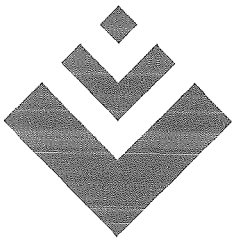
Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte
FZKA 5863



UIS

Baden-Württemberg

Projekt GLOBUS
Vorstudie zur Konzeption und prototypischen Realisierung einer aktiven Auskunftskomponente für globale Umweltsachdaten
Phase I 1994

R. Mayer-Föll, A. Jaeschke (Hrsg.)

Ministerium für Umwelt und Verkehr Baden-Württemberg

Forschungszentrum Karlsruhe
Institut für Angewandte Informatik

November 1994

Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte
FZKA 5863

Projekt GLOBUS

Vorstudie zur Konzeption und prototypischen Realisierung einer aktiven Auskunftskomponente für globale Umweltsachdaten Phase I 1994

R. Mayer-Föll, A. Jaeschke (Hrsg.)

Ministerium für Umwelt und Verkehr Baden-Württemberg

Forschungszentrum Karlsruhe
Institut für Angewandte Informatik

Forschungszentrum Karlsruhe GmbH, Karlsruhe
1994

Als Manuskript verbreitet im November 1994

Nachdruck als Wissenschaftlicher Bericht FZKA 5863 im Januar 1997

Druck und Vertrieb

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

ISSN 0947-8620

Projekt GLOBUS

Vorstudie zur Konzeption und prototypischen Realisierung einer aktiven Auskunftskomponente für globale Umweltsachdaten

Phase I - 1994

Projektträger:

Umweltministerium Baden-Württemberg;
I. Henning, R. Mayer-Föll

Landesanstalt für Umweltschutz Baden-Württemberg (LfU);
M. Müller, H. Spandl

Projektpartner:

Institut für Photogrammetrie und Fernerkundung
der Universität Karlsruhe (IPF);
J. Wiesel

Institut für Kernenergetik und Energiesysteme
der Universität Stuttgart (IKE);
F. Schmidt

Forschungsinstitut für anwendungsorientierte Wissensverarbeitung
an der Universität Ulm (FAW);
W. F. Riekert

Forschungszentrum Informatik
an der Universität Karlsruhe (FZI);
R. Kramer

Vorwort

Das Umweltministerium Baden-Württemberg baut gemeinsam mit anderen Ministerien und der Stabsstelle Verwaltungsstruktur, Information und Kommunikation ein ressortübergreifendes Umweltinformationssystem (UIS) auf. Dieses UIS ist Teil des Landessystemkonzepts Baden-Württemberg. Das UIS ist der Rahmen für die Bereitstellung von Umweltdaten und die Bearbeitung von fachbezogenen und fachübergreifenden Aufgaben im Umweltbereich. Die große Bedeutung der Informations- und Kommunikationstechnik (IuK) für eine wirkungsvolle Unterstützung von Umweltaufgaben wurde bereits 1984 erkannt. Art, Vielfalt, Menge und Verteilung der Daten erfordern den Einsatz von Informations- und Kommunikationstechnik (IuK) für den Umweltschutz. Große Teile der benötigten Software mußten bisher individuell entwickelt werden. Dies hatte leistungsfähige, aber proprietäre Systeme mit erheblichem Wartungs- und Pflegeaufwand zur Folge.

Das Umweltministerium und die Landesanstalt für Umweltschutz (LfU) führten mit den Instituten

- Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW) an der Universität Ulm,
- Forschungszentrum Informatik (FZI) an der Universität Karlsruhe,
- Institut für Kernenergetik und Energiesysteme (IKE) der Universität Stuttgart,
- Institut für Photogrammetrie und Fernerkundung (IPF) der Universität Karlsruhe

bereits mehrere UIS-Projekte erfolgreich durch.

Mit dem Forschungsprojekt GLOBUS - Vorstudie zur Konzeption und prototypischen Realisierung einer aktiven Auskunftskomponente für globale Umweltsachdaten im Umweltinformationssystem Baden-Württemberg, Phase I 1994 - wurden diese 4 Institute als Konsortium beauftragt. Die Federführung und die Projektleitung übernahm das IPF.

In GLOBUS I richtete sich das Hauptaugenmerk auf die Nutzbarmachung der Internet-Architektur und -Technik sowie seiner frei verfügbaren Werkzeuge mit dem Ziel der Qualitätsverbesserung und Zeiteinsparung bei der Informationssuche. Bei der Heterogenität der Informationsbestände galt dabei die besondere Aufmerksamkeit den Metadaten („Information über die Sachdaten") und deren Verwaltung. Weiter sollte untersucht werden, ob vorhandene monolithische Systeme durch modular verfügbare Dienste mit derselben Leistungsfähigkeit abgelöst werden können.

Wir möchten an dieser Stelle den am Projekt beteiligten Instituten für ihre ideenreiche Arbeit danken. Mit der erstellten Studie wurden die Vorteile des Einsatzes von Internet-Architektur für die Fortscheibung der UIS-Rahmenkonzeption und die Machbarkeit von Neuentwicklungen mit diesen Techniken exemplarisch aufgezeigt.

Inge Henning, Roland Mayer-Föll, Umweltministerium Baden-Württemberg
Manfred Müller, Horst Spandl, Landesanstalt für Umweltschutz

Zusammenfassung

Anhand von Dokumenten verschiedenster Herkunft über Umweltsachdaten wurde gezeigt, daß der Einsatz von World Wide Web (WWW) Werkzeugen, wie Mosaic, httpd und Imagemap leicht zu bedienende und einheitliche Nutzerschnittstellen zu erstellen erlaubt. Da die meisten Komponenten lizenzkostenfrei sind, bleiben auch die Folgekosten bei breitem Einsatz niedrig.

Im Projekt wurde auf den unterschiedlichsten Rechnerplattformen (VMS, Ultrix, Sunos, MS-Windows, Mac-OS) gezeigt, daß die WWW-Architektur herstellerübergreifend auch auf preiswerten Arbeitsplatzrechnern anwendbar ist.

Insbesondere ist auch ein Zugang zu relationalen Datenbanken und GIS möglich, ohne kostenpflichtige Softwarekomponenten in der Fläche installieren zu müssen.

Beispielhaft wurden vernetzte Anwendungen unter Nutzung LVN (Umweltberichte, UDK, Kartenzugriff) auf der Basis von WWW (Mosaic, httpd) sowohl im UM als auch in der LfU implementiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Ziel des Projektes GLOBUS ist es, vorrangig für Referenten im Umweltministerium und in der Landesanstalt für Umweltschutz eine benutzerfreundliche, systemunabhängige aktive Auskunftskomponente zu entwickeln. Diese soll unterschiedliche Datenbestände auch über den Rahmen des Umweltinformationssystems hinaus erschließen, ohne daß sie neu strukturiert und gepflegt werden müssen. Die Anwendung soll von den bestehenden Arbeitsplätzen (PCs, X- Terminals etc.) aus nutzbar sein, durch die Integration bestehender, frei verfügbarer Software soll keine Neuentwicklung erforderlich werden. Auch im Hinblick auf die zunehmenden Auskunftspflichten im Umweltbereich (s. Gesetz vom 08. Juli 1994 zur Umsetzung der Richtlinie 90/313 EWG des Rates vom 07. Juni 1990 über den freien Zugang zu Informationen über die Umwelt) ist es erforderlich, Daten und Informationen über bestehende Rechner- und Datenbanksysteme hinweg zugänglich zu machen. Die Verbreitung von Personal-Computern bietet zwar technisch ständig zunehmende Leistungsverbesserungen; die ebenfalls entsprechend anwachsenden Informationsbestände in den unterschiedlichsten Datenbanken können aufgrund unterschiedlicher Zugangswege und meist unbekannter Datenstrukturen häufig garnicht oder nur mit hohem Aufwand vom Anwender erschlossen werden.

Der wirtschaftliche Nutzen des Projektes besteht somit in einer wesentlichen Qualitätsverbesserung bei der Informationssuche sowie in der Einsparung von Arbeitszeit beim Referenten. Außerdem werden durch das bereitgestellte Produkt Softwareentwicklungs- und Lizenzkosten eingespart. Auf dem neuesten Stand der IuK-Technik soll über den Einsatz von sog. "intelligenten Agenten" beim Zugriff auf semantisch heterogene Informationsquellen die Lösung zahlreicher Probleme erfolgen (z. B. automatische Reaktion auf temporär nicht verfügbare Teilsysteme, Zugriff auf mehrere Datenquellen, Verknüpfung und Aggregation der Teilergebnisse zum Endergebnis, übergreifende Suche in Datenkatalogen, Verbund unterschiedlichster Dokumente). Bei solchen "Agenten" handelt es sich im Prinzip um Software-Stücke, die Benutzeraufträge beliebiger Komplexität mit einem wesentlich höheren Abstraktionsniveau als bei den gängigen Standardabfragesprachen ausführen.

Kapitel 2

Eine kurze Einführung in WorldWideWeb (WWW)

2.1 Was ist World Wide Web?

World Wide Web (auch WWW oder W3 genannt) ist ein Projekt, das auf Initiative einer Gruppe von Physikern und Informatikern am CERN in Genf entstand. Es hatte ursprünglich zum Ziel, ein interaktives, weltumspannendes Informationssystem fuer die Hochenergiephysik zu schaffen. Das Grundprinzip war und ist, daß *Hypermediadokumente* ortsunabhängig durchstöbert (engl.: to browse) werden können. Weiteres wichtiges Ziel war, daß die sehr unterschiedlichen Dienste im Internet, d.h. NetNews, Gopher, FTP, WAIS und Telnet, unter einer einheitlichen Bedieneroberfläche und Adressierung genutzt werden sollten.

W3 basiert auf einer klassischen Client/Server Architektur: Mittels eines Klienten *browsed* ein Nutzer durch die von den Servern vorgehaltenen Dokumente oder nutzt andere angebotene Dienste.

Voraussetzung für WWW sind die Internetprotokolle TCP/IP, aber *nicht* ein Anschluß an das Internet. WWW kann auch in hausinternen Netzen oder auch auf nur einem einzigen Rechner eingesetzt werden.

Zur Kommunikation zwischen Server und Client wird das HTTP (*Hyper-Text Transfer Protocol*) eingesetzt. WWW-Server (HTTP-Daemonen) existieren für praktisch alle gängigen Betriebssysteme (alle Unix-Varianten, DEC-VMS, Windows-NT, MS-Windows 3.1, VM/CMS, MVS). Für Dienste, die ein HTTPD nicht selbst erledigen kann bedient er sich sogenannter Gateways. über ein *Common Gateway Interface* (CGI) kommuniziert der HTTPD mit den Dienstanbietern hinter den Gateways.

Für den Zugang zu geographischen daten besonders gut geeignet ist ein Mechanismus mit dem Namen *Imagemap*. Ein kooperierender Client (z.B. Mosaic) überträgt Maus-Clicks und Koordinaten an einen HTTPD. Dieser liest die Koordinaten aus, übergibt sie inklusive Link an eine Imagemap-Anwendung, die entsprechend mit Texten, Formularen oder Bildern antwor-

tet. Nachfolgend ist eine Anwendung von Imagmap wiedergegeben, die als Uni- bzw. Gebäudeplan mit Personen-/Zimmerzuordnung dient:

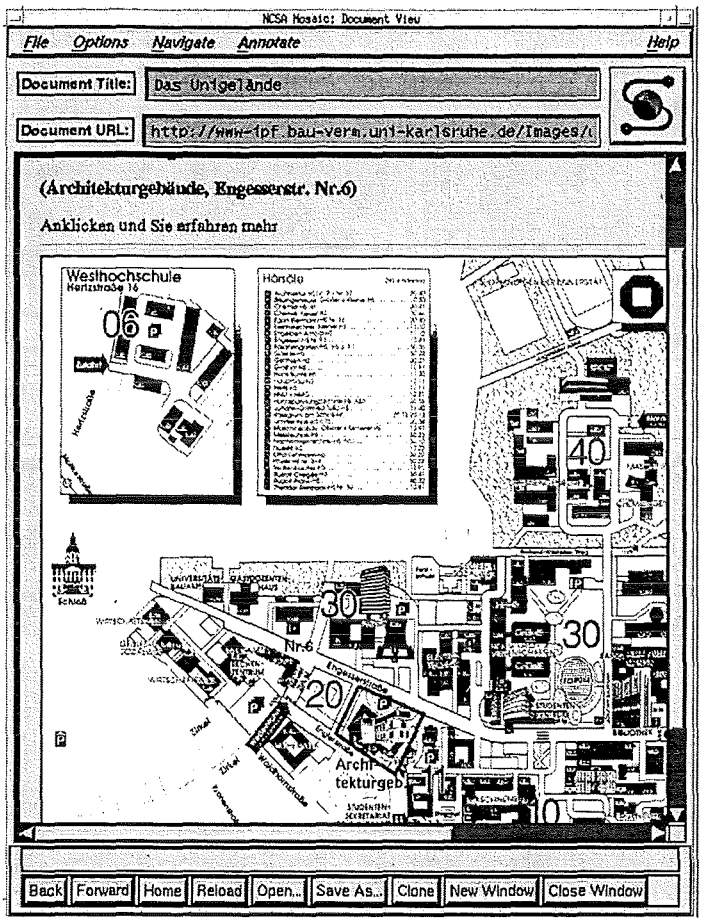


Abbildung 2.1: GLOBUS/WWW: Uniplan im WWW

Auf der Client-Seite (das ist die Benutzerschnittstelle) gibt es derzeit mehrere Alternativen. Am bekanntesten ist der am National Center for Supercomputing Applications (NCSA) in Urbana, Illinois entwickelte *Mosaic* Browser. Er ist für Unix/X, VMS/X, MS-Windows und MAC-OS verfügbar und kostenlos bei nichtkommerziellem Einsatz (siehe Abb. 2.3).

Kommerzielle Varianten von Mosaic findet man in den Angeboten von TCP/IP Software für Windows (z.B. SPRY Air fow Windows).

Ein neues kommerzielles Paket ist *NetScape* von Mosaic Communications (siehe Abb. 2.4), das ebenfalls auf allen gängigen Plattformen (Unix/X, MS-Windows, Mac-OS) verfügbar ist.

Mosaic und Netscape sind grafisch orientierte Anwendungen, die über Menüs und Maus bedient werden. Sie stellen Texte und Grafiken im GIF selbst dar. Beide beherrschen auch FTP, Telnet und Gopher-Protokolle. Für Dokumentarten, die nicht direkt verstanden werden, bedienen sie sich sogenannter *Helper Applications*, externer Programme, die z.B. Film- oder Audiosequenzen wiedergeben oder etwa PostScriptdateien anzeigen können. Gängi-

Abbildung 2.2: GLOBUS/WWW: Gebäudeplan im WWW

ge Programme (alle kostenfrei erhältlich) sind in der Unix-Welt z.B. XV, xli und ImageMagick für Bilddarstellungen, mpegplay für Video, showaudio für Audiodateien , in der MS-Windows-Welt LVIEW für die Bilddarstellung, MPLAY und Media Player für Videos und Audio.

Steht kein grafischer Arbeitsplatz zur Verfügung, so kann man *Lynx* benutzen, ein zeichenorientierter Client, der auf allen Textbildschirmen mit gutem Bedienungskomfort verwendbar ist (Curses-Applikation). In Abb. 2.5 ist die gleiche *Home Page* wie in Abb. 2.3 dargestellt, um einen Eindruck von den Gestaltungsmöglichkeiten einer reinen Textbedienoberfläche zu geben.

Da Multimediadokumente besonders hohe Anforderungen an die Übertragungsbandbreite von Datennetzen stellen, haben die WWW-Entwickler einen Caching-Mechanismus vorgesehen. Ein WWW-Server diesseits einer langsamen (oder teuren) Kommunikationsverbindung speichert die bereits schon einmal von irgendeinem Nutzer geholten Dokumente zwischen und stellt sie bei Anforderung lokal bereit (Proxy-Mechanismus, Abb. 2.6). Cache-Lebenszeit und Größe sind einstellbar.

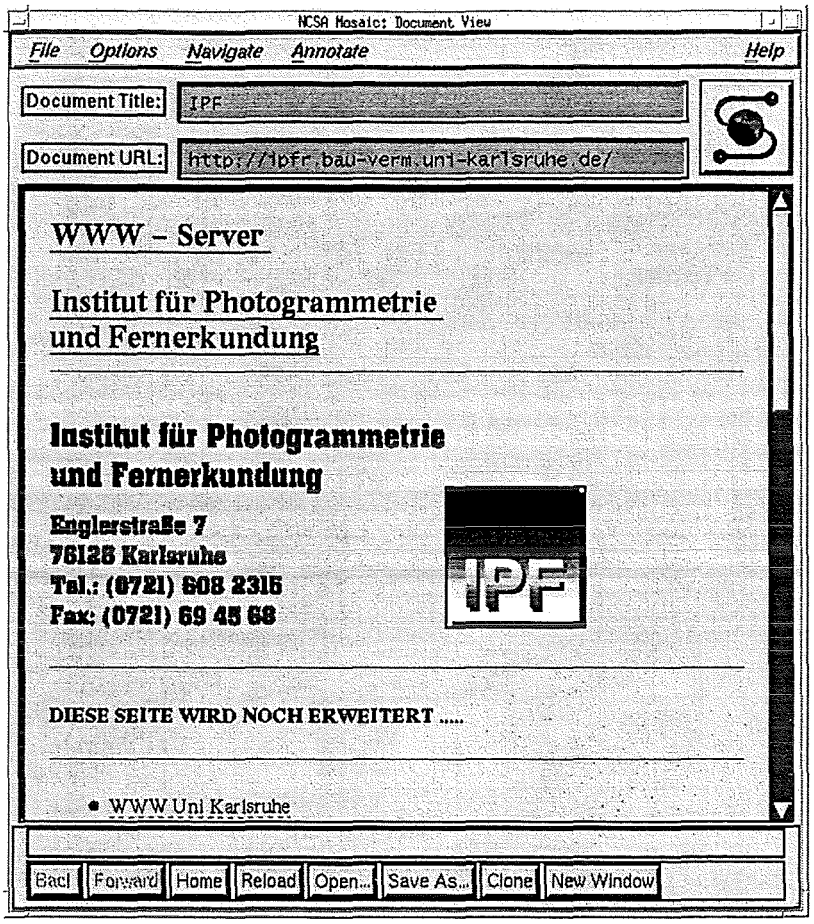


Abbildung 2.3: GLOBUS/WWW: NCSA-Mosaic WWW-Client

2.2 Protokolle und Datentypen im WWW

Ein wichtiges Resultat der WWW-Entwicklung war die Schaffung eines universellen Namensschemas, den URL (Universal Resource Locators). Sie bezeichnen Daten und deren Operatoren. Beispielsweise bezeichnet der URL
`file://server/file.html`
eine Datei (Schlüsselwort `file:`) `file.html` im Dateienbaum eines Rechners `server`.
Weitere typische URLs benennen Dienste im Internet:

`ftp://server/directory`	`Anonymous FTP`
`telnet://server:user`	`Telnet als Nutzer user nach server`
`http://server/directory`	`Hypertextdokument`
`gopher://server/service`	`Gopher Suche`
`wais://server`	`Wais Bibliothekszugriff`

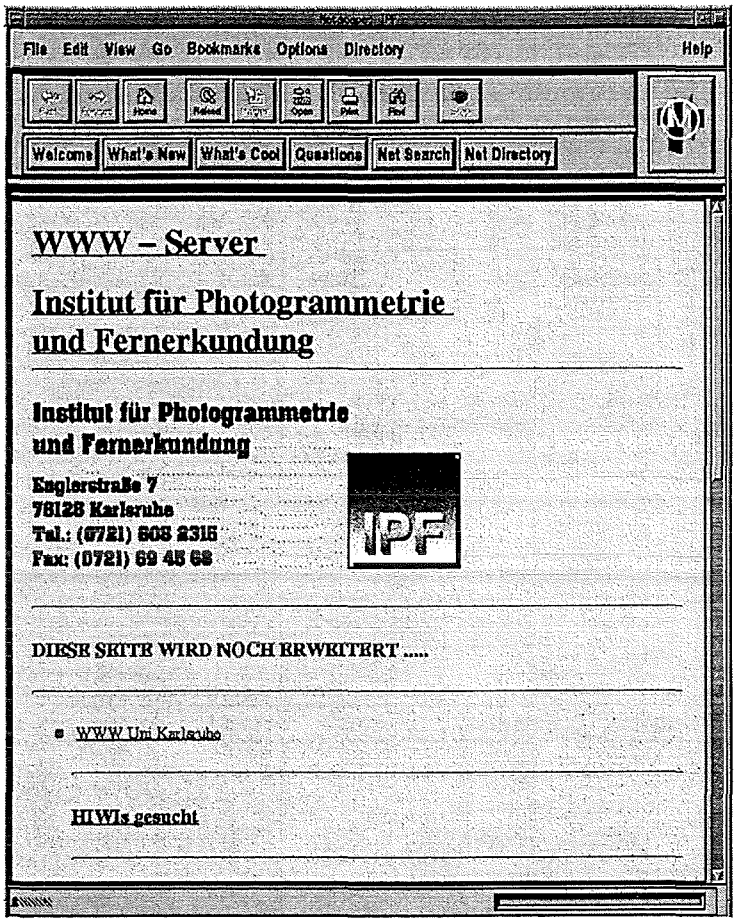


Abbildung 2.4: GLOBUS/WWW: Netscape WWW-Client

Die Eingabe des Kommandos: `Mosaic ftp://rechner/Directory` öffnet z.B. eine FTP-Sitzung zum Rechner `rechner` und zeigt den Inhalt des Verzeichnisses `Directory` als klickbare Liste an.

Zur Übertragung von Hypertextdokumenten wird das Hypertext Transfer Protocol (HTTP) benutzt. Artikel und Dokumentationen zu WWW, HTML usw. sind z.B. in [Ian94, Nat94] und [Klu94c, Klu94b, Klu94a] zu finden. Im Anhang B ist eine übersichtliche HTML-Einführung wiedergegeben.

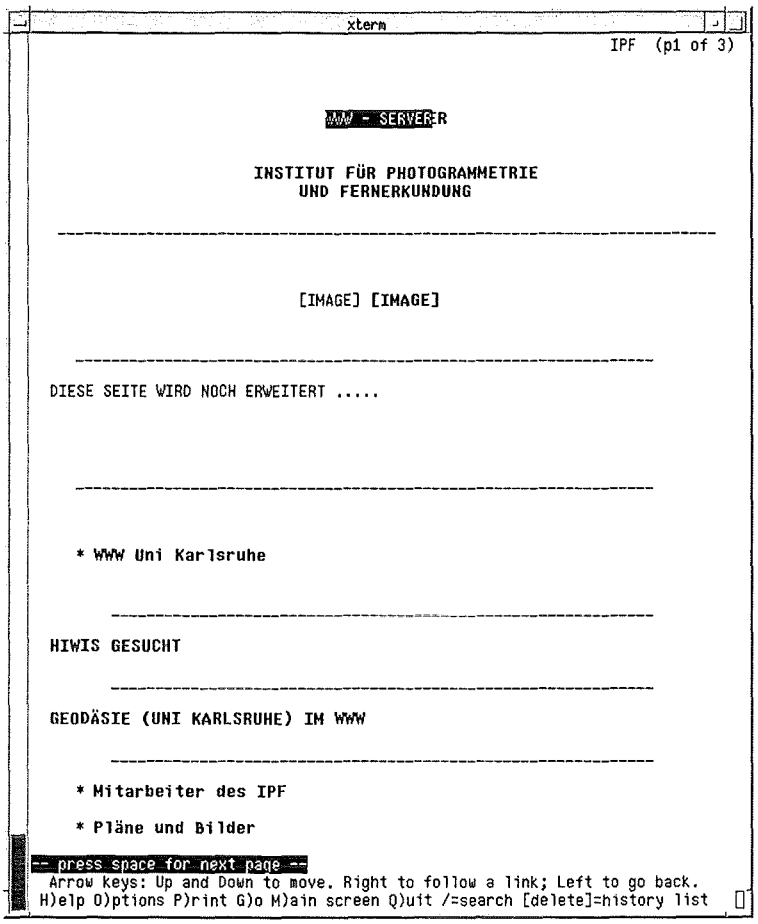


Abbildung 2.5: GLOBUS/WWW: Lynx WWW-Client

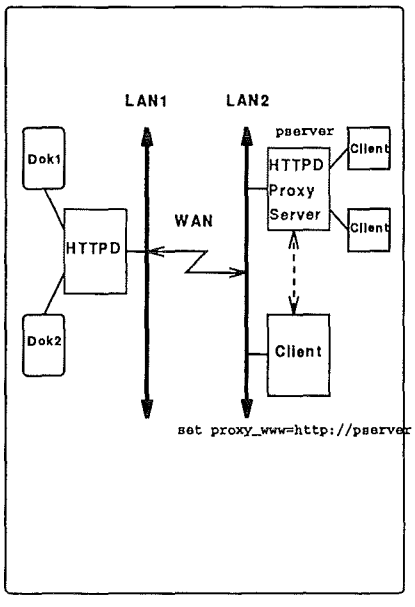


Abbildung 2.6: GLOBUS/WWW: WWW Proxy-Server

Kapitel 3

Dokumentenkonvertierung in das HTML-Format

3.1 Werkzeuge zum Bearbeiten von HTML

Anschließend ist ein Schnappschuß von einem der wichtigsten WWW-Server (`http://info.cern.ch/hypertext/WWW/Tools`) wiedergegeben, der die derzeit verfügbaren Werkzeuge zur Bearbeitung von HTML auflistet.

```
Editing HTML

   Some conventional word-processors and general-purpose SGML
editors can be used with HTML. More information about these
possibilities can be found here . In addition, here are some
purpose-made html editors:

   BBEdit Extensions
          Allow easier edit of HTML files with BBedit on the
          Mac.

   NeXTStep editor
          WYSIWYG hypertext.

   html-mode for Emacs
          Not WYSIWIG but useful.

   html assistant
          for MS Windows

   HTMLed
          an HTML editor for MS-Windows
```

```
   HTML Writer
          another HTML editor for MS-Windows

   HoTMetal
          and HoTMetal Pro from SoftQuad.

   htmltext
          an editor based on the Andrew Toolkit.

   tkHTML
          based on tk/tcl.

   S H E
          Simple HTML Editor - a Hypercard stack.

   HTML Editor
          for the Mac by Rick Giles.

   BBEditLite
          another editor for the Mac.

   Phoenix
          WYSIWIG editor for X-Windows

   ANT
          templates for Word for Windows 6.0 implementing Word
          to HTML and HTML to Word WYSIWYG Conversion Utilities

Generating things from HTML

   Converters
          exist for plain text, LaTeX, MML, RTF etc.

   Other sed scripts can be used to combine documents at
   various levels into one big book.

Maintaining HTML documents

   Analysing HTML
          html_analyzer for assisting in the maintenance of
          HTML databases.
```

Im gleichen WEB-Server findet man auch die zur Zeit (7.12.94) aktuelle Liste von Konvertern, die existierende Dokumente in HTML zu übertragen

verstehen.

PostScript

ps2html Contact: guru@stasi.bradley.edu (Jerry Whelan)

PS2HTML Contacts: agostini@server.area.fi.cnr.it (Alessandro Agostini), ced@server.area.fi.cnr.it (Stefano Cerreti)

MS Word / RTF

* rtftohtml (based on the RTF tools of P.Dubois at Wisconsin). Produces HTML from documents containing graphics, tables, text and equations. Runs on UNIX and Macintosh, but will translate documents produced on DOS and Windows. Latest release 2.7.5. Now also available for DOS.
  Contact: cjh@cray.com (Chris Hector)
* rtftoweb extends the above with some extra features. New version 1.5. Contact: zzhibol@rrzn-user.uni-hannover.de (Christian Bolik)
* rtf2html (more limited)
  Contact: cshotton@oac.hsc.uth.tmc.edu (Chuck Shotton)
* See also Rainbow below
* RTFTOHTM WinWord template and conversion tools.
  Contact: Jorma.Hartikka@csc.fi
* ANT_HTML.DOT is a Word for Windows 6.0 template designed to convert Word documents into HTML documents in a WYSIWYG environment.
  Contact: jswift@freenet.fsu.edu (Jill Swift)
* CU_HTML.DOT is a Word for Windows 2.0 and 6.0 document template that allows users to create HTML documents inside Word in a WYSIWYG manner and generate a corresponding HTML file.
  Contact: anton-lam@cuhk.hk (See also a development based on this by Quantum Research)
* GT_HTML.DOT is an alpha release by Georgia Tech Research Institute (GTRI) of a set of Microsoft Word for Windows macros to facilitate HTML document authoring. The macros are contained in a document template which provides a pseudo WYSIWYG authoring environment.
  Contact: gt_html@gtri.gatch.edu (Jeffrey L.Grover and John H.Davis III)
* SGML Tag Wizard makes Word 6.0 into an interactive SGML and HTML editor.

```
    Contact: evh@dxcern.cern.ch (Eric van Herwijnen)
  * HTML for Word 2.0 by NICE technologies, France, creates
    a structured document environment for Word 2.0.
    Contact: evh@altern.com (Eric van Herwijnen)
  * See also the information on add-ons for HoTMetal.

Word Perfect

  * WPTOHTML WordPerfect macros to convert from WordPerfect
    5.1 for DOS and WordPerfect 6.0 for DOS to HTML.
    Contact: hmonroe@us.net (Hunter Monroe)
  * wp2x (word perfect 5.1 filter converts to html and other
    formats)
    Contact: mcr@ccs.carleton.ca (Michael Richardson)
  * wpmacro (word perfect 5.1 macro, now largely superseded
    by WPTOHTML)
    Contact: steeve@stoner.eps.mcgill.ca (Steeve McCauley)
  * See also the information on add-ons for HoTMetal.

FrameMaker / MIF

  * WebMaker converts FrameMaker documents and books to a
    network of HTML files.
    Contact: webmaker@cern.ch (Bertrand Rousseau et al.)
  * Frame2html. See release note for version 0.8.7e
    Contact: jons@nta.no (Jon Stephenson von Tetzchner)
  * WebWorks Document Translator from Quadralay Corporation.
  * MifMucker is an application for manipulating Frame
    documents and books. It contains a filter to convert
    FrameMaker documents into HTML.
    Contact: harward@convex.com (Ken Harward)
  * www_and_frame (sources and note on current status)
    Contact: connolly@atrium.com (Dan Connolly)
  * miftran (file miftran.tgz). (An MIF to HTML converter
    written in C. See README).
    Contact: jimmc@eskimo.com (Jim McBeath)
  * mif.pl is a Perl library to parse Frame Maker Inter-
    change Format (MIF), designed to be used by filters.
    Contact: ehood@convex.com (Earl Hood)
  * edc2html is a Perl program that generates an HTML
    document to allow navigation through the structure of a
    FrameBuilder Element
    Catalog. Contact: ehood@convex.com (Earl Hood)
```

troff etc

* ms2html (ms to html - good)
  Contact: oscar@iam.unibe.ch (Oscar Nierstrasz)
* me2html for the troff me macros is based on this.
  Contact: troyer@socrates.ucsf.edu (John M. Troyer)
* troff2html. A perl script for the me macros.
  Contact: troyer@cgl.ucsf.edu (John M. Troyer)
* mm2html pre-alpha release of an nroff mm to HTML perl
  script based heavily on ms2html and fm2html.
  Contact: J.Crowcroft@cs.ucl.ac.uk

LaTeX, BibTeX

* latex2html LaTeX to HTML converter program - not only
  does it successfully show the more complex LaTeX
  formatting, for example for mathematics, but it also has
  a set of iconic images, which are included for navigation,
  and to mark footnotes and references. (new version 0.6.2).
  Here is a short tutorial from the CERN computer newsletter.
  Contact: nikos@cbl.leeds.ac.uk (Nikos Drakos)
* vulcanize is a simple perl script that takes care of the
  bulk of most nonmathematical LaTeX documents.
  Contact: mjd@saul.cis.upenn.edu (Mark-Jason Dominus)
* l2h (not as ambitious)
  Contact: jkimball@src.honeywell.com
* Hyperlatex allows the use of a subset of LaTeX to produce
  documents in HTML.
  Contact: otfried@cs.ruu.nl (Otfried Schwarzkopf)
* tex2rtf will convert LaTeX to HTML as well as RTF, Win-
  dows Help RTF and wxHelp. (The source is available here).
  Contact: J.Smart@ed.ac.uk
* The LaTeX parser l2x can be used to produce HTML.
  Contact: hgs@fokus.gmd.de (Henning Schulzrinne)
* axtex is a filter to extract LaTeX "objects" embedded in
  HTML produce a .tex and ultimately an image file for each
  one, and produce an HTML file with the corresponding
  inlines.
  Contact: thrift@ra.csc.ti.com (Philip Thrift)
* The network bibliography at AT&T uses scripts to index
  BibTeX files and to convert them to html on the fly on
  retrieval. More details and software.
  Contact: hgs@research.att.com (Henning Schulzrinne)
* The bibliography on autonomous agents in HCI and CSCW at

Birmingham UK also has tools for BibTeX searching and adding html markup to BibTeX files.
Contact: amw@cs.bham.ac.uk (Andy Wood)

* There is also information on Manipulating BibTeX Bibliographies used for the Computer Science Bibliographies server at Karlsruhe.
Contact: achilles@ira.uka.de (Alf-Christian Achilles)

Texinfo

texi2html (new version Jan 1994)
Contact: cons@dxcern.cern.ch (Lionel Cons)

DECwrite, VAX Document

* decw2html (script to process SGML output)
Contact: p.lister@cranfield.ac.uk
* Here is an awk script to convert SDML to HTML.
Contact: lionel@quark.enet.dec.com (Steve Lionel)

Interleaf

* il2html (scripts to help conversion)
Contact: wilhelms@dlrtcs.da.op.dlr.de (Hartmut Wilhelms)
* and another recipe with comments on WorldView files etc.
Contact: hmiller@tasc.com (Hal Miller)
* and another using Framemaker.
Contact: skip@automatrix.com (Skip Montanaro)

QuarkXPress

* qt2www Perl script to convert Quark-tagged plain text files to HTML. This is a general-purpose conversion utility; it reads in a map file that lists the Quark Tags used in the source file and how they should be translated into HTML. Versions for Unix, DOS and Macintosh.
Contact: jerhy@ltt1.lcs.mit.edu (Jeremy Hylton)
* export filter kits are also available.

Scribe

There is a technical report Producing HTML Documents with Scribe for users of the Scribe document production system.
Contact: trewitt@pa.dec.com (Glenn Trewitt)

```
Linuxdoc-SGML

   Linuxdoc-SGML is a text-formatting package based on SGML that
   will produce HTML as well as other marked up documents.

Rainbow

   Rainbow is an annotated SGML DTD with converters for word
   processer documents (in RTF format) and documents created by
   the Ventura Publisher and Quark Express DTP packages.
   Additional software is needed to convert the SGML into HTML.


------------------------------------------------------------
MS, 7 December 1994
```

Wenn man sich einige der Werkzeuge näher ansieht stellt man fest, daß das Konvertieren zwischen Markupsprachen nahezu perfekt funktioniert. Der LaTeX nach HTML Konverter erzeugt zum Beispiel automatisch Hyperlinks von Inhaltsverzeichnissen zu den Kapiteln und von Kapitel zu Kapitel. Konverter, die existierende Textverarbeitungsdateien umwandeln, leisten dagegen wenig mehr als das reine Umsetzen von Syntax. Dies liegt daran, daß Strukturierung und grafische Darstellung in diesen Systemen nicht strikt getrennt sind. Häufig sind Dokumente auch überhaupt nicht strukturiert, d.h. zum Auszeichnen von z.B. Kapitelüberschriften werden Textattribute verwendet (*fett*), statt sie als *Kapitelbeginn* zu markieren.

Liegen Texte im HTML-Format vor, so bieten verschiedene Tools auch die Rücktransformation in andere Textsprachen. Der im AnhangB abgedruckte *HTML Primer* ist ein Beispiel für eine HTML-nach-LaTeX Konvertierung, die ohne jeglichen manuellen Eingriff stattfand.

3.2 Vorgehensweise bei der Konvertierung

Die zu bearbeitenden Dokumente liegen im Format des Corel Ventura Publisher (Version 4) und in Quark Express vor.

Beiden Formaten ist gemeinsam, daß sie proprietär und - was schlimmer ist - nicht frei zugänglich sind. Bei Corel Ventura ist eine erhebliche Lizenzgebühr und das Unterschreiben eines *Non Disclosure Agreement* nötig, um eine Formatbeschreibung vom Hersteller zu bekommen. Die Verhältnisse bei Quark sind ähnlich.

Anfragen bei Spezialfirmen aus der Druckindustrie zu Konvertern für Publisher-Dateien blieben erfolglos.

Eine nähere Analyse der von der LfU übergebenen Dateien zeigte, daß der Publisher Dokumente aus Steuerdateien (Document- und Chapter-Dateien), den Textdateien und Grafikdateien zusammensetzt. Aus den Chapterdateien geht hervor, welche Textdateien und Grafiken zu einem Kapitel gehören und (vermutlich) an welcher Stelle die Bilder in dem fertig gesetzten Layout stehen. Genaueres zur Positionierung der Bilder war jedoch auch nach intensiven Dekodierungsversuchen, der Konsultation der Handbücher und Befragen von Nutzern nicht herauszufinden. Wir haben uns deshalb entschlossen, alle Hyperlinks zu Bilder eines Kapitels inklusive der Bildunterschriften am Ende eines Kapitels anzuordnen. Die Bilder selbst werden, ebenfalls mit Unterschriften, im GIF als separate Dateien gespeichert.

Die eigentlichen Textdateien sind im Format des Systems abgespeichert, mit dem sie geschrieben wurden (z.B. WinWord, WordPerfect). Sie enthalten jedoch noch weitere Publisher-spezifische Steuersequenzen.

Das gleiche gilt für die Abbildungen, die (weit über 100) in den unterschiedlichsten Formaten erstellt waren (EPS, WMF, BMP, GIF, PIC) und von Ventura mit eigenen Steuersequenzen verändert waren. Alle Bilddateien mußten daher von Hand in die ursprüngliche Form gebracht werden.

Das Ergebnis der Dateianalyse ist im Anhang wiedergegeben.

Zusammenfassend kann man festhalten, daß das LfU-Dokument unvollständig war (es fehlten ganze Abschnitte des Berichts), daß Bilder fehlten und der Text nicht in der letzten, endgültigen Fassung vorlag.

Wie kommt man nun von einem Ventura-Publisher Dokument zu einem HTML Dokument?

1. Alle Texte müssen im Publisher in 7-Bit-ASCII umgewandelt werden. Das Dokument muß jetzt neu abgespeichert werden.

2. Grafikdateien, die nicht im GIF vorliegen müssen nach GIF konvertiert werden. In unserem Fall waren Dateien im WMF, EPS, PIC und BMP Format zu bearbeiten. Zur Konvertierung wurden `pbmplus`, `ghostscript` und *Grafics Workshop* verwendet. Aus allen Bilddateien mußten vorher Publisher-spezifische Steuercodes von Hand entfernt werden.

3. Tabellen müssen mit einem externen Programm als Grafiken eingefangen werden. Wir haben dazu PaintShop und Grafics Workshop benutzt. Diese Grafiken müssen ins gleiche Verzeichnis wie das Dokument als GIF-Dateien abgelegt werden.

4. Es muß eine `ORD` Datei erstellt werden. Sie muß den Namen der Dokumentes tragen und mit `.ord` enden. Bsp: `muell.chp -> muell.ord` Aus der `<kapitelname>.chp` Datei kopiert man alle Pfadangaben mit Ausnahme des ersten Styledateipfades in die `<kapitelname>.ord` Datei und löscht die restlichen Angaben der Zeile. Bsp:

```
.chp
#G 07 c:\1ud\ai\ad_abb01.eps 0040 02B1 02D8
->
.ord
c:\1ud\ai\ad_abb01.eps
```

5. Nun müssen die Pfadnamen nach der Verwendung im Text sortiert werden. Am Anfang der Datei müssen alle Textdatei in der richtigen Reihenfolge stehen. Dann folgen die Grafiken auch in der richtigen Reihenfolge. Besteht das Publisherkapitel aus mehreren realen Kapiteln müssen die Grafiken den einzelnen Kapiteln zugeordnet werden. Zwischen Grafiken verschiedender Kapitel muß eine Leerzeile stehen. Hat ein Kapitel keine Grafiken muß das Schlüsselwort `NoImages` eingetragen werden.

```
Bsp: a-i-01.ord

c:\1ud\ai\bevent-1.doc
c:\1ud\ai\flae_1.doc

c:\1ud\ai\ad_abb01.eps
c:\1ud\missing.gif        U=4
c:\1ud\ai\bevent-1.eps

c:\1ud\ai\a-i-01-t.gif    T
c:\1ud\ai\siedlew.eps     G=2
c:\1ud\ai\siedlfl.eps
```

6. Zuvor in Grafiken verwandelte Tabellen müssen an den richtigen Stellen in der `.ord` Datei hinzugefügt werden und durch ein T nach dem Pfad gekennzeichnit werden. Die Nummer der Bildunterschrift wird wenn kein U oder g Tag angegeben ist aus der Reihenfolgen bestimmt. Bei jedem neuen Kapitel beginnt die Nummer wieder von 1. Stimmt der Text nicht mit diesem System überein, so kann mit Hilfe eines U Tags die Abbildungsnummer bestimmt werden. U=4 bedeutet somit, daß diese Abbildung einen Bilduntertitel hat, der mit Abb. 4 beginnt. Für Spezialfälle können auch reguläre Ausdrücke angegeben werden. Haben mehrere Grafiken einen gemeinsamen Untertitel, so kann das durch einen G Tag beschrieben werden. Aus G=2 folgt, daß diese und die darauffolgende Grafik zusammen einen Untertitel besitzen.

7. Das so vorbereitete Dokument kann jetzt mit cvp2html ubersetzt werden.

Weiterhin möchte ich noch bemerken, daß diese Vorgehensweise höchst unwirtschaftlich ist, da zu viel Handarbeit nötigt ist. Falls der HTML-Umweltbericht kommen soll, dann mit einem Publisher, dessen Format bekannt ist oder der selbst HTML Text erzeugt. Das von uns entwickelte Verfahren ist nicht dazu geeignet, alle folgenden Berichte zu übersetzen.

Ruft man cvp2html ohne Parameter auf so wird eine kleine Usage ausgegeben:

- `-sd` Bezeichnet das source Verzeichnis. In diesem Verzeichnis müssen sich die zu übersetzenden Files stehen.
- `-dd` Bezeichnet das Zielverzeichnis. In diesem werden die neuen HTML Files erstellt.
- `-hd` Pfadanfang der HTML-Links. Somit können Zielverzeichnis und späteres HTML Verzeichnis verschieden sein.
- `-cf` Hier mir der Pfad der `ORD` Datei angegeben werden.

`Cvp2html` kann nur eine `ORD` Datei somit nur eine Publisherdatei übersetzen. Für das ganze Dokument wurde ein Shell-Script geschrieben, das `cvp2html` für alle `ORD` Dateien des Umweltberichts aufruft. Es steht auch in dem oben genannten Verzeichnis und heißt `convertall`. `Cvp2html` gibt auf `stdout` eine Pfadliste der erzeugten Kapiteldateien aus. Auf `stderr` kommen die Fehlermeldungen.

3.3 Vorschläge für zukünftiges Vorgehen

Zukünftige Berichte und Dokumente sollten nicht nur mit der Zielsetzung gestaltet und technisch zusammengestellt werden, daß sie *einmal* druckreif werden sollen und dann als Datenbestand gelöscht werden. Vielmehr muß die Mehrzweckverwendung als elektronisches Dokument im Vordergrund stehen.

Das bedeutet, daß beim Design eines Berichts die Strukturierung des Dokuments im Vordergrund stehen muß. In einer idealen Publishing-Umgebung sollten sich Berichtsautoren nur auf ihre Manuskripte und Inhalte konzentrieren können, während ein Layouter die grafische Ausgestaltung übernimmt.

Eine Auszeichnungssprache, wie z.B. SGML (oder HTML, LaTeX ...) legt das *WAS* fest (Kapitel, Unterkapitel, Hervorhebung, Aufzählungen, Abbildungen, Zitieren, Fußnote usw.), während das Publishing-Programm das *WIE* bestimmt (Schriftgrößen, Seitenstile, Textfonts, Kapitelnumerierung).

Autorenhilfsmittel verstecken die für gelegentliche Nutzer solcher Sprachen kompliziert erscheinende Syntax hinter einer freundlichen Bedieneroberfäche.

Alle wichtigen Hersteller von Textsystemen (Microsoft, Corel) haben entsprechende Zusätze für ihre Systeme angekündigt oder liefern sie bereits aus

(z.B. WordPerfect, Frame). Selbst Microsoft hat während der Herbst-Comdex 1994 einen *Internet Assistant for Word* angekündigt, mit dem man HTML-Dokumente inklusive Hypertexlinks erzeugen und bearbeiten kann.

Eine wünschenswerte Dokumentenproduktion sollte auf einem Drehscheibenformat basieren (siehe Abb. 3.1) (SGML mit entsprechenden DTDs) und darum herum gelagerten Spezialwerkzeugen für die Druckaufbereitung, (z.B. Framemaker) Hypertextpräsentation (HTML), das Editieren (SGML-Editor) und Texterfassen (z.B. Winword mit SGML-Macros).

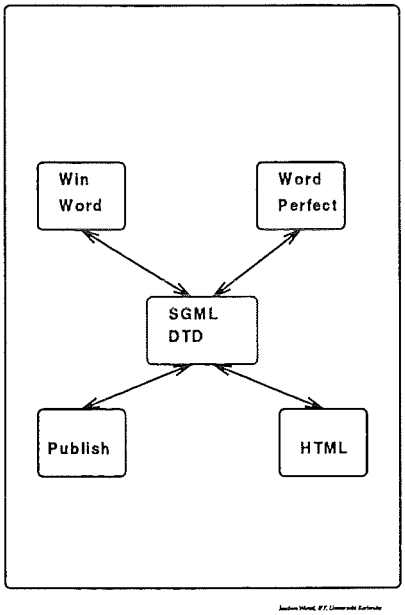


Abbildung 3.1: GLOBUS/Dokumente: Dokumenten-Architektur

Kapitel 4

Dokumentenaufschluß mittels Thesaurus

4.1 Umweltberichte als Hypertext

Durch die in den nachfolgenden Abschnitten beschriebenen Programme wird eine Informationsstruktur bereitgestellt, mit der über verschiedene Einstiegspunkte auf die Informationen im Umweltbericht zugegriffen werden kann (siehe auch Folie im Anhang).

Eine erste Möglichkeit ist die Navigation über die hierarchische Kapitelstruktur des Berichtes, d.h. ausgehend vom Gesamtinhaltsverzeichnis kann zu den Inhaltsverzeichnissen der einzelnen Kapitel und von dort zu den Berichtsabschnitten gesprungen werden.

Ein weiterer Einstiegspunkt ist über den alphabetischen Index, der Deskriptoren und Synonymbegriffe enthält, gegeben. Von den Begriffen des alphabetischen Index kann direkt auf das Deskriptordokument des entsprechenden Deskriptors zugegriffen werden. Von diesem Deskriptordokument wird auf Ober- und Unterbegriffe dieses Deskriptors verwiesen; auf diese Weise sind die Deskriptordokumente ebenfalls hierarchisch untereinander verknüpft. Außerdem ist hier die Verbindung zwischen der Berichtsstruktur und dem Thesaurus gegeben dadurch, daß eine Deskriptorseite mit allen Berichtsabschnitten verknüpft ist, denen dieser Deskriptor zugeordnet wurde.

4.1.1 Automatische Zuordnung von Deskriptoren aus dem Thesaurus zu den Berichtsabschnitten

Im ersten Schritt werden die bereits als HTML-Dokumente vorliegenden und in eine hierarchische Verzeichnistruktur eingeordneten Berichtsabschnitte automatisch verschlagwortet, d.h. in jede Seite werden dabei Zeilen eingefügt, die die Deskriptoren für diesen Berichtsabschnitt enthalten. Verwendet wird dabei der Umwelt-Thesaurus des Umweltbundesamtes, der in unterschiedlichen Ausgabeformen zur Verfügung steht, wobei für die Deskriptorvergabe

die (bearbeitete) systematische Liste benutzt wird.

Alle Programme und Prozeduren zur Deskriptorvergabe werden von einem Shell-Script ("index.csh") aufgerufen. Diesem Script ist als Parameter der Name der HTML-Datei zu übergeben; falls kein Parameter übergeben wird, wird der Dateiname abgefragt. Der Dateiname muß mit ".html" enden; diese Endung muß jedoch nicht mit eingegeben werden. Ausgabe dieses Shell-Scripts ist dann die HTML-Datei mit eingefügten Deskriptoren. Dieser Gesamtablauf ist in der folgenden Abbildung dargestellt:

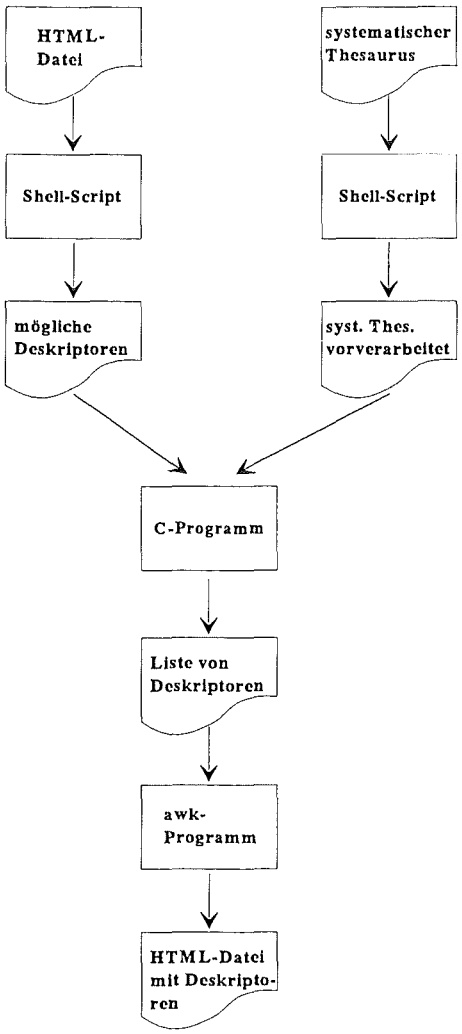


Abbildung 4.1: Gesamtablauf

Zuerst wird die HTML-Datei in einzelne Wörter aufgeteilt, d.h. jede Zeile der Ausgabe enthält ein Wort der HTML-Datei, wobei bereits eine Vorverar-

beitung stattfindet. Die Aufteilung in Wörter erfolgt in mehreren Schritten, die durch "Pipes" miteinander verbunden sind.

Im ersten Teilschritt durchlaufen die Eingabedaten ein sed(= "stream editor")-Script ("deltag.sed"), das

- alle Zeilen bis <BODY> löscht,
- alle Hypertext-Links löscht,
- alle sonstigen HTML-Tags löscht,
- Sonderzeichen durch Blanks ersetzt und
- Umlaute durch ae (oe, ue) bzw. ß durch ss ersetzt.

Danach ersetzen tr-Befehle (tr = "translate characters") alle Blanks durch Return und alle Groß- durch Kleinbuchstaben. Die daraus entstehende Wortfolge wird mit "sort" alphabetisch sortiert und als Eingabe für ein awk("pattern scanning and processing language")-Programm ("stopwort.awk") benutzt, welches Stopwörter, Zahlen, einbuchstabige Wörter und Leerzeilen entfernt.

Die Ausgabe bis zu diesem Zeitpunkt besteht also aus einer Liste von Wörtern, die möglicherweise als Deskriptoren zu verwenden sind (siehe Abbildung).

Diese Liste wird zusammen mit dem (vorverarbeiteten) systematischen Thesaurus in einem C-Programm ("deskript.c") verarbeitet, welches aus den beiden Eingabedateien diejenigen Wörter heraussucht, die in beiden Dateien vorkommen bzw. die sich nur durch die letzten beiden Buchstaben unterscheiden (siehe Abbildung).

Falls gleiche Wörter gefunden werden, die im systematischen Thesaurus mit "BS" ("Benutze Synonym") oder mit "BK" ("Benutze Kombination") gekennzeichnet sind, werden sie in der Ausgabe durch die entsprechenden Synonyme bzw. Kombinationen ersetzt.

Als Ergebnis liefert das C-Programm also eine Liste von Deskriptoren, wobei Deskriptoren, die im HTML-Dokument mehrfach vorkommen, auch mehrfach aufgeführt sind. Diese Liste ist wegen der durch Synonyme oder Kombinationen ersetzten Wörter nicht mehr alphabetisch sortiert.

Für das C-Programm wurde der systematische Thesaurus mit Hilfe eines Shell-Scripts ("systkurz.csh"), bestehend aus awk-Befehlen (siehe Abbildung), vorverarbeitet. Dabei werden aus dem Thesaurus die zu jedem Deskriptor aufgeführten Informationen, wie Ober- und Unterbegriffe, englische Begriffe usw., entfernt, da sie für die Deskriptorvergabe nicht relevant sind. Die zu Nicht-Deskriptoren aufgeführten und statt deren zu benutzende Synonyme und Kombinationen (s.o.) bleiben erhalten. Der vorverarbeitete systematische Thesaurus entspricht also eigentlich dem alphabetischen Thesaurus; nur sind zusätzlich abweichende Benennungen mit den dafür zu benutzenden Synonymen oder Kombinationen enthalten.

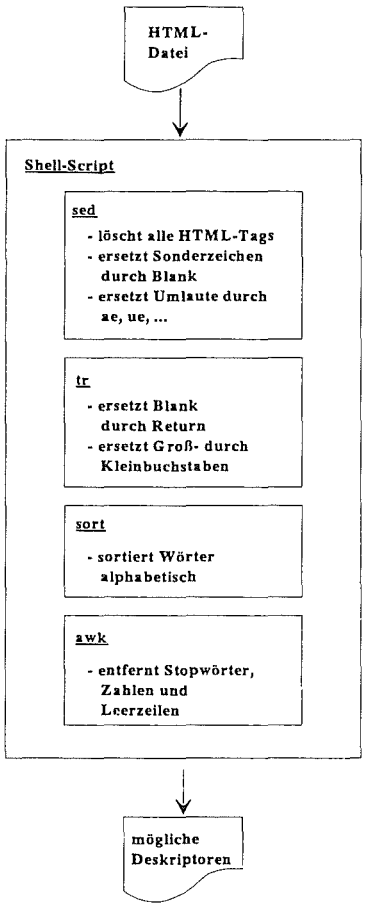


Abbildung 4.2:

Das Einfügen der Deskriptoren als Links in die HTML-Datei wird von einem awk-Programm ("deskript.awk") übernommen, welchem als Parameter der Name der HTML-Datei und eine Liste der einzufügenden Deskriptoren übergeben werden. Diese Deskriptorliste baut auf der vom C-Programm ausgegebenen Liste auf; im Unterschied zu dieser ist sie jedoch nach der Häufigkeit, mit der die einzelnen Deskriptoren vorkommen, sortiert, und es gibt keine Mehrfach-Einträge mehr. Das awk-Programm fügt - neben den Deskriptor-Links - auch (unsichtbare) Kommentarmarken ein, so daß die Stelle, ab der die Deskriptoren aufgeführt sind (bzw. wo die Auflistung endet), bei späteren Verarbeitungsschritten (z.B. bei der manuellen Korrektur) eindeutig festgestellt werden kann.

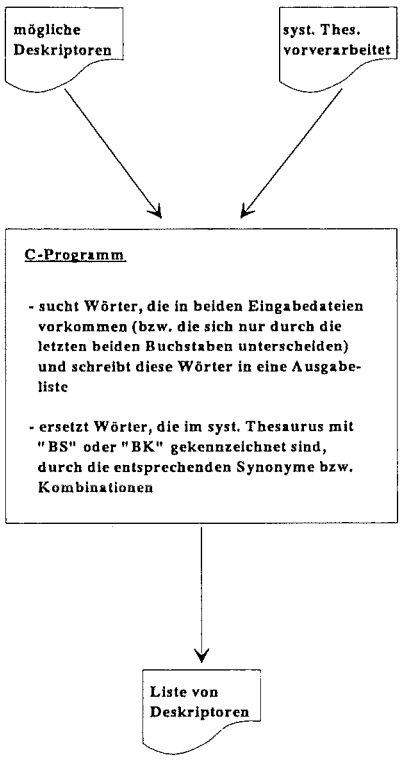


Abbildung 4.3:

4.1.2 Manuelle Korrektur der Deskriptoren in den Berichtsabschnitten

Die automatisch eingefügten Deskriptoren sind manuell änderbar, d.h. es ist möglich, einen Deskriptor zusätzlich einzugeben (aus einer fest vorgegebenen Liste) oder einen der automatisch zugeordneten zu löschen. Realisiert sind diese Funktionen durch ein TCL/TK-Skript.

4.1.3 Umsetzung des Thesaurus in einen hierarchischen Hypertext aus Deskriptordokumenten

In einem weiteren Schritt wird der konkrete Thesaurus aufgebaut, d.h. es werden die Deskriptordokumente generiert. Jeder Deskriptor in einem Berichtsabschnitt führt nun zu einem Deskriptordokument, in das im nächsten

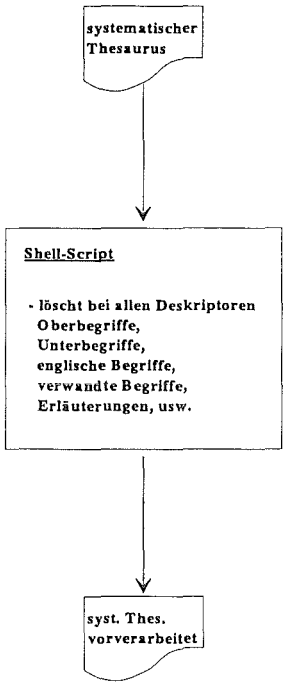


Abbildung 4.4:

Schritt alle Berichtsabschnitte (wiederum als Hyperlinks), die dieses Wort als Deskriptor enthalten, eingefügt werden. Die Deskriptordokumente werden mit Hilfe eines C-Programmes generiert, das dazu einen (vorverarbeiteten) systematischen Thesaurus benutzt.

Die Vorverarbeitung des Thesaurus sieht in diesem Fall so aus, daß ein awk-Programm alle Synonyme und Kombinationen sowie die englischen Übersetzungen aus dem systematischen Thesaurus entfernt; übrig bleiben also nur die Deskriptoren mit Oberbegriffen, Unterbegriffen, verwandten Begriffen usw. Diese Informationen erscheinen dann auch in den Deskriptordokumenten, wobei z.B. ein Unterbegriff in einem Deskriptordokument, zu dem wieder eine eigene Deskriptorseite vorhanden ist, als Hypertext-Link realisiert ist. Zusätzlich ist der Deskriptor ein Hyperlink, der zu einerm alphabetischen Index aller Deskriptoren und Synonymbegriffe führt.

4.1.4 Aufbau eines alphabetischen Index für Deskriptoren und Synonymbegriffe

Der alphabetische Index wird aus dem vorverarbeiteten systematischen Thesaurus, der auch für die Deskriptorvergabe benutzt wurde, von einem awk-Programm ("alphhtml.awk") generiert. Berücksichtigt werden bei diesem alphabetischen Index nur Einwort-Begriffe, da auch bei der Deskriptorvergabe bisher nur nach Einwort-Begriffen gesucht wurde.

4.1.5 Aufbau von Hyperlinks zwischen Berichtsabschnitten und Deskriptordokumenten

Die Hyperlinks von den Berichtsabschnitten zu den Deskriptordokumenten werden bereits bei der Deskriptorvergabe angelegt, da die zu jedem Berichtsabschnitt gefundenen Deskriptoren sofort als Hyperlinks eingefügt werden. Die Verweise in der umgekehrten Richtung, also von jedem Deskriptordokument zu allen Berichtsabschnitten, die diesen Begriff als Deskriptor aufweisen, werden in einem weiteren Schritt eingefügt. Der Ablauf für einen bestimmten Berichtsabschnitt sieht dabei so aus, daß alle Deskriptoren dieses Berichtsabschnittes gelesen werden. Anschließend wird in die entsprechenden Deskriptordokumente der Dateiname dieses Berichtsabschnittes (bzw. der Titel) als Hyperlink eingefügt.

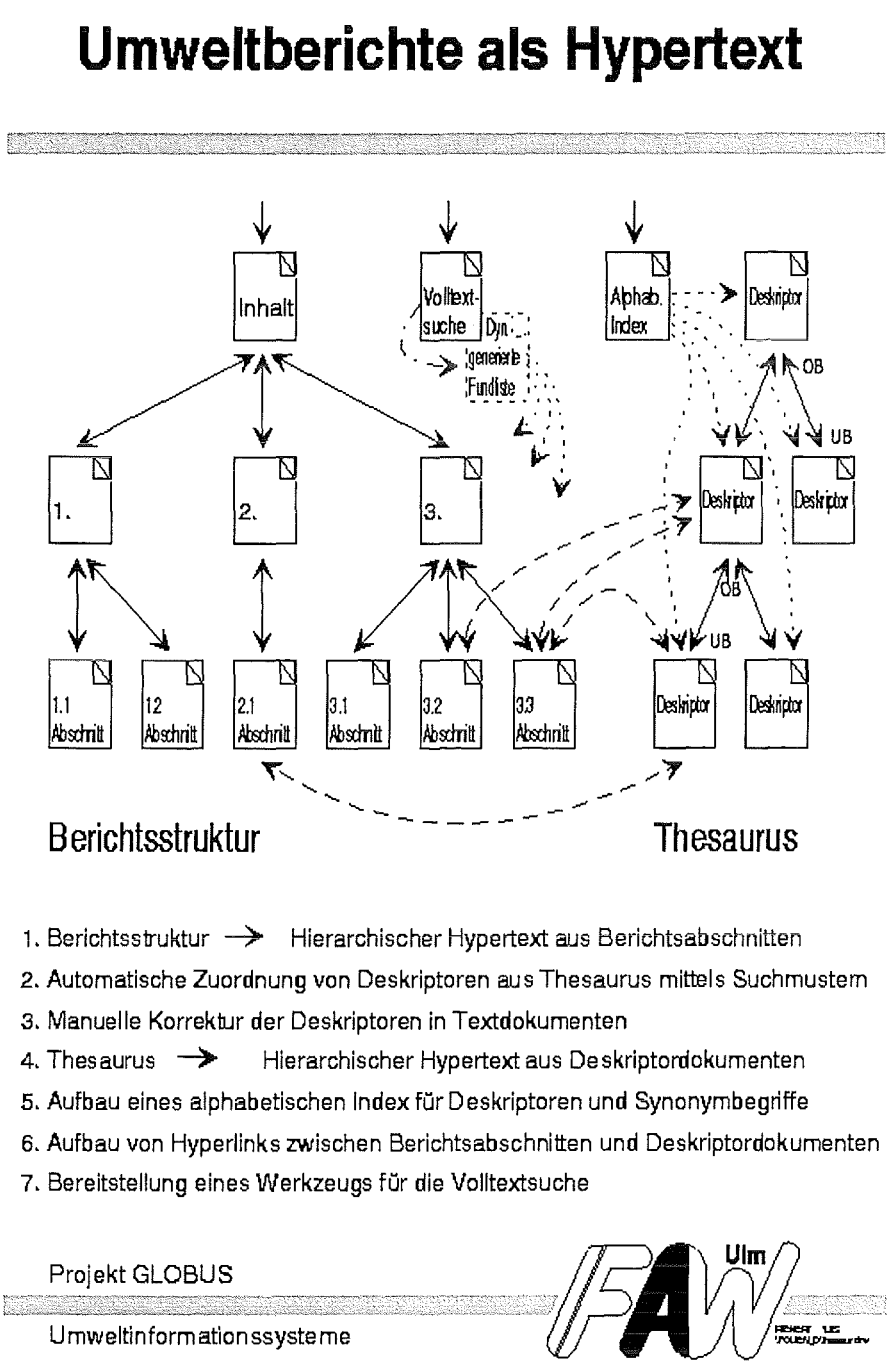


Abbildung 4.5:

Kapitel 5

Zugriff auf den UDK via WWW/Oracle Gateway

5.1 Einleitung

In diesem Teil der GLOBUS-Dokumentation wird der Zugriff via Mosaic auf die Daten des Umweltdatenkatalog UDK beschrieben, wie er am Forschungszentrum Informatik (FZI), Karlsruhe, entwickelt und realisiert wurde. Die Daten des UDK liegen dabei in einem relationalen Datenbanksystem (Oracle Version 7) vor. Die Besonderheit – und damit zugleich die Herausforderung bei der Entwicklung und Realisierung – liegt daher im Zugriff auf einen sich dynamisch ändernden Datenbestand. Die letztendlich benötigten HTML-Seiten liegen im Gegensatz zu den übrigen Teilen von GLOBUS nicht statisch vor, sondern werden – in Abhängigkeit von der aktuellen Benutzeranfrage – zur Laufzeit dynamisch erzeugt.

Als schriftliche Dokumentation, in der die Realisierung der PC-Anwendung des UDK beschrieben wird, standen im wesentlichen [Nie94] und [Dr.93] zur Verfügung. Als Daten stand ein Datenbankabzug der LfU Karlsruhe zur Verfügung. Im folgenden Abschnitt 5.2 beschreiben wir zunächst die Software-Architektur des realisierten Systems. In Abschnitt 5.3 wird die realisierte Funktionalität aus Anwendersicht beschrieben. Abschnitt 5.4 faßt knapp die erzielten Ergebnisse zusammen. Abschnitt 5.5 gibt schließlich einen Ausblick auf sinnvolle Fortführungen. Im Anhang 5.6 werden Hinweise zur Installation gegeben.

5.2 Software-Architektur

Abbildung 5.1 zeigt die Software-Architektur des realisierten Systems. Über einen Verweis auf die Startseite, die als HTML-Seite beim WWW-Server vorliegt, erhält man Zugang zum UDK. Die weiteren Anfragemasken und Ergebnisseiten werden dynamisch durch das Programm `udk_zugriff` generiert und an den jeweiligen Anwender geschickt. Ausgehend von einer solchen

Seite kann der Anwender innerhalb der Standardfunktionalität von Mosaic rückwärts navigieren oder die jeweils folgende Seite anfordern.

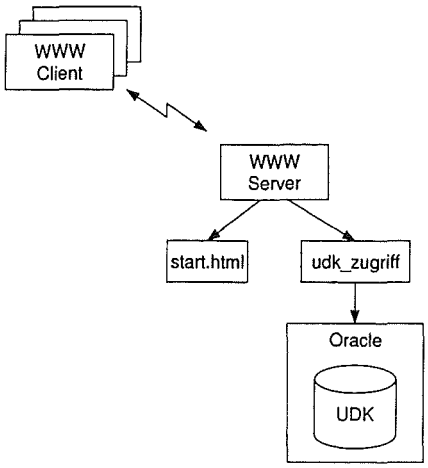


Abbildung 5.1: GLOBUS/UDK: Software-Architektur

5.3 GLOBUS/UDK aus Anwendersicht

Abbildung 5.2 zeigt die Anfangsmaske (`start_html`), von der ausgehend die Seite zur Katalogauswahl erreicht werden kann.

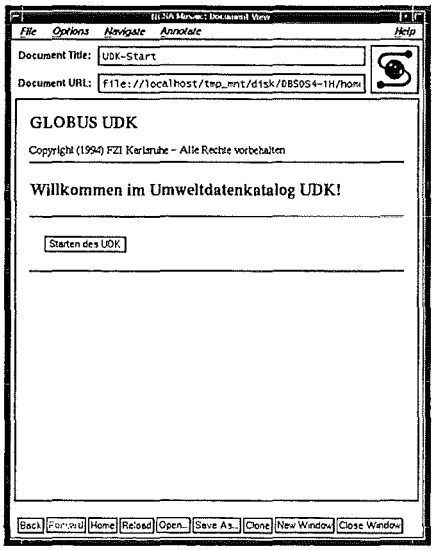


Abbildung 5.2: GLOBUS/UDK: Anfangsmaske

5.3.1 Generelle Auslegung der Suche

Für die bei der allgemeinen Suche nach Adressen bzw. UDK-Objekten (Abschnitt 5.3.3.1 bzw. 5.3.4.1) sowie bei der Detailsuche nach Objekten

(Abschnitt5.3.4.2) anzugebenden Suchbegriffe gilt folgendes:

- Es wird zwischen Groß- und Kleinschreibung unterschieden.
- Entsprechend dem SQL-Standard ist % der Platzhalter für kein, ein oder beliebig viele Zeichen.
- Als Platzhalter für genau ein Zeichen fungiert ?.

5.3.2 Katalogauswahl

Abbildung 5.3 zeigt die Maske zur Katalogauswahl. In dieser werden sämtliche der in der Relation `katalog` verzeichneten Kataloge zur Auswahl angeboten. Die in den Abschnitten 5.3.3.1, 5.3.4.1 und 5.3.4.2 beschriebenen Suchen, die in diesen Masken angewählt werden können, erstrecken sich jeweils auf den in diesem Schritt gewählten UDK-Katalog bzw. die damit implizit gewählte Kombination von `staat` und `land`

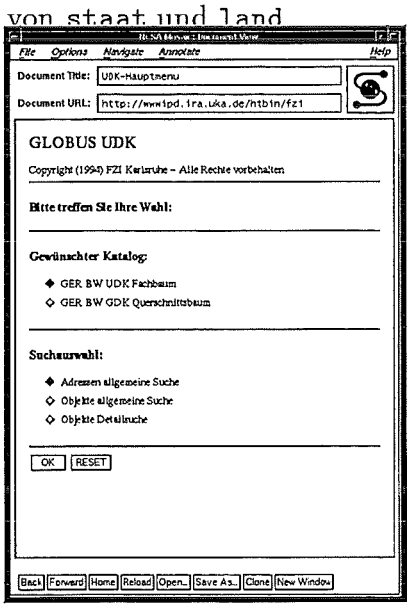


Abbildung 5.3: GLOBUS/UDK: Maske zur Katalogauswahl

5.3.3 Suche nach Adressen

Die in den folgenden Abschnitten beschriebene und realisierte Funktionalität entspricht der in [Dr.93, Nie94] für die entsprechenden Teile beschriebenen.

5.3.3.1 Allgemeine Suche

Abbildung 5.4 zeigt die Maske zur allgemeinen Adreßsuche. Mit dem hier eingegebenen Suchstring wird in den folgenden Relationen in den aufgeführten Attributen gesucht:

- Relation `institut`
  - Attribut `name`
- Relation `abt` (Abteilung)
  - Attribut `bereich`
  - Attribut `bezeichnung`
- Relation `stelle`
  - Attribut `adr_kennz`
  - Attribut `nachname`
- Relation `adr_search`
  - Attribut `suchbegriff`

In den einzelnene Relationen logisch gelöschte Datensätze (im Attribut `aenderung` markiert) werden *nicht* mitausgegeben.

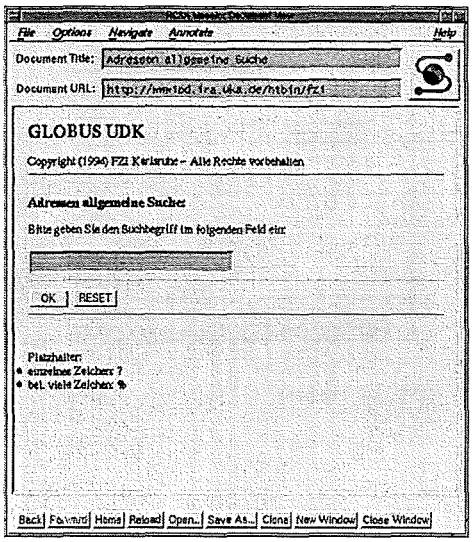


Abbildung 5.4: GLOBUS/UDK: Maske zur allgemeinen Adreßsuche

5.3.3.2 Ausgabe Adreßliste

Abbildung 5.5 zeigt die Ausgabe der Adreßliste. Ausgehend von dieser Liste können eine oder beliebig viele Adressen für eine vollständige Ausgabe ausgewählt werden.

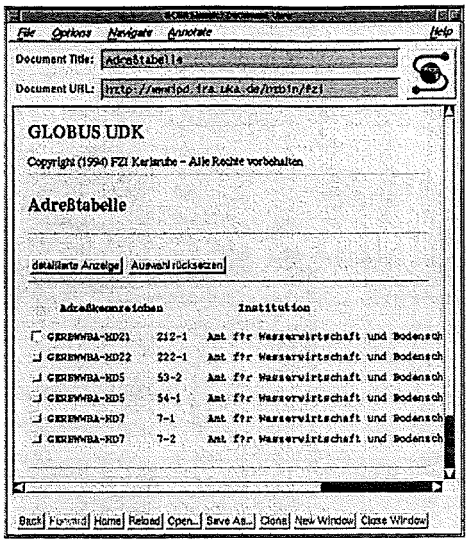


Abbildung 5.5: GLOBUS/UDK: Ausgabe der Adreßliste

5.3.3.3 Ausgabe der vollständigen Adressen

Abbildung 5.6 zeigt die Ausgabe der vollständigen Adressen. Zu jedem der zuvor ausgewählten Adreßkennzeichen werden sämtliche verfügbaren Adressen angezeigt; aufgrund der Eigentümlichkeiten des Datenbankentwurfs (s.a. Abschnitte 5.4 und 5.5) sind hier mehrere Adressen möglich.

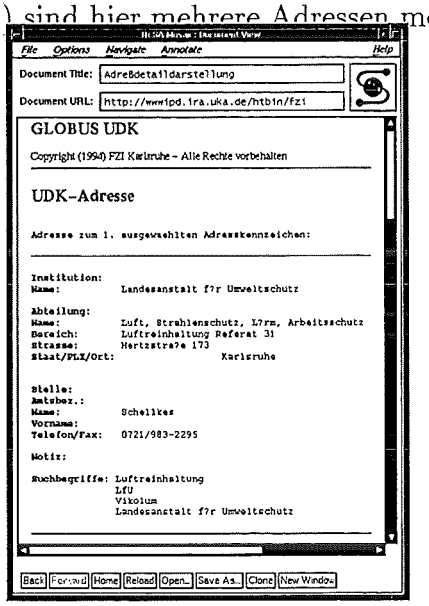


Abbildung 5.6: GLOBUS/UDK: Ausgabe der vollständigen Adressen

5.3.4 Suche nach UDK-Objekten

Die in den folgenden Abschnitten beschriebene und realisierte Funktionalität entspricht wiederum im wesentlichen der in [Dr.93, Nie94] für die entspre-

chenden Teile beschriebenen. Abweichungen ergeben sich in der Detailsuche nach Objekten (s. Abschnitt 5.3.4.2) aufgrund des Kommunikationsmodells von WWW-Anwendungen (s.a. Abschnitt 5.4).

5.3.4.1 Allgemeine Suche

Abbildung ?? zeigt die Maske zur allgemeinen Suche nach UDK-Objekten. Mit dem hier eingegebenen Suchstring wird in den folgenden Relationen in den aufgeführten Attributen gesucht:

- Relation `dek`
  - Attribut `obj_name1`
  - Attribut `obj_name2`
- Relation `obj`
  - Attribut `qadr_kennz`
- Relation `obj_lines`
  - Attribut `beschreibung`
- Relation `obj_search`
  - Attribut `suchbegriff`

Logisch gelöschte Datensätze in den einzelnen Relationen werden wiederum nicht berücksichtigt.

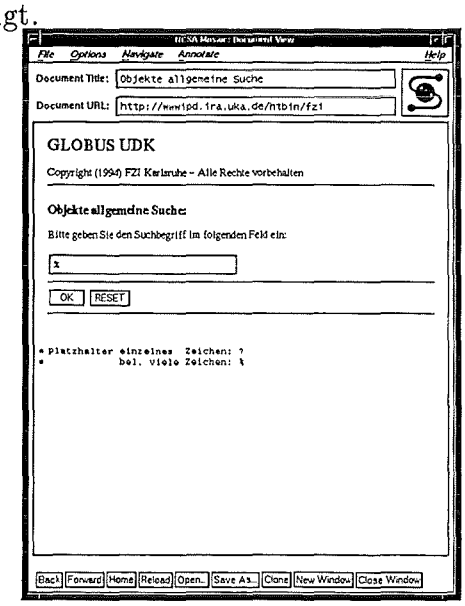


Abbildung 5.7: GLOBUS/UDK: Maske zur allgemeinen UDK-Objektsuche

5.3.4.2 Detailsuche

Abbildung 5.8 zeigt die Maske zur Detailsuche nach UDK-Objekten. Mit der Detailsuche hat der Benutzer die Möglichkeit, eine komplexe Suchabfrage zusammenzustellen. Er kann hierzu aus den folgenden Bezugsfeldern auswählen, die nach Suchbegriffen durchsucht werden sollen:

1. Objektname
2. Adreßkennzeichen
3. Dekadische Notation
4. Beschreibung
5. Raumbezug
6. Zeitbezug
7. Fachbezug

Mi den Werten für die drei letzten Bezugsfelder werden *sämtliche* Attribute der jeweils zugrundeliegenden Relation durchsucht.

Rechts von den Bezugsfeldern können folgende Operatoren ausgewählt werden, die für den Vergleich von Bezugsfeld und dem folgenden Textfeld verwendet werden:

- =
- <>
- >=
- <=
- >
- <

In das anschließende Textfeld kann der Suchbegriff eingegeben werden. Als Platzhalter für ein Zeichen ist das ?-Zeichen vorgesehen. Sollen beliebig viele Zeichen ersetzt werden, so kann das %-Zeichen verwendet werden. Bei fehlender Texteingabe wird automatisch ein %-Zeichen für die Suche benutzt.

Um eine komplexe Suchabfrage zu erstellen, können maximal 5 einzelne Suchabfragen miteinander verbunden werden. Der `Und`-Operator bildet die Schnittmenge aus den Ergebnismengen der verknüpften Einzelabfragen, der `Oder`-Operator vereinigt die Ergbnismengen. Die Anfragen werden hierbei entsprechend dem aktuellen Implementierungsstand des Oracle-Datenbanksystems sequentiell ausgewertet.

Um eine komplexe Anfrage abzuschließen, ist nach der letzten Einzelanfrage der `Ende`-Operator auszuwählen.

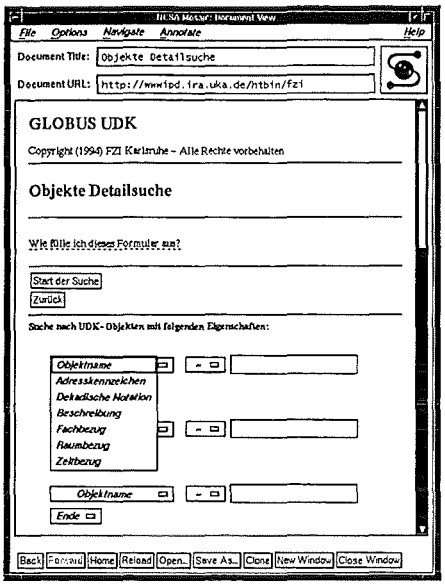


Abbildung 5.8: GLOBUS/UDK: Maske zur Detailsuche nach UDK-Objekten

5.3.4.3 Ausgabe UDK-Objektliste

Abbildung 5.9 zeigt die Ausgabe der UDK-Objekt als Liste. Ausgegeben wird der jeweilige Objektname (Attribut `obj_name1` der Relation `dek`). Für eine vollständige Ausgabe können wiederum eines oder beliebig viele der Objekte ausgewählt werden.

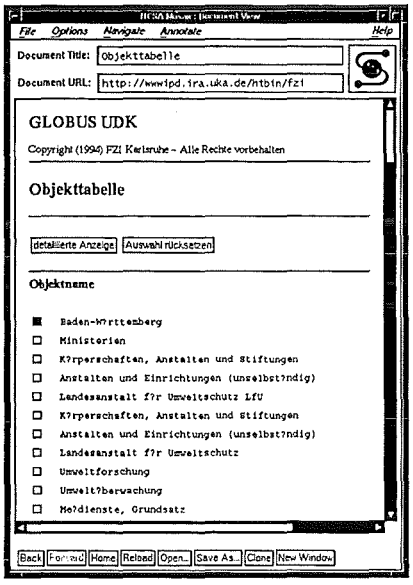


Abbildung 5.9: GLOBUS/UDK: Ausgabe der UDK-Objektliste

5.3.4.4 Ausgabe der vollständigen UDK-Objekte

Abbildung 5.10 zeigt die Ausgabe der vollständigen UDK-Objekte. Die folgenden Attribute werden ausgegeben:

- Relation `dek`
  - Attribut `obj_name1`
  - Attribut `obj_name2`
  - Attribut `dek`; dekadische Notation
- Relation `obj`
  - Attribut `oadr_kennz` als Adreßkennzeichen;
  - Attribut `qadr_kennz` als Datenauskunft;
  - Attribut `showlevel` als Freigabe;
  - Attribut `daten_art` als Datenart;
  - Attribut `verweis_fis` als IS;
- Relation `obj_lines`
  - Attribut `beschreibung`

  Die Ausgabe erfolgt als Notiz bzw. Beschreibung.
- Relation `obj_search`
  - Attribut `suchbegriff`
- Relation `obj_mess` mit Ausnahme des zusammengesetzten Primärschlüssels vollständig;
- Relation `obj_ort` mit Ausnahme des zusammengesetzten Primärschlüssels vollständig;
- Relation `obj_zeit` mit Ausnahme des zusammengesetzten Primärschlüssels vollständig;

5.4 Wesentliche Ergebnisse

Die wesentlichen Ergebnisse und Erkenntnisse des Teilprojekts GLOBUS/UDK können in den folgenden Punkten zusammengefaßt werden:

1. Es wurde am Beispiel des Umweltdatenkatalogs UDK exemplarisch gezeigt, wie mit dem Public Domain Tool Mosaic lesend auf einen bereits existierenden (Meta-) Datenbestand, der in einem relationalen Datenbanksystem verwaltet wird, zugegriffen werden kann. Insbesondere wurde gezeigt, wie bei der Detailsuche nach Objekten (s. Abschnitt 5.3.4.2 und Abbildung 5.8) auch eine sehr komfortable Anfragefunktionalität realisiert werden kann.

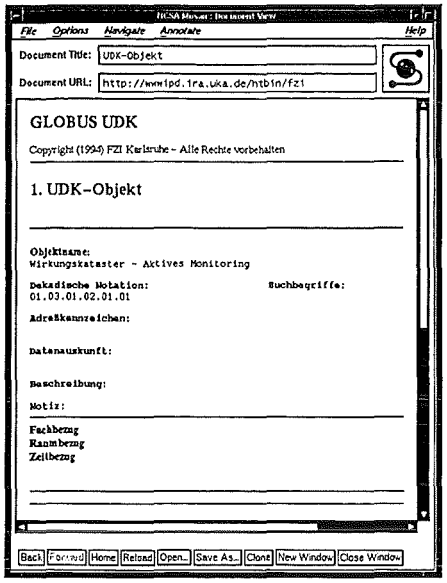


Abbildung 5.10: GLOBUS/UDK: Ausgabe der vollständigen UDK-Objekte

2. Die Realiserung kombiniert die hierarchischen Navigationsmöglichkeiten von Mosaic mit der deklarativen Anfragesprache SQL: Im Gegensatz zu einer reinen Mosaic-/HTML-Anwendung kann der Anwender damit das ihn interessierende Ergebnis beschreiben, ohne – direkt oder indirekt – navigieren zu müssen.

3. Das Kommunikationsmodell für WWW-Server kennt keinen Sessionbegriff, wie er aus dem ISO/OSI-Schichtenmodell bekannt und für klassische Datenbanksystem-Anwendungsprogramme üblich ist. Die sich hieraus ergebenden Konsequenzen wurden bereits bei Programmen, die ausschließlich lesende Datenbankzugriffe umfassen, deutlich.

4. Der Aufwand zur Realisierung eines Datenbankzugriffs wie Mosaic wird wesentlich von der Qualität des Datenbankentwurfs bestimmt. Ist der Datenbankentwurf wie im hier vorliegenden Fall unzureichend dokumentiert und in wesentlichen Details alles andere als unabhängig von der Implementierung eines bereits existierenden Anwendungsprogramms und damit anhand der gebräuchlichen Entwurfskriterien kaum nachvollziehbar, sind die im Einzelfall erforderlichen SQL-Anfragen nur mit extrem hohem Aufwand zu ermitteln.

5.5 Ausblick

Aufbauend auf den bereits erzielten Ergebnissen sind die folgenden Punkte einer eingehenderen Betrachtung wert:

1. *Überarbeitung des Datenbankentwurfs für den UDK*

Um die Qualität der in UDK erfaßten Metadaten langfristig auch bei einer breiten Nutzung sicherzustellen, erscheint es u.E. dringend geboten, beim Datenbankentwurf elementare Entwurfsprinzipien (wie Datenunabhängigkeit) zu berücksichtigen sowie die Möglichkeiten heutiger relationaler Datenbanksysteme (wie Fremdschlüsseldeklarationen) zu nutzen.

2. *Überdenken der Anfragefunktionalität der UDK-Anwendung*

   Wird ein Anfrageparameter wie das sog. Bezugsfeld `Raumbezug` bei der Detailsuche nach UDK-Objekten (s. Abschnitt 5.3.4.2) grundsätzlich auf *sämtliche* Attribute der zugrundeliegenden Relation abgebildet, wird bereits bei einfachen Abfragen auf Gleichheit eine Obermenge der tatsächlich interessierenden Ergebnisdatensätze geliefert, aus denen dann die tatsächlich interessierenden Datensätze manuell herausgesucht werden müssen.

   Wird beispielsweise nach den X-, Y- und Z-Koordinaten mit den Werten 1, 2, bzw. 3 gesucht, so werden zugleich auch sämtliche Permutationen dieser Werte, insgesamt also 6 Datensätze statt des gesuchten einen Datensatzes, ausgegeben.

3. *Dezentrale Metadatenbestände*

   Dem UDK liegt implizit die Prämisse eines zentralen Metadatenbestandes zugrunde. Es ist zu prüfen, ob diese Prämisse bei Nutzung von Rechnernetzen weiterhin sinnvoll ist oder ob nicht verteilte Kataloge, die on-line zugreifbar sind, eine sinnvollere Alternative (einfachere Administration, erhöhte Aktualitat) darstