MONTHS
## This is an associative array containing the names of
## each month of the year. At the left you can see the
## abbreviation (3 chars long) and at the right the
## long Version.

## The german version
## (comment it out, if you prefer the english version below)
%months = (
        "jan"     => "Januar",
        "feb"     => "Februar",
        "mar"     => "M\&auml\;rz",
        "apr"     => "April",
        "may"     => "Mai",
        "jun"     => "Juni",
        "jul"     => "Juli",
        "aug"     => "August",
        "sep"     => "September",
        "oct"     => "Oktober",
        "nov"     => "November",
        "dec"     => "Dezember",
);

## The english version: (comment it out if necessary)
#%months = (
#       "jan"     =>      "January",
#       "feb"     =>      "February",
#       "mar"     =>      "March",
#       "apr"     =>      "April",
#       "may"     =>      "May",
#       "jun"     =>      "June",
#       "jul"     =>      "July",
#       "aug"     =>      "August",
#       "sep"     =>      "September",
#       "oct"     =>      "October",
#       "nov"     =>      "November",
#       "dec"     =>      "December",
#);
```

```
## TYPES
## This is an associative array containing the type
## of a document on the left and its header on the right.
%types = (
   "diplomarbeiten"  => "Diplomarbeiten / Master's Theses",
   "dissertationen"  => "Dissertationen / Ph.D. Theses",
   "fopras"          => "Systementwicklungsprojekte / Advanced Practicals",
   "projekte"        => "Projektberichte / Project Reports",
   "publikationen"   => "Publikationen / Publications"
);
```

Anhang B

Listing zum Script mnmlit2html.pl

```
#!/usr/local/dist/bin/perl
require 5.002;

use File::Basename;

## Speichere die Kommandozeilen-Parameter ab.
$argv = join(' ',@ARGV);

## Hole $HOME aus der Environment Variable
push(@INC,$ENV{'HOME'});
$HOME = $ENV{'HOME'};

## Versionsnummer, Patchlevel und Erscheinungsdatum
$TPATCHLEVEL = "1a";
$TVERSION = "97";
$RELDATE = "(Nov 26, 1997)";

$MNMlit2htmlV_SHORT = $TVERSION . '.' . "$TPATCHLEVEL";
$MNMlit2htmlVERSION = $MNMlit2htmlV_SHORT . ' ' . $RELDATE;

## Hinweise zum Author
$AUTHORADDRESS =
   "http://www.informatik.tu-muenchen.de/~allgeyer/index.html";


## ------------------------------------------------------------------
##  BEGIN             Configurabel parameter                   BEGIN
## ------------------------------------------------------------------

## Set debug-level
## Debugging off
$DEBUG = 0;

## Wo befindet sich das Initialisierungsprogramm?
## Default ist das Verzeichnis in dem sich das
## Hauptprogramm befindet. Wird das, Hauptprogramm
## "uber Links referenziert, so _muss_ diese Variable
## explizit gesetzt werden.
```

```
$INI_DIR = dirname( &getcwd . "/" . "$0" );

# ---------------------------------------------------
#            TEIL 1
# ---------------------------------------------------

## Latex2html findet sich hier:
$LATEX2HTML_DIR = "/usr/local/dist/bin";

## Ort, an dem  die fertigen HTML-Dokumente
## abgelegt werden.
$HTML_DATA_DIR = "/proj/Literatur/public-htdocs/MNMPub";

## Literaturverzeichnis, hier werden die Dokumente abgelegt,
## die zur Bearbeitung durch mnnlit2html anstehen.
$INPUT_DIR = "/proj/Literatur/MNM/Upload";

## Bibliographie-Verzeichnis,
## enthaelt Informationen ueber die
## einzelnen Dokumente und deren Verfassern
$BIB_DIR = "/proj/Literatur/MNM/Bib";

## Temporaeres Verzeichnis, dieses sollte
## ausreichend Platz zur Verfuegung stellen
$TMP = "/tmp";

## Bezeichnet den HTML-Befehl, der Hauptteil und
## Fuss voneinander trennt. Wird in latex2html.config
## gesetzt und sollte entsprechend uebernommen werden,
## damit trotz fehlendem Abstract eine Zusammenfassung
## korrekt gefunden wird.
## Default-Eintrag ist fuer latex2html-{96.1,97.1} gueltig
$CHILDLINE = "<BR>.*<HR>";


# -----------------------------------------------------
#          Teil 2
# -----------------------------------------------------

## ZUSAMMENFASSUNG
@ZUSAMMENFASSUNG = ("Einf\"uhrung", "Zusammenfassung");

##
## Die erste Zeile gibt an, nach welchen Feldern
## in den einzelnen Bibliographie-Eintragen gesucht wird.
## Die zweite Zeile muss Eintraege aus der ersten enthalten
## und gibt an, welche Eintraege publiziert werden
##
## ARTICLE
@article =
   ("author","title","journal","year","type","month","note","key");
@article_to_publish =
   ("author","title","journal","month","year");
```

```
## BOOK
@book =
   ("author","editor","title","publisher","year","volume",
    "series","address","edition","month","note","key");
@book_to_publish =
   ("author","editor","title","publisher","year","volume",
    "series","address","edition","month","note","key");

## BOOKLET
@booklet =
   ("title","author","howpublished","address","month",
    "year","note","key");
@booklet_to_publish =
   ("title","author","howpublished","address","month",
    "year","note","key");

## CONFERENCE
@conference =
   ("author","title","book","title","year","editor","pages",
    "organization","publisher","address","month","note","key");
@conference_to_publish =
   ("author","title","book","title","year","editor","pages",
    "organization","publisher","address","month","note","key");

## INBOOK
@inbook =
   ("author","editor","title","chapter","pages","publisher",
    "year","volume","series","address","edition","month","note","key");
@inbook_to_publish =
   ("author","editor","title","chapter","pages","publisher",
    "year","volume","series","address","edition","month","note","key");

## INCOLLECTION
@incollection =
   ("author","title","booktitle","year","publisher","editor",
    "chapter","pages","address","month","note","key");
@incollection_to_publish =
   ("author","title","booktitle","year","publisher","editor",
    "chapter","pages","month");

## INPROCEEDINGS
@inproceedings =
   ("author","title","booktitle","publisher","school",
    "year","type","month","key");
@inproceedings_to_publish =
   ("author","title","booktitle","publisher","school",
    "month","year");

## MANUAL
@manual =
   ("title","author","organization","address","edition",
    "month","year","note","key");
```

```
@manual_to_publish =
   ("title","author","organization","address","edition",
    "month","year","note","key");

## MASTERSTHESIS
@mastersthesis =
   ("author","title","school","year","type","month","key");
@mastersthesis_to_publish =
   ("author","title","school","month","year");

## MISC
@misc =
   ("author","title","howpublished","month",
    "year","note","key");
@misc_to_publish =
   ("author","title","howpublished","month",
    "year","note","key");

## PHDTHESIS
@phdthesis =
   ("author","title","school","year","type","address",
    "month","note","key");
@phdthesis_to_publish =
   ("note","author","title","school","year","address");

## PROCEEDINGS
@proceedings =
   ("title","year","editor","publisher","organization",
    "address","month","note","key");
@proceedings_to_publish =
   ("title","year","editor","publisher","organization",
    "address","month","note","key");

## TECHREPORT
@techreport =
   ("author","title","institution","year","type",
    "month","key");
@techreport_to_publish =
   ("author","title","institution","month","year");

## UNPUBLISHED
@unpublished =
   ("author","title","note","month","year","key");
@unpublished_to_publish =
   ("author","title","note","month","year","key");


# --------------------------------------------------------
#             Teil 3
# --------------------------------------------------------

## MONTHS
## This is an associative array containing the names of
```

```
## each month of the year. At the left you can see the
## abbreviation (3 chars long) and at the right the
## long Version.

## The german version
## (comment it out, if you prefer the english version below)
%months = (
        "jan"     => "Januar",
        "feb"     => "Februar",
        "mar"     => "M\&auml\;rz",
        "apr"     => "April",
        "may"     => "Mai",
        "jun"     => "Juni",
        "jul"     => "Juli",
        "aug"     => "August",
        "sep"     => "September",
        "oct"     => "Oktober",
        "nov"     => "November",
        "dec"     => "Dezember",
);

## The english version: (comment it out if necessary)
#%months = (
#       "jan"     =>      "January",
#       "feb"     =>      "February",
#       "mar"     =>      "March",
#       "apr"     =>      "April",
#       "may"     =>      "May",
#       "jun"     =>      "June",
#       "jul"     =>      "July",
#       "aug"     =>      "August",
#       "sep"     =>      "September",
#       "oct"     =>      "October",
#       "nov"     =>      "November",
#       "dec"     =>      "December",
#);


## TYPES
## This is an associative array containing the type
## of a document on the left and its header on the right.
%types = (
   "diplomarbeiten"  => "Diplomarbeiten / Master's Theses",
   "dissertationen"  => "Dissertationen / Ph.D. Theses",
   "fopras"          => "Systementwicklungsprojekte / Advanced Practicals",
   "projekte"        => "Projektberichte / Project Reports",
   "publikationen"   => "Publikationen / Publications"
);

## -------------------------------------------------------------------
##  END               Configurabel parameter                    END
## -------------------------------------------------------------------
```

```
## Statusausgabe
print <<EOF;
This is MNMlit2html.pl Version $MNMlit2htmlVERSION by Peter Allgeyer,
Institut fuer Informatik, Technical University of Munich.

EOF

## Process debug switch
if ( $argv =~ /-debug/ )
   {
   ## -debug wurde eingegeben
   $DEBUG = 1;
   }
## end (if)


## .mnmlit2html.ini einlesen, wenn vorhanden
##
if (! (-f "$ENV{'HOME'}/.mnmlit2html.ini") and
   ! ( (! (&getcwd eq $ENV{'HOME'} )) &&
   (-f "./.mnmlit2html.ini")) and
   ! (-f "$INI_DIR/.mnmlit2html.ini"))
   {
   ## Kein .mnmlit2html.ini file gefunden?
   ## Dann Fehlermeldung.
   print "\nNo .mnmlit2html.ini file found!\n";
   print "Using defaults, declared in mnmlit2html.pl.\n";
   print "Hope that's Ok.\n";
   }

## -----------------------------------------------------------------
##  there shouldn't be no need to change anything behind this line
## -----------------------------------------------------------------
else
   {
   ## .mnmlit2html.ini einlesen, wo vorhanden
   ##
   ## ACHTUNG:
   ## Jede gefundene Datei wird gescannt!!!
   ##
   ## Suchreihenfolge:
   ## 1) im Verzeichnis, in dem mnmlit2html.pl liegt
   ##    (bei Symlinks muss die Variable $INI_DIR
   ##     per Hand gesetzt werden, s.o.!!!)
   ## 2) aktuelles Verzeichnis
   ## 3) Homeverzeichnis
   ##
   if (-f "$INI_DIR/.mnmlit2html.ini")
      {
      ## .mnmlit2html.ini einlesen, wenn in $INI_DIR vorhanden
      (require("$INI_DIR/.mnmlit2html.ini"));

      ## Debugging
```

```
      if ( $DEBUG )
         {
         print ".mnmlit2html.ini im Verzeichnis,";
         print "in dem mnmlit2html.pl liegt, gefunden.\n";
         }
      ## end (if)
      }
   if ( (! (&getcwd eq $ENV{'HOME'} )) && (-f "./.mnmlit2html.ini"))
      {
      ## .mnmlit2html.ini einlesen,
      ## wenn im aktuellen Verzeichnis vorhanden
      (require("./.mnmlit2html.ini"));

      ## Debugging
      print ".mnmlit2html.ini im aktuellen Verzeichnis gefunden.\n"
      if $DEBUG;
      }
   if (-f "$ENV{'HOME'}/.mnmlit2html.ini")
      {
      ## .mnmlit2html.ini einlesen,
      ## wenn im Home-Verzeichnis vorhanden
      require("$ENV{'HOME'}/.mnmlit2html.ini");

      ## Debugging
      print ".mnmlit2html.ini im Home-Verzeichnis gefunden.\n"
      if $DEBUG;
      }
   print "\n" if $DEBUG;
   }
## end if


## Process switches
while ($ARGV[0] =~ /^-/)
   {
   $_ = shift;

   ## -debug wurde eingegeben
   if (/^-debug$/)
      {
      $DEBUG = 1;

      ## Debugging
      print "Recognised -debug.\n" if $DEBUG;
      }

   ## -latex2htmldir wurde eingegeben
   elsif (/^-latex2htmldir$/)
      {
      $_ = shift;
      ((($LATEX2HTML_DIR) = /^(.+)$/) || &usage &&
      print("Unrecognised value for -latex2htmldir: $_\n") && die);
```

```
   ## Debugging
   print "Recognised -latex2htmldir $1.\n" if $DEBUG;
   }

## -htmldir wurde eingegeben
elsif (/^-htmldir$/)
   {
   $_ = shift;
   ((($HTML_DATA_DIR) = /^(.+)$/) || &usage &&
   print("Unrecognised value for -htmldir: $_\n") && die);

   ## Debugging
   print "Recognised -htmldir $1.\n" if $DEBUG;
   }

## -inputdir wurde eingegeben
elsif (/^-inputdir$/)
   {
   $_ = shift;
   ((($INPUT_DIR) = /^(.+)$/) || &usage &&
   print("Unrecognised value for -inputdir: $_\n") && die);

   ## Debugging
   print "Recognised -inputdir $1.\n" if $DEBUG;
   }

## -bibdir wurde eingegeben
elsif (/^-bibdir$/)
   {
   $_ = shift;
   ((($BIB_DIR) = /^(.+)$/) || &usage &&
   print("Unrecognised value for -bibdir: $_\n") && die);

   ## Debugging
   print "Recognised -bibdir $1.\n" if $DEBUG;
   }

## -tmpdir wurde eingegeben
elsif (/^-tmpdir$/)
   {
   $_ = shift;
   ((($TMP) = /^(.+)$/) || &usage &&
   print("Unrecognised value for -tmpdir: $_\n") && die);

   ## Debugging
   print "Recognised -tmpdir $1.\n" if $DEBUG;
   }

## -childline wurde eingegeben
elsif (/^-childline$/)
   {
   $_ = shift;
   ((($CHILDLINE) = /^(.+)$/) || &usage &&
```

```
      print("Unrecognised value for -childline: $_\n") && die);

      ## Debugging
      print "Recognised -childline $1.\n" if $DEBUG;
      }

   ## -h(elp) wurde eingegeben
   elsif (/^-h(elp)?$/)
      {
      ## Debugging
      print "Recognised -(h)elp.\n" if $DEBUG;

      ## Rufe Hilfestellung zum Programmaufruf auf und
      ## beende dann das laufende Programm mit Status 0
      &usage;
      exit 0;
      }

   ## Keinen Switch erkannt?
   ## Dann rufe Hilfestellung zum Programmaufruf
   ## auf und beende dann das laufende Programm
   else
      {
      &usage;
      die "Unrecognised switch: $_\n";
      }
   }
## end (while)

## No arguments left!!
(&usage && die "Unrecognised switch: @ARGV!\n") if @ARGV;


# #################### begin (main) ####################

## Debugging
print "Arguments left after processing switches: @ARGV\n\n" if $DEBUG;

## $INPUT_DIR nach *.tgz, *.tar.gz, *.tar.Z durchsuchen
@inputs = glob ("$INPUT_DIR/{*.tgz,*.tar.gz,*.tar.Z}");

## Fehlermeldung und Abbruch, wenn keine Eingabedateien vorhanden sind
die "Input directory empty?\n" if (!defined @inputs);

## Trenne von jeder Eingabedatei Pfad und Endung ab
foreach $inputfile ( @inputs )
   {
   my ($name, $path, $suffix) =
   fileparse ($inputfile, "\.tgz", "\.tar\.gz", "\.tar\.Z");

   ## Packe gefundene Datei aus
   print "Unpack $name$suffix to $TMP ...\n";
```

```
   ## entpacke Archiv in temporaerem Verzeichnis
   system ("( cd $TMP; gunzip < $inputfile | tar xf - 2> /dev/null )");

   $inputfile = $name;
   }

## Statusausgabe
print "\nRunning 'mnmlit2html.pl $argv @inputs' ...\n\n";


## Neues Directory anlegen
&new_dir ("$HTML_DATA_DIR") unless (-d "$HTML_DATA_DIR" );


## Zusammenfassung abstrahieren
## @ZUSAMMENFASSUNG beinhaltet eine Liste von Schlagwoertern,
## nach denen im Zusammenhang mit einer Kurzfassung fuer den
## vorliegenden Text gesucht wird.
## @ZUSAMMENFASSUNG wird in .mnmlit2html.ini spezifiziert
foreach $ZUSAMMENFASSUNG (@ZUSAMMENFASSUNG)
   {
   $ZUSAMMENFASSUNG = &substitute_umlaut ($ZUSAMMENFASSUNG);
   }
## end (foreach)



##############################################################
## Zunaechst wird eine Datenbank erzeugt. Sie besteht
## aus saemtlichen Arbeiten, die in den angegebenen
## Bibliographien gefunden werden. Der Aufbau einer Arbeit
## wird gemaess den Angaben in der .ini-Datei vorgenommen.
##############################################################

## Orte fuer *.bib-Files
@BIB_DIR = split /:/, $BIB_DIR;

foreach $DIR (@BIB_DIR)
   {
   unless ( -d $DIR  )
      {
      warn "WARNING: Can't open $DIR!\n";
      next;
      }

   opendir BIB_DIR, "$DIR";
      @allfiles =
         (@allfiles, grep /bib$/, map "$DIR/$_", readdir BIB_DIR);
   closedir BIB_DIR;

   ## Debugging
   print "Found *.bib files: @allfiles\n" if $DEBUG;
   }
```

```
## end (foreach)


## Erzeuge HTML Titelseiten fuer Diplomarbeiten, Fopras etc.
if (defined @allfiles)
   {
   &create_kind_title_page (@allfiles);
   }
else
   {
   die "No \@BIB_DIR in .mnmlit2html?\n";
   }


## Nun erstelle HTML Titelseiten fuer
## jede eingegebene Arbeit
KEY: foreach $inputfile ( @inputs )
   {
   ## Zusammenfassung muss neu geholt werden
   undef $abstract;

   ## $key enthaelt aktuelle Arbeit
   $key = $inputfile;

   ## Art der Arbeit
   ($kind = $authmap{$key}) ||
   (print "WARNING: No $key in *.bib files!\n" && next KEY);

   ## Debugging
   print "KIND: $kind; KEY: $key\n" if $DEBUG;

   ## Liefert Mehrzahl des angegebenen Typs zurueck
   ## $type = &get_type($bibmap{$key});
   $type = $bibmap{$key};

   ## Zielverzeichnis, falls vorhanden, vor dem
   ## Reinschreiben der Daten loeschen
   &rm_dir ("$HTML_DATA_DIR/\u\L$type\E/$key") if
   ( -d "$HTML_DATA_DIR/\u\L$type\E/$key" );

   ## Erzeuge Verzeichnisstruktur
   &new_dir ("$HTML_DATA_DIR/\u\L$type\E/$key") unless
   ( -d "$HTML_DATA_DIR/\u\L$type\E/$key" );

   ## liefert Name des TeX-files zurueck
   ($texfile = &get_texfile ($key));

   ## LaTeX Sourcefile gefunden?
   ## Dann benenne HTML-file entsprechend
   if ( $NO_TEXFILE =~ 0 )
      {
      ## Name des HTML-files
      ## aus <name>.tex herausfiltern ...
```

```
   ($name, $path, $suffix) = fileparse ($texfile, "\.tex");

   ## ... und mit entsprechender Endung versehen
   $htmlfile = "$name\.html";
   }

## Suche nach evtl. vorhandenem
## Postscript- und PDF-File
$psfile = &get_psfile ($key);
$pdffile = &get_pdffile ($key);

## Debugging
print "HTMLFILE: $htmlfile\n" if $DEBUG;
print "PSFILE:   $psfile\n" if $DEBUG;
print "PDFFILE:  $pdffile\n" if $DEBUG;

## Copy $psfile nach $HTML_DATA_DIR/$type/$key/Postscript-Version
&copy_ps_file ($key, $type, $psfile);

## Copy $pdffile nach $HTML_DATA_DIR/$type/$key/PDF-Version
&copy_pdf_file ($key, $type, $pdffile);

## Abfrage nach NOHTML
&no_html_file($key);

## LaTeX2html main.tex,
&create_html_structur ($key, $type, $texfile);

## HTML Titelseite erstellen
&create_work_title_page ($key, $type, $htmlfile, $psfile);

## Loesche HTML-Baum, weil NOHTML gesetzt war!
&rm_dir ("$HTML_DATA_DIR/\u\L$type\E/$key/HTML-Version") if $NO_HTMLFILE =~ 1;

## Loesche Datei aus dem Inputverzeichnis
&rm_dir ("$INPUT_DIR/$inputfile\.*");

## Loesche temporaer angelegte Verzeichnisse und Dateien
&rm_dir ("$TMP/$inputfile");
&rm_dir ("$TMP/*.css ");
&rm_dir ("$TMP/*.html")
&rm_dir ("$TMP/*.tex")
&rm_dir ("$TMP/*.pl")
&rm_dir ("$TMP/.*.db");
&rm_dir ("$TMP/.*.dir");
&rm_dir ("$TMP/.*.pag");

## Erfolgsmeldung
print "Added to WWW: $inputfile\n\n";

## @success gibt an, welche Arbeiten fehlerlos
## ins WWW gehaengt wurden
push (@success, $inputfile)
```

```
   }
## end (foreach)

## Erfolgsmeldung
if (defined @success)
   {
   print "Documents requested to add to WWW: @inputs.\n";
   print "Successfully added to WWW: @success\n";
   }
else
   {
   print "An error occured, no Documents added to WWW!\n"
   }

# ############################## end (main) ####################################



# ######################### begin (subroutines) ################################

##############################################################
## Erzeuge eine HTML Titelseite, fuer jede
## Art von Arbeit (Fopra, Diplomarbeit, Paper etc.)
##############################################################
sub create_kind_title_page
   {
   local (@allfiles) = @_;
   local ($type, $html_bib_file, $file);

   foreach $file (@allfiles)
      {
      ## Statusausgabe
      print "Loading $file...\n";

      ## HTML Seite aus BIB-File erzeugen
      &create_html_bib_file ($file);

      ## HTML-BIB-FILE scannen und Variablen in einen assoziativen Vektor schreiben
      $type = basename ($file, ".bib");
      print "TYPE: $type\n" if $DEBUG;

      $html_bib_file = "$type" . ".bib.html";
      print "HTML_BIB_FILE: $html_bib_file\n" if $DEBUG;

      ## Statusausgabe
      print "\nLoading $TMP/$html_bib_file...\n";
      print "Scanning ...\n\n";

      ## Liefert Mehrzahl des angegebenen Typs zurueck
      ## $type = &get_type($type);

      ## Verzeichnisstruktur aufbauen
      &new_dir ("$HTML_DATA_DIR/\u\L$type") unless
```

```
      ( -d "$HTML_DATA_DIR/\u\L$type");

      ## Datei nach Inhalt scannen
      &scan_html_bib_file ("$TMP/$html_bib_file", "$type");
      }
   ## end (foreach)
   }
## end (sub create_kind_title_page)



#############################################################
## Rufe latex2html auf,
## Destinationdirectory ist $HTML_DATA_DIR/$type/$key
#############################################################
sub create_html_structur
   {
   local ($key, $type, $texfile) = @_;

   ## Debugging
   print "\nKEY: $key, TYPE: $type\n" if $DEBUG;

   ## Destination
   $HTML_DIR = "$HTML_DATA_DIR/\u\L$type\E/$key/HTML-Version";

   # Verzeichnis loeschen, falls vorhanden
   &rm_dir ("$HTML_DIR") if ( -d "$HTML_DIR" );

   ## Lege Verzeichnisse an, wenn sie nicht schon existieren
   &new_dir ("$HTML_DIR") unless ( -d "$HTML_DIR");

   ## Debugging
   print "Aufruf von latex2html mit folgender LaTeX Main-Datei: $texfile\n"
   if $DEBUG;

   ## Statusausgabe
   if ( $NO_TEXFILE =~ 0 )
      {
      print "Running LaTeX2HTML for $inputfile:\n\n";
      }

   -e "$LATEX2HTML_DIR/latex2html" or
   die "No such file $LATEX2HTML_DIR/latex2html.\n";

   ## latex2html mit Argumenten aufrufen:
   ##
   ##  Subdirectory anlegen
   ##  |    kein Abschnitt: About this document ...
   ##  |    |
   ##  |    |
   ##  |    |__________________________________________.
   ##  |_________________________________.              |
   ##                                  .|.           .__|_.
```

```
   system ("$LATEX2HTML_DIR/latex2html -dir $HTML_DIR -info 0 $texfile")
   if $NO_TEXFILE =~ 0;
   }
## end (sub create_html_structur)



############################################################
## Erzeuge eine HTML Titelseite fuer jede uebergebene
## Arbeit aus den Informationen, die in
## der Datenbank gespeichert wurden
############################################################
sub create_work_title_page
   {
   local ($key, $type, $htmlfile, $psfile) = @_;
   local ($output) = "$HTML_DATA_DIR/\u\L$type\E/$key/$key.shtml";
   local (@nf, @node_files, @not_node_files);

   ## $HTML_DIR wurde in &create_html_structur spezifiziert
   ## @node_files enthaelt alle Dateien mit Namen node\d+.html
   unless ( -d $HTML_DIR  )
      {
      warn "WARNING: Can't open $HTML_DIR, so this file won't be added!\n";
      next KEY;
      }

   opendir HTMLDIR, "$HTML_DIR";
      @node_files = (@node_files, grep /node\d+\.html$/,
      map "$HTML_DIR/$_", readdir HTMLDIR);
   closedir HTMLDIR;

   ## $HTML_DIR wurde in &create_html_structur spezifiziert
   ## @not_node_files enthaelt alle Dateien mit der Endung .html,
   ## die nicht mit node\d+.html oder footnote.html heissen
   opendir HTMLDIR, "$HTML_DIR";
      @not_node_files = (@not_node_files, grep /.*\.html$/,
      grep !/(node\d+|footnode)\.html$/, map "$HTML_DIR/$_", readdir HTMLDIR);
   closedir HTMLDIR;

   ## Durchsuche main.html
   foreach $html_page (@not_node_files)
      {
      $abstract = &get_abstract ($html_page, $key);
      last if ("$abstract" ne "");
      }
   ## end (foreach)

   ## Durchsuche alle *.html -files
   if ("$abstract" eq "")
      {
      foreach $html_page (@node_files)
         {
         $abstract = &get_abstract ($html_page, $key);
```

```
        last if ("$abstract" ne "");
        }
      ## end (foreach)
      }
   ## end (if)

   ## Durchsuche alle *.html -files nach @ZUZSAMMENFASSUNG
   if ("$abstract" eq "")
      {
      foreach $html_page (@node_files)
         {
         $abstract = &get_introduction ($html_page, $key);
         last if ("$abstract" ne "");
         }
      ## end (foreach)
      }
   ## end (if)

   print "Keine Zusammenfassung gefunden!\n" if ("$abstract" eq "");

   ## HTML-Header erzeugen und nach $output schreiben
   &print_header ($type, $authmap{$key}, $key, $output);

   ## HTML-Body erzeugen und nach $output schreiben
   &print_body ($type, $authmap{$key}, $key, $abstract, $output, $htmlfile);

   ## HTML-Footer erzeugen und nach $output schreiben
   &print_footer ($output);

   ## HTML - Inhaltsangabe erzeugen
   ## Diese Seite enthaelt eine Liste aller Arbeiten, die im dazugehoerigen
   ## BIB-File erwaehnt sind. Per Hyperlink ist eine kleine Zusammenfassung
   ## der angewaehlten Arbeit zu erreichen.

   &print_inhalt ($type, $key);
   }
## end (sub create_work_title_page)


#####################################################
## Eingabe Fehler
#####################################################
sub usage
   {
   print <<EOF;

Usage: mnmlit2html.pl
   [-latex2htmldir <directory>]
   [-bibdir <directory>]
   [-inputdir <directory>]
   [-htmldir <directory>]
   [-tmpdir <directory>]
   [-childline <reg-expr>]
```

```
   [-debug]
   [-h(elp)]

EOF
   }
## end (sub usage)



######################################################
## Get current working directory
## Aktuelles Arbeitsverzeichnis aufloesen
######################################################
sub getcwd
   {
   local($cwd);
   chop($cwd = `pwd`);
   $cwd;
   }
## end (sub getcwd)



######################################################
## Loescht ein Verzeichnis rekursiv
######################################################
sub rm_dir
   {
   ## Verzeichnis rekursiv loeschen,
   ## falls es vorhanden ist und
   ## der EUID des ausfuehrenden Programms
   ## gehoert.
   local($_) = @_;
   system ("rm -rf $_");
   }
## end (sub rm_dir);


######################################################
## Erzeugt ein neues Directory, falls moeglich
######################################################
sub new_dir
   {
   ## Verzeichnis anlegen und
   ## mit Rechten rwxrwsr-x versehen,
   ## damit Unterverzeichnisse gleich der
   ## richtigen Gruppe gehoeren und von
   ## dieser schreibbar sind.
   local($_) = @_;
   mkdir($_, oct(755)) ||
   warn "mkdir: cannot make directory `$_': Permission denied\n";
   system ("chmod 775 $_") if ( -d "$_" );
   system ("chmod g+ws $_") if ( -d "$_" );
```

```
   }
## end (sub new_dir);


#####################################################
## Latex-Umlaute durch HTML-Umlaute ersetzen
#####################################################
sub substitute_umlaut
   {
   local ($umlaut) = @_;

   $umlaut =~ s/\"a/\\\&\\\#228\\\;/g; ## "a --> &#228;
   $umlaut =~ s/\"o/\\\&\\\#246\\\;/g; ## "o --> &#246;
   $umlaut =~ s/\"u/\\\&\\\#252\\\;/g; ## "u --> &#252;
   $umlaut =~ s/\"A/\\\&\\\#196\\\;/g; ## "A --> &#196;
   $umlaut =~ s/\"O/\\\&\\\#214\\\;/g; ## "O --> &#214;
   $umlaut =~ s/\"U/\\\&\\\#220\\\;/g; ## "U --> &#220;
   $umlaut =~ s/\"s/\\\&\\\#223\\\;/g; ## "s --> &#223;

   $umlaut =~ s/\\\"a/\\\&\\\#228\\\;/g; ## "a --> &#228;
   $umlaut =~ s/\\\"o/\\\&\\\#246\\\;/g; ## "o --> &#246;
   $umlaut =~ s/\\\"u/\\\&\\\#252\\\;/g; ## "u --> &#252;
   $umlaut =~ s/\\\"A/\\\&\\\#196\\\;/g; ## "A --> &#196;
   $umlaut =~ s/\\\"O/\\\&\\\#214\\\;/g; ## "O --> &#214;
   $umlaut =~ s/\\\"U/\\\&\\\#220\\\;/g; ## "U --> &#220;
   $umlaut =~ s/\\\"s/\\\&\\\#223\\\;/g; ## "s --> &#223;

   $umlaut =~ s//\\\&\\\#228\\\;/g; ## "a --> &#228;
   $umlaut =~ s//\\\&\\\#246\\\;/g; ## "o --> &#246;
   $umlaut =~ s//\\\&\\\#252\\\;/g; ## "u --> &#252;
   $umlaut =~ s//\\\&\\\#196\\\;/g; ## "A --> &#196;
   $umlaut =~ s//\\\&\\\#214\\\;/g; ## "O --> &#214;
   $umlaut =~ s//\\\&\\\#220\\\;/g; ## "U --> &#220;
   $umlaut =~ s//\\\&\\\#223\\\;/g; ## "s --> &#223;

   return $umlaut;
   }
## end (sub substitute_umlaut)



#####################################################
## Abgekuerzte Monatsnamen durch ihr Pendant ersetzen
#####################################################
sub substitute_month
   {
   local ($mth) = @_;

   foreach $month (keys(%months))
      {
      if ($month =~ /$mth/i)
         {
         return $months{$mth};
```

```
         }
      ## end (if)
      }
   ## end (foreach)

   return $mth;
   }
## end (sub substitute_month)



######################################################
## Liefere main.ps zurueck
######################################################
sub get_psfile
   {
   local ($key) = @_;
   local ($SRCDIR, $file, @allfiles);

   ## Flag zuruecksetzen
   $NO_PSFILE = 0;

   ## Wo liegt die Dokumentation?
   $SRCDIR = "$TMP/$key/Dokumentation/Postscript";

   unless ( -d $SRCDIR  )
      {
      warn "WARNING: Can't open $SRCDIR!\n";

      ## Flag setzen
      $NO_PSFILE = 1;

      ## und zurueckkehren
      return;
      }

   opendir SRCDIR, "$SRCDIR";
      @allfiles = (@allfiles, grep /ps$/, readdir SRCDIR);
   closedir SRCDIR;

   ## Debugging
   print "Found *.ps files: @allfiles\n" if $DEBUG;

   ## Flag setzen, falls kein Postscriptfile gefunden wurde
   if (@allfiles eq "")
      {
      ## Flag setzen
      $NO_PSFILE = 1;

      ## und zurueckkehren
      return;
      }
```

```
   ## $key.ps vorhanden?
   foreach $file (@allfiles)
      {
      $file =~ /$key\.ps$/ && return $file;
      }
   ## end (foreach)

   ## main.ps vorhanden?
   foreach $file (@allfiles)
      {
      $file =~ /main\.ps$/ && return $file;
      }

   ## Nehme blindlings das erste, gefundene PS-File
   return $allfiles[0];
   }
## end (sub get_psfile)



#####################################################
## Liefere main.pdf zurueck
#####################################################
sub get_pdffile
   {
   local ($key) = @_;
   local ($SRCDIR, $file, @allfiles);

   ## Flag zuruecksetzen
   $NO_PDFFILE = 0;

   ## Wo liegt die Dokumentation?
   $SRCDIR = "$TMP/$key/Dokumentation/PDF";

   unless ( -d $SRCDIR  )
      {
      warn "WARNING: Can't open $SRCDIR!\n";

      ## Flag setzen
      $NO_PDFFILE = 1;

      ## und zurueckkehren
      return;
      }

   opendir SRCDIR, "$SRCDIR";
      @allfiles = (@allfiles, grep /pdf$/, readdir SRCDIR);
   closedir SRCDIR;

   ## Debugging
   print "Found *.pdf files: @allfiles\n" if $DEBUG;

   ## Flag setzen, falls keine PDF-Datei gefunden wurde
```

```
   if (@allfiles eq "")
      {
      ## Flag setzen
      $NO_PDFFILE = 1;

      ## und zurueckkehren
      return;
      }

   ## Mehrere *.pdf Dateien vorhanden?
   foreach $file (@allfiles)
      {
      $file =~ /$key\.pdf$/ && return $file;
      }
   ## end (foreach)

   ## Mehrere *.pdf Dateien vorhanden?
   foreach $file (@allfiles)
      {
      $file =~ /main\.pdf$/ && return $file;
      }
   ## end (foreach)

   ## Nehme blindlings die erste, gefundene PDF-Datei
   return $allfiles[0];
   }
## end (sub get_pdffile)


######################################################
## Liefere main.tex zurueck
######################################################
sub get_texfile
   {
   local ($key) = @_;
   local ($SRCDIR, $file, @allfiles);

   ## Flag zuruecksetzen
   $NO_TEXFILE = 0;

   ## Wo liegt die Dokumentation?
   $SRCDIR = "$TMP/$key/Dokumentation/Latex";

   unless ( -d $SRCDIR  )
      {
      warn "WARNING: Can't open $SRCDIR!\n";

      ## Flag setzen
      $NO_TEXFILE = 1;

      ## und zurueckkehren
      return;
```

```
      }

   opendir SRCDIR, "$SRCDIR";
      @allfiles = (@allfiles, grep /tex$/, map "$SRCDIR/$_", readdir SRCDIR);
   closedir SRCDIR;

   ## Debugging
   print "Found *.tex files: @allfiles\n" if $DEBUG;

   ## main.tex oder $key.tex vorhanden?
   ## => liefere als Filename zurueck!
   foreach $file (@allfiles)
      {
      $file =~ /main\.tex$/ && return $file;
      $file =~ /$key\.tex$/ && return $file;
      }

   foreach $file (@allfiles)
      {
      open (FILE, "$file");

         while (<FILE>)
            {
            ## LaTeX 1.09
            if (/\\documentstyle/ && /\\begin\{document\}/)
               {
               (close (FILE) && return $file);
               }
            ## LaTeX 2e
            elsif (/\\documentclass/ && /\\begin\{document\}/)
               {
               (close (FILE) && return $file);
               }
            ## Egal, welche Versionsnummer
            elsif (/\\begin\{document\}/)
               {
               (close (FILE) && return $file);
               }
            }
         ## end (while)

      close (FILE);
      }
   ## end (foreach)

   ## Kein Main TeX-File gefunden!
   print "WARNING: No main LaTeX-File found for $key in $SRCDIR!\n";

   ## Flag setzen
   $NO_TEXFILE = 1;
   }
## end (sub get_texfile)
```

```
################################################################################
## Postscriptfile nach $HTML_DATA_DIR/$type/$key/Postscript-Version
## kopieren, falls vorhanden
################################################################################
sub copy_ps_file
   {
   local ($key, $type, $psfile) = @_;
   local ($SRCDIR, $PS_DIR);

   ## Verzeichnis fuer Dokumente in Postscriptformat
   $SRCDIR = "$TMP/$key/Dokumentation/Postscript";

   ## Flag auf Null setzen:
   ## Ist $NO_PSFILE =~ 1, so gibt es keine
   ## Postscript-Version des Dokumentes
   $NO_PSFILE = 0;

   ## Destination
   $PS_DIR = "$HTML_DATA_DIR/\u\L$type\E/$key/Postscript-Version";

   # Verzeichnis loeschen, falls vorhanden
   &rm_dir ("$PS_DIR") if ( -d "$PS_DIR" );

   # Neues Verzeichnis erstellen
   &new_dir ("$PS_DIR") unless (-d "$PS_DIR" );

   ## File zippen und kopieren
   if ( -f "$SRCDIR/$psfile" )
      {
      ## Statusmeldung
      print "Zipping $psfile.\n";

      ## Komprimiere und kopiere File
      system ("gzip -9 -c $SRCDIR/$psfile > $PS_DIR/$psfile.gz");

      ## Aendere Rechte
      system ("chmod 664 $PS_DIR/$psfile.gz");

      ## Link von $key.ps -> main.ps
      system ("( cd $PS_DIR; ln -sf $psfile.gz $key.ps.gz )")
      unless ( "$psfile" eq "$key.ps" );

      ## Berechne Groesse des komprimierten Files
      $size_ps = (-s "$PS_DIR/$psfile.gz");

      ## Runde auf volle Kilobyte
      $size_ps = (( $size_ps / 1024 ) + 0.5);

      ## Debugging
      printf "\nFILE: %s.gz\nSIZE: %.1f KB\n\n", $psfile, $size_ps if $DEBUG;
      }
```

```
   else
      {
      ## Flag setzen
      $NO_PSFILE = 1;

      ## Warnen, dass kein Postscriptfile vorhanden ist
      print "WARNING: No postscript-file found for $key";
      print "in $TMP/$key/Dokumentation!\n";
      }
   ## end (if)
   }
## end (sub copy_ps_file)



################################################################################
## PDF-Datei nach $HTML_DATA_DIR/$type/$key/PDF-Version
## kopieren, falls vorhanden
################################################################################
sub copy_pdf_file
   {
   local ($key, $type, $pdffile) = @_;
   local ($SRCDIR, $PDF_DIR);

   ## Verzeichnis fuer Dokumente in PDF-Format
   $SRCDIR = "$TMP/$key/Dokumentation/PDF";

   ## Flag auf Null setzen:
   ## Ist $NO_PDFFILE =~ 1, so gibt es keine
   ## Postscript-Version des Dokumentes
   $NO_PDFFILE = 0;

   ## Destination
   $PDF_DIR = "$HTML_DATA_DIR/\u\L$type\E/$key/PDF-Version";

   # Verzeichnis loeschen, falls vorhanden
   &rm_dir ("$PDF_DIR") if ( -d "$PDF_DIR" );

   ## Neues Verzeichnis erstellen
   &new_dir ("$PDF_DIR") unless (-d "$PDF_DIR" );

   ## File kopieren
   if ( -f "$SRCDIR/$pdffile" )
      {
      ## Kopiere File
      system ("cp $SRCDIR/$pdffile $PDF_DIR");

      ## Aendere Rechte
      system ("chmod 664 $PDF_DIR/$pdffile");

      ## Link von $key.pdf -> main.pdf
      system ("( cd $PDF_DIR; ln -sf $pdffile $key.pdf )")
      unless ( "$pdffile" eq "$key.pdf" );
```

```
      ## Berechne Groesse des Files
      $size_pdf = (-s "$PDF_DIR/$pdffile");

      ## Runde auf volle Kilobyte
      $size_pdf = (( $size_pdf / 1024 ) + 0.5);

      ## Debugging
      printf "\nFILE: %s\nSIZE: %.1f KB\n\n", $pdffile, $size_pdf if $DEBUG;
      }
   else
      {
      ## Flag setzen
      $NO_PDFFILE = 1;

      ## Warnen, dass keine PDF-Datei vorhanden ist
      print "WARNING: No PDF-file found for $key in $TMP/$key/Dokumentation!\n";
      }
   ## end (if)
   }
## end (sub copy_pdf_file)


################################################################################
## Ueberpruefen, ob NOHTML gesetzt ist
################################################################################
sub no_html_file
   {
   local ($key) = @_;
   local ($SRCDIR);

   ## Flag zuruecksetzen
   $NO_HTMLFILE = 0;

   ## Wo soll das Programm nach NOHTML suchen?
   $SRCDIR = "$TMP/$key";

   ## NOHTML vorhanden?
   if ( -e "$SRCDIR/NOHTML" )
      {
      # Ja, dann Flag setzen
      $NO_HTMLFILE = 1;
      }
   ## end (if)
   }
## end (sub no_html_file)



################################################################################
## {diplomarbeiten,fopra,papers}.bib: Anpassung an deutsche Sonderzeichen
## Anschliessend: 'latex2html {diplomarbeiten,fopra,papers}.bib'
################################################################################
```

```
sub create_html_bib_file
   {
   local ($bib_file) = @_;
   local ($line, $vorspann);

   $vorspann = "\\usepackage\{german\}";
   $basename_bib_file = basename($bib_file, "");

   ## Vorspann erzeugen
   open (BIBFILE, ">$TMP/$basename_bib_file.tex") ||
   die ("\n$TMP/$basename_bib_file.tex not writeable?\n");
      print BIBFILE ("$vorspann\n");
   close (BIBFILE);

   open (BIB, $bib_file)   || die ("\nCan't open $bib_file!\n");

      while ($line = <BIB>)
         {
         ## Entferne ueberfluessige Leerzeilen
         $line =~ s/^\s*\n$//;

         ## Damit aus dem von latex2html generierten File eine korrekte
         ## Endemarkierung fuer jeden Block (<P>) eingefuegt wird, muss
         ## vor jeder Gattung (@book, @article, ...) eine Leerzeile
         ## eingefuegt werden.
         $line =~ s/\@(.*\s*)/\n\@$1/;

         ## Damit aus dem von latex2html generierten File die Gattung vom
         ## Key getrennt werden kann, muss hinter der Gattung eine oeffnende
         ## geschweifte Klammer und dann mindestens ein Leerzeichen kommen.
         $line =~ s/\@(\w+)\s*\{\s*(\w+)\s*,\s*/\@$1\{ $2,\n/;

         ## Doppelte Anfuehrungszeichen in geschweifte Klammern uebersetzen.
         ## key = "...", --> key = {...},\n
         $line =~ s/^\s*(\w+)\s*=\s*\"(.*)\"\s*,\s*$/$1 = \{$2\},\n/;

         ## key = "..."  --> key = {...}\n
         $line =~ s/^\s*(\w+)\s*=\s*\"(.*)\"\s*$/$1 = \{$2\}\n/;

         ## key = "...   --> key = {...\n
         $line =~ s/^\s*(\w+)\s*=\s*\"(.*)\s*$/$1 = \{$2\n/;

         ## ...",        --> ...},\n
         $line =~ s/^\s*(.*)\"\s*,\s*$/ $1\},\n/;

         ## ..."         --> ...}\n
         $line =~ s/^\s*(.*)\"\s*$/ $1\}\n/;

         open (NEWBIB, ">>$TMP/$basename_bib_file.tex");
            print NEWBIB "$line";
         close (NEWBIB);
         }
      ## end (while)
```

```
   close (BIB);

   ## Statusausgabe
   print "Saving $TMP/$basename_bib_file.tex...\n\n";

   -e "$LATEX2HTML_DIR/latex2html" or
   die "No such file $LATEX2HTML_DIR/latex2html.\n";

   ## latex2html mit Argumenten aufrufen:
   ##
   ##  kein weiters Subdirectory anlegen, stattdessen
   ##  in $TMP zwischenspeichern
   ##  |    kein Abschnitt: About this document ...
   ##  |    |    keine Navigationshilfen im Kopf (oder auch Fuss)
   ##  |    |    |______________________.
   ##  |    |_________________.         |
   ##  |____________.       |           |
   ##           .___|__.  .__|_.  ._____|______.
   $OPTIONS = "-dir $TMP -info 0 -no\_navigation";
   system ("$LATEX2HTML_DIR/latex2html $OPTIONS $TMP/$basename_bib_file.tex");
   }
## end (sub create_html_bib_file)


###################################################
## Benoetigte Variablen aus
## {Diplomarbeiten,Fopras}.bib File extrahieren
## und in einen assoziativen Vektor schreiben
###################################################
sub scan_html_bib_file
   {
   local ($file, $type) = @_;
   local ($work);
   local ($index) = 0;

   open (BIB, $file) || die ("\nCan't open $file!\n");

      BIB: while (<BIB>)
         {
         ## @article, @book, ... gefunden?
         if (/\@(\w+)\s+(\w+)\s*,\s*/)
            {
            $index++;      ## Index um eins erhoehen

            $kind = lc $1;    ## lowercase @article, @book, ...
            $who = $2;        ## Lehrstuhlinterner key (schm97 bspw.)

            ${$type}{$index} = $who;   ## Zuordnung von index => author
                                       ## (Durchnummerieren)
            print "INDEX: $index: ${$type}{$index}\n" if $DEBUG;
```

```
            $authmap{$who} = $kind;    ## Zuordnung von author => kind
            $bibmap{$who}  = $type;    ## Zuordnung von author => type

            print "EINTRAG GEFUNDEN: $who ist vom Typ $bibmap{$who}\n"
            if $DEBUG;

            @bib_fields = eval "\@${kind}";  ## Felder der einzelnen
                                             ## Gattungen dem Array
                                             ## @bib_fields zuordnen;
                                             ## Aufbau findet
                                             ## sich in der ini-Datei
            &make_bibhash ("$kind");         ## Erzeuge assoziativen Array
            }
         }
      ## end (while)

   close (BIB);

   ## Debugging
   if ( $DEBUG )
      {
      foreach $work (sort keys(%masterthesis))
         {
         print "$work:\n";
         foreach $field (sort keys(%{$mastersthesis{$work}}))
            {
            print "$field = $mastersthesis{$work}{$field}\n";
            }
         print "\n";
         }
      ## end (foreach)
      }
   ## end (if)
   }
## end (sub scan_html_bib_file)


######################################
## Erzeuge den assoziativen Array
######################################
sub make_bibhash
{
   local ($kind) = @_;     ## Parameteruebergabe der Gattung
   local ($who) = $who;    ## Lehrstuhlinterner key
   local ($field);         ## lokale Laufvariable

   BIBF: while (<BIB>)
      {
      foreach $field (@bib_fields)  ## beachte jedes Feld der
         {                          ## uebergebenen Gattung
         if (/^\s*$field\s*=\s*(.*),\s*$/i)  ## Einzeiliges Feld in *.bib.html
            {                                ## gefunden? (zu Erkennen am Komma
```

```
                                       ## am Ende der Zeile)
            ${$kind}{$who}{$field} = $1;     ## dann speichere Key und Value
                                       ## in %kind ab.

            ## Doppelte 'and' entfernen, d.h.
            ## aus "... and ... and ..." mache
            ## "..., ... and ..."
            while ( ${$kind}{$who}{$field} =~ /\s+and\s+(.*)\s+and\s+/ )
               {
               ${$kind}{$who}{$field} =~ s/\s+and\s+(.*)\s+and\s+/, $1 and /;
               }
            }
         elsif (/^\s*$field\s+=\s+(.*\s*)/i) ## Mehrzeiliges Feld in *.bib.html
            {                                ## gefunden? (kein Komma)
            ${$kind}{$who}{$field} = $1;     ## dann speichere Key und Value
                                             ## in %kind ab,

            FIELD:                         ## setze Sprungmarke,
               while (<BIB>)               ## durchsuche naechste Zeile
                  {
                  ## Abbruch, wenn Leerzeile gefunden wird
                  last FIELD if /^\s*$/;

                  ## sonst: haenge Zeile an,
                  ${$kind}{$who}{$field} .= $_;

                  ## Abbruch, wenn Zeile mit Komma endet.
                  last FIELD if /\,\s*$/;
                  }

            ## Make into one long line.
            ## ersetze Zeilenumbruch durch Leerzeichen
            ${$kind}{$who}{$field} =~ s/\n\s*/ /g;

            ## kuerze mehrere Leerzeichen auf eines
            ${$kind}{$who}{$field} =~ s/\s+\s*/ /g;

            ## entferne nachfolgende Leerzeichen
            ${$kind}{$who}{$field} =~ s/\s+$//;

            ## loesche Kommata am Ende einer Zeile
            ${$kind}{$who}{$field} =~ s/\,\s*$//;
            }
         ## end (if)
         }
      ## end (foreach)

      redo BIB if /\@/; ## Abbruch, wenn Bereichstrenner gefunden wird.
      }
   ## end (while)
   }
## end (sub)
```

```
#########################################
## Zusammenfassung extrahieren
#########################################
sub get_abstract
   {
   local ($html_page, $work) = @_;
   local ($abstract) = "";
   local ($file, @allfiles);

   open (INPUT, "$html_page") || die ("\nCan't open $html_page!\n");

      #
      # CLASS=ABSTRACT gefunden?
      # Dann suche nach Bereichstrenner
      # und liefere alles bis dahin gefundene
      # als $abstract zurueck
      #
      while (<INPUT>)
         {
         if (/ABSTRACT/)
            {
            while (<INPUT>)
               {
               return $abstract if m%\</P\>%;
               $abstract .= $_;
               }
            }
         ## end (if)
         }
      ## end (while)

   close (INPUT);

   return $abstract;
   }
## end (sub get_abstract)



#########################################
## Einfuehrung extrahieren
#########################################
sub get_introduction
   {
   local ($html_page, $work) = @_;
   local ($abstract, $file, @allfiles);

   open (INPUT, "$html_page")    || die ("\nCan't open $html_page!\n");

      #
      # Suche der Reihe nach alle Eintraege fuer @ZUSAMMENFASSUNG
```

```
      # durch. Zusammenfassung gefunden?
      # Dann suche nach Bereichstrenner
      # und liefere alles bis dahin gefundene
      # als $abstract zurueck.
      #

      while (<INPUT>)
         {
         foreach $ZUSAMMENFASSUNG (@ZUSAMMENFASSUNG)
            {
            if (/$ZUSAMMENFASSUNG\<\/A\>/i)
               {
               LOOP:
                  while (<INPUT>)
                     {
                     last LOOP unless (/^<\/H.>/ || /^<P>/);

                     while (<INPUT>)
                        {
                        ## Wonach gesucht wird, bestimmt die
                        ## Variable $CHILDLINE = "<BR><HR>\n";
                        ## in latex2html.config
                        return $abstract if /$CHILDLINE/;
                        $abstract .= $_;
                        }
                     ## end (while)
                     }
                  ## end (while)
               }
            ## end (if)
            }
         ## end (foreach)
         }
      ## end (while)
   close (INPUT);

   return $abstract;
   }
## end (sub get_introduction)



#################################################
## Mehrzahl des angegebenen Typs zurueckliefern
#################################################
sub get_type
   {
   local ($type) = @_;
   local ($key);

   foreach $key (sort keys(%types))
      {
      if ($type =~ /$key/i)
```

```
         {
         return $types{$key};
         }
      ## end (if)
      }
   ## end (foreach)

   return "Others";
   }
## end (sub type)



#######################################
## Inhaltsangabe erzeugen
#######################################
sub print_inhalt
   {
   local ($type, $key) = @_;     ## Uebergebene Gattung
   local ($work);

   ## Debugging
   if $DEBUG
      {
      print "Inhaltsuebersicht; HTML Seite: ${type}.shtml; ";
      print "KIND: $kind; TYPE: $type\n;";
      }
   ## end (if)

   ## Gewaehrleiste Schreibrechte fuer
   ## die entsprechende Gruppe;
   ## entpricht 775 fuer Verzeichnisse und
   ## 664 fuer Dateien
   umask 002;

   ## publikationen.shtml erstellen
   open (NAME, ">$HTML_DATA_DIR/${type}.shtml") ||
   die ("\nCan't create $HTML_DATA_DIR/${type}.shtml!\n");

   ## Kopf
   &print_header_inhalt ($type);

   ## Body
   ## Sortiere alle Indexe der Nummer nach aufsteigend
   foreach $ind ( sort { $a <=> $b } keys (%{$type}))
      {
      ## Debugging
      print "$ind: ${$type}{$ind}\n" if $DEBUG;

      ## Zuordnung: index -> key
      $work = ${$type}{$ind};

      ## Debugging
```

```
      print "SUCCESS: $work -> $authmap{$work} -> $bibmap{$work}\n"
      if $DEBUG;

      if ( -e "$HTML_DATA_DIR/\u\L$type\E/$work/$work.shtml" )
         {
         ## Debugging
         print "$HTML_DATA_DIR/\u\L$type\E/$work/$work.shtml VORHANDEN\n"
         if $DEBUG;

         &print_body_inhalt ($type, $authmap{$work}, $work, "LINK");
         }
      else
         {
         ## Debugging
         print "$HTML_DATA_DIR/\u\L$type\E/$work/$work.shtml nicht vorhanden\n"
         if $DEBUG;

         &print_body_inhalt ($type, $authmap{$work}, $work, "NOLINK");
         }
      ## end (if)
      }
   ## end (foreach)

   ## Fuss
   &print_footer_inhalt;

   close (NAME);
   }
## end (sub print_inhalt)



#########################################
## HTML-Header erzeugen
#########################################
sub print_header
   {
   local ($type, $kind, $work, $output) = @_;

   ## Ueberschrift fuer Kopfleiste aus %type holen
   $type = &get_type($type);

   open (NAME , ">$output") || die ("\nCan't create $output!\n");

      print NAME ("<HTML>\n");
      print NAME ("<HEAD>\n");

      ## Kopfleiste
      print NAME ("<TITLE>$type, $work</TITLE>\n");
      print NAME ("</HEAD>\n");

   close (NAME);
   }
```

```
## end sub (print_header)



########################################
## HTML-Body erzeugen
########################################
sub print_body
   {
   local ($type, $kind, $key, $abstract, $output, $htmlfile) = @_;

   open (NAME, ">>$output") || die ("\nCan't open $output!\n");

      ## Anfang Body
      print NAME ("<BODY>\n");
      print NAME ("<!--\#include virtual=\"/includes/header.shtml\" -->\n");

      ## Autor (Erscheinungsjahr)
      print NAME ("<H2>${$kind}{$key}{author} (${$kind}{$key}{year}):</H2>\n");

      ## Titel
      print NAME ("<H1>${$kind}{$key}{title}</H1>\n");
      print NAME ("<P>\n");

      ## Zusammenfassung
      print NAME ("  $abstract\n");
      print NAME ("</P>\n");
      print NAME ("<HR>\n");
      print NAME ("<UL>\n");

      ## Link zu Postscript-Version
      if ($NO_PSFILE =~ 0)
         {
         print NAME ("  <LI><A HREF = \"Postscript-Version/$key.ps.gz\">");
         print NAME ("GNU-zipped Postscript Version</A>\n");
         print NAME ("    <FONT SIZE=-1>\n");
         printf NAME "      \(%.1f KB\)\n", $size_ps;
         print NAME ("    </FONT>\n");
         }
      ## end (if)

      ## Link zu PDF-Version
      if ($NO_PDFFILE =~ 0)
         {
         print NAME ("  <LI><A HREF = \"PDF-Version/$key.pdf\">");
         print NAME ("PDF Version</A>\n");
         print NAME ("    <FONT SIZE=-1>\n");
         printf NAME "      \(%.1f KB\)\n", $size_pdf;
         print NAME ("    </FONT>\n");
         }
      ## end (if)

      ## Link zu HTML-Version
```

```
      if ($NO_TEXFILE =~ 0 && $NO_HTMLFILE =~ 0)
         {
         print NAME ("  <LI><A HREF = \"HTML-Version/$htmlfile\">");
         print NAME ("HTML Version</A>\n");
         }
      ## end (if)

      print NAME ("</UL>\n");
      print NAME ("</P>\n");

   close (NAME);
   }
## end sub (print_body)



##########################################
## HTML-Footer erzeugen
##########################################
sub print_footer
   {
   local ($output) = @_;

   open (NAME, ">>$output") || die ("\nCan't open $output!\n");

      print NAME ("<P>\n");
      print NAME ("<HR>\n");
      print NAME ("<FONT SIZE=-1>\n");
      print NAME ("<ADDRESS>\n");

      ($sec, $min, $hour, $tag, $monat, $jahr, @rest) = localtime (time);
      $monat++;

      print NAME ("Created by MNMlit2html, $tag.$monat.$jahr.<BR>\n");
      print NAME ("<!--\#include virtual=\"/includes/trailer.shtml\" -->\n");
      print NAME ("</FONT>\n");
      print NAME ("</BODY>\n");
      print NAME ("</HTML>\n");

   close (NAME);
   }
## end (sub print_footer)



################################################
## HTML-Header erzeugen -- Inhaltsverzeichnis
################################################
sub print_header_inhalt
   {
   ## Ueberschrift aus %type holen
   local ($type) = &get_type(@_);
```

```
   ## Seitenanfang
   print NAME ("<HTML>\n");

   ## Kopf
   print NAME ("<HEAD>\n");

   ## Kopfleiste
   print NAME ("<TITLE>$type</TITLE>\n");

   ## Kopf -- Ende
   print NAME ("</HEAD>\n");

   ## Angang Body
   print NAME ("<BODY>\n");

   ## Lehrstuhl Kopf
   print NAME ("<!--\#include virtual=\"/includes/header.shtml\" -->\n");

   ## Absatz -- Anfang
   print NAME ("<P>\n");

   ## Titel
   print NAME ("<H1>$type</H1>\n");

   ## Absatz -- Ende
   print NAME ("</P>\n");

   ## Horizontale Linie
   print NAME ("<HR>\n");

   ## Absatz -- Anfang
   print NAME ("<P>\n");
   }
## end (sub print_header_inhalt)



################################################
## HTML-Body erzeugen -- Inhaltsverzeichnis
################################################
sub print_body_inhalt
   {
   local ($type, $kind, $key, $link) = @_;
   local (@to_publish) = eval "\@${kind}_to_publish";
   local ($flag) = 0;

   @to_publish = eval "\@${kind}" if !(@to_publish);

   ## Unnumerierte Liste
   ## <UL type=[circle/square/disk]> ist
   ## Bestandteil von HTML 3.2
   if ($link =~ "^LINK")
      {
```

```
   print NAME ("<UL type=disk>\n");
   }
else
   {
   print NAME ("<UL type=circle>\n");
   }
## end (if)

## Listeneintrag
print NAME ("<LI>\n");

foreach $entrie (@to_publish)
   {
   ## Monatsnamen ausschreiben
   if ("$entrie" =~ /^month$/i)
      {
      ${$kind}{$key}{$entrie} = &substitute_month (${$kind}{$key}{$entrie});
      }
   ## end (if)

   if ($flag == 1 && "${$kind}{$key}{$entrie}" ne "")
      {
         print NAME (", ");
         $flag = 0;
      }
   ## end (if)

   ## Listeneintrag der Form <a name = hkn96>
   if ($entrie =~ /^title/)
      {
      print NAME ("<A NAME = $key>\n");
      }

   ## Listeneintrag mit Link auf die betreffende Arbeit
   if ($entrie =~ /^title/ and $link =~ "^LINK")
      {
      print NAME ("<A HREF = \"\u\L$type\E/$key/$key.shtml\">");
      }

   ## Trage den Titel ein, sofern vorhanden
   if ("${$kind}{$key}{$entrie}" ne "")
      {
      print NAME ("${$kind}{$key}{$entrie}");

      if ($entrie =~ /^title/ and $link =~ "^LINK")
         {
         ## Listeneintrag abgeschlossen durch ',<\A>'
         print NAME (",</A> ");

         $flag = 0;
         }
      elsif ($entrie =~ /^title/ and $link =~ "NOLINK")
         {
```

```
            ## Listeneintrag, aber kein Link
            ## => Titel mit ',' statt '<\A>' abschliessen
            print NAME (", ");

            $flag = 0;
            }
         else
            {
            $flag = 1;
            }
         ## end (if)
         }
      ## end (if)
      }
   ## end (foreach)

   ## Ende der unnumerierten Liste
   print NAME ("\n</UL>\n");
   }
## end (sub print_body_inhalt)



#################################################
## HTML-Footer erzeugen -- Inhaltsverzeichnis
#################################################
sub print_footer_inhalt
   {
   ($sec, $min, $hour, $tag, $monat, $jahr, @rest) = localtime (time);
   $monat++;

   ## Absatz -- Ende
   print NAME ("</P>\n");

   ## Absatz -- Anfang
   print NAME ("<P>\n");

   ## Horizontale Linie
   print NAME ("<HR>\n");

   ## Kleinerer Font
   print NAME ("<FONT SIZE=-1>\n");

   ## Zusatzangaben
   print NAME ("<ADDRESS>\n");

   ## Informationen zur Seite
   print NAME ("Created by MNMlit2html, $tag.$monat.$jahr.\n");

   ## Zeilenumbruch
   print NAME ("<BR>\n");

   ## Lehrstuhlfuss
```

```
   print NAME ("<!--\#include virtual=\"/includes/trailer.shtml\" -->\n");

   ## Kleinerer Font -- Ende
   print NAME ("</FONT>\n");

   ## Absatz -- Ende
   print NAME ("</P>\n");

   ## Body -- Ende
   print NAME ("</BODY>\n");

   ## Seitenende
   print NAME ("</HTML>\n");
   }
## end (sub print_footer_inhalt)

# ########################### end (subroutines) #################################
```

Anhang C

Listing zum Wrapper mnmlit2html

```
#!/bin/sh
#
# Wrapper fuer mnmlit2html.pl
# Stellt sicher, dass Software nur an vorgegeben
# Orten gefunden wird.
#
#set -x

# Wichtig: Suche nur hier nach Software
PATH=/usr/local/dist/bin:/usr/local/bin:/usr/local/bin/X11:\
/usr/bin/X11:/usr/openwin/bin:/bin:/usr/bin:/usr/ucb:/usr/sbin:\
/sbin:/usr/ccs/bin:/usr/bin/X11:.

# Setze Hilfspfade
MNMBINDIR=/usr/local/mnmcommon/bin

# Mailprogramm und -user
MAILCMD=/usr/bin/mailx
MAILUSER=$LOGNAME
MAIL="${MAILCMD} -s \"Logfile of mnmlit2html.pl\" ${MAILUSER}"

# Software vorhanden?
test -x ${MNMBINDIR}/mnmlit2html.pl || exit 0
test -x ${MAILCMD} || exit 0

# Starte mnmlit2html.pl
${MNMBINDIR}/mnmlit2html.pl $* 2>&1 | eval ${MAIL}
```

Anhang D

Listing zu makebib

```
#!/usr/local/dist/bin/perl
#
# Dieses Programm sucht alle Literaturverweise in saemtlichen
# LaTeX Dateien des aktuellen Verzeichnisses. Anhand der
# Environment-Variable $BIBINPUTS findet es die dazugehoerigen
# *.bib Dateien und erzeugt eine neue Bibliotheksdatei mit Namen
# main.bib im aktuellen Verzeichnis. Falls hier eine solche Datei
# schon existiert, so kann man den Namen der neuen Bibliotheksdatei
# dem Programm auch als Switch mitgeben.


## Argumente bearbeiten
## Aktuelle Kommandozeilenparameter in $argv ablegen
$argv = join(' ',@ARGV);

## Versionsnummer, Patchlevel und Erscheinungsdatum
$PATCHLEVEL = "1a";
$VERSION = "97";
$RELDATE = "(Sep 29, 1997)";

$MakebibV_SHORT = $VERSION . '.' . "$PATCHLEVEL";
$MakebibVERSION = $MakebibV_SHORT . ' ' . $RELDATE;

## Hinweise zum Author
$AUTHORADDRESS =
   "http://www.informatik.tu-muenchen.de/~allgeyer/index.html";

## Name der neuen *.bib Datei
$newbibfile = "main.bib";

## Parameter aufloesen
while ($ARGV[0] =~ /^-/)
   {
   $_ = shift;
   ## Debugging gewuenscht
   if (/^-debug$/)
      {
      $DEBUG = 1;
```

```
      }
   ## -newbibfile (file) wurde spezifiziert:
   elsif (/^-newbibfile$/)
      {
      $_ = shift;
      ((($newbibfile) = /^(.+)$/) ||
      print "\nUnrecognised value for -newbibfile: $_\n" &&
      &usage && die);
      }
   ## -h(elp) wurde eingegeben:
   elsif (/^-h(elp)?$/)
      {
      &usage;
      exit 0;
      }
   ## Kein bekannter Parameter?
   else
      {
      &usage;
      die "Unrecognised switch: $_\n";
      }
   ## end (if)
   }
## end (while)

## Hole $HOME aus der Environment Variable
## und erweitere @INC damit
push(@INC,$ENV{'HOME'});
$HOME = $ENV{'HOME'};

## Statusausgabe
print "Running \'makebib.pl $argv\': ..";

## Hole $BIBINPUTS aus der Environment Variable
if ( $ENV{'BIBINPUTS'} eq '' )
   {
   $BIBINPUTS = '.';
   }
else
   {
   $BIBINPUTS = $ENV{'BIBINPUTS'};
   }
## end (if)

## Aktuelles Verzeichnis nach LaTeX Dateien durchsuchen
@texfiles = glob ("*.tex");

foreach $texfile (@texfiles)
   {
   ## Debugging:
   print "\nTexfile found: $texfile\n" if $DEBUG;

   &search_cite ($texfile);
```

```
   &search_bibliography ($texfile);
   }

## Suchpfad aus $BIBINPUTS setzen
@searchpath = split /:/, $BIBINPUTS;

## In jedem Verzeichnis aus $BIBINPUTS
foreach $dir (@searchpath)
   {
   opendir (THISDIR, $dir);

      ## extrahiere alle *.bib Dateien
      @allbibfiles = grep /.*\.bib/, map "$dir/$_", readdir THISDIR;

      ## und suche diejenigen heraus, auf die im
      ## vorliegenden Dokument Bezug genommen wird.
      foreach $file (@bibliographylist)
         {
         @bibfiles = (@bibfiles, grep /${file}\.bib$/, @allbibfiles);
         }
      ## end (foreach)

   closedir (THISDIR);
   }
## end (foreach)

## Debugging:
print "\nBibfiles: @bibfiles\n" if $DEBUG;

## Neuen Bibfile-Hash erstellen
foreach $bibfile (@bibfiles)
   {
   &make_bibfile_hash ($bibfile);
   }

# Statusausgabe
print "\n\n$index entries of bibliography found in all LaTeX-files.\n";
print "Can assign them $index2 entries ";
print "of bibliography found in all the BibTeX-files.\n";

if ( $index2 > $index )
   {
   print "\nMaybe there are double entries in the BibTeX-files!\n";
   print "So I'll only put $index entries of bibliography into";
   print "$newbibfile!\n";
   }
## end (if)

## Neues Bibfile ausgeben
&make_new_bibfile;

## Statusmeldung
print "\nDone.\n";
```

```
##########################################
##   Unterprogramme                     ##
##########################################
sub make_bibfile_hash
   {
   ## Statusausgabe
   print "..";

   local($bibfile) = @_;
   local($cite);

   open (BIBFILE, $bibfile) || die ("Kann $bibfile nicht lesen!\n");
      LINE: while (<BIBFILE>)
         {
         if (/\@\w+\s*\{\s*(\S*)\s*,\s*/)   ## @MasterThesis, @Book ...
            {                               ## gefunden!
            CITE: foreach $cite (@citelist) ## Durchlaufe alle Lit.-
               {                            ## Verweise und schaue, ob
               next CITE if ($cite ne $1);  ## die gesuchte dabei ist.

               $newcitelist{$cite} = $_;  ## Gesuchter Literaturverweis
               $index2++;                 ## vorhanden! Speichere ihn
                                          ## erhoehe Index2.
               while (<BIBFILE>)          ## Fuege alle weiteren Zeilen
                  {                       ## bis zur naechsten Typangabe
                  if (/\@/)               ## (bspw. @Book, ...) hinzu
                     {
                     ## Debugging
                     print "\nCite_elem: $newcitelist{$cite}\n" if $DEBUG;

                     redo LINE;           ## und gehe zurueck.
                     }
                  else
                     {
                     s/%.*//;             ## Sonst entferne Kommentare
                     s/^\s*\n$//;         ## und Leerzeichen und
                                          ## fuege aktuelle Zeile an.
                     $newcitelist{$cite} .= $_;
                     }
                  ## end (if)
                  }
               ## end (while)
               }
            ## end (foreach)
            }
         ## end (if)
         }
      ## end (while)

   close (BIBFILE);
   }
```

```
## end (sub make_bibfile_hash)


sub search_cite
   {
   ## Statusausgabe
   print "..";

   local($texfile) = @_;

   open (TEXFILE, $texfile)   || die ("Can't read $texfile!\n");
      LINE: while (<TEXFILE>)
         {
         while (m/\\cite\{(\S*)\}/g)            ## Suche nach \cite{(...)}
            {
            @tempcitelist = (split /,/, $1);    ## Erzeuge Referenzliste

            foreach $tempcite (@tempcitelist)   ## Fuer jede Referenz
               {                                ## in der Referenzliste:
               foreach $cite (@citelist)        ## Durchlaufe jedes
                  {                             ## Listenelement,
                  next LINE if ($cite eq $tempcite);  ## ueberpruefe, ob
                  }                             ## Listeneintrag $tempcite
               ## end (foreach)                 ## schon in der Liste
                                                ## @citelist vorhanden ist

               @citelist = (@citelist, $tempcite);    ## Wenn nicht, dann
               $index++;                              ## erhoehe Index und
               }                                      ## nehme $tempcite in
            ## end (foreach)                          ## @citelist auf.
            }
         ## end (if)
         }
      ## end (while)

   close (TEXFILE);

   ## Debugging
   print "Citelist: @citelist\n\n" if $DEBUG;
   }
## end (sub search_cite)



sub make_new_bibfile
   {
   ## Abfrage, wenn main.bib schon vorhanden
   print "\n$newbibfile still exists. Continue (y/n)? " and
   &yes_no if -e "$newbibfile";

   ## Leerzeile
   print "\n";
```

```
   ## Ueberschreiben? Ja, dann lege ein neues File an.
   open (NEWBIBLIST, ">$newbibfile");
   close (NEWBIBLIST);

   ## Setze $index zurueck
   $index = 0;

   ## Statusausgabe
   print "Following entries will be added:\n";
   print "--------------------------------\n";

   ## Sortierte Ausgabe aller BibTeX-Eintraege
   ## Fuer jeden Wert im Hash %newcitelist
   foreach $newcite (sort keys(%newcitelist))
      {
      ## Erhoehe index
      $index++;

      ## Statusausgabe
      printf "%4d) %s\n", $index, $newcite;

      ## Neue main.bib Datei erstellen
      open (NEWBIBLIST, ">>$newbibfile");
         print NEWBIBLIST ("$newcitelist{$newcite}\n");
      close (NEWBIBLIST);
      }
   }
## end (sub make_new_bibfile)


sub search_bibliography
   {
   ## Statusausgabe
   print "..";

   local($texfile) = @_;

   open (TEXFILE, $texfile) ||
   die ("Kann $texfile nicht lesen!\n");

   LINE: while (<TEXFILE>)
      {
      if (/\\bibliography\{([\w\s,]*)\}/g) ## Suche nach
         {                                 ## \bibliography{(...)}
         ## Gefundenen String splitten und
         ## der Bibliographieliste anhaengen.
         @bibliographylist = (@bibliographylist, split /,/, $1);

         ## Debugging
         print "\nBibliographylist: @bibliographylist\n" if $DEBUG;
         }
      ## end (if)
      }
```

```
   ## end (while)

   close (TEXFILE);
   }
## end (sub search_bibliography)


## Abfrage nach Ja oder Nein
sub yes_no
   {
   ## Akzepiere eingegebene Zeichenkette nur,
   ## wenn 'Y' oder 'y' eingegeben wurde.
   $ans = <stdin>;
   return if $ans =~ /^[Yy]$/;

   ## Falsche Taste erwischt oder Unsinn eingegeben?
   if ($ans !~ /^[Nn]$/)
      {
      print "Please type Y,y or N,n! (y,n)? " and &yes_no;
      }
   else
      {
      ## Statusausgabe
      print "\nQuit!\n";

      exit 1;
      }
   ## end (if)
   }
## end (sub yes_no)


## Eingabe Fehler
sub usage
   {
   print <<EOF;
This is Makebib Version $MakebibVERSION by Peter Allgeyer,
Institut fuer Informatik, Technical University of Munich.

Usage: makebib.pl
   [-newbibfile <file>]
   [-debug]
   [-h(elp)]

EOF
   }
## end (sub usage)
```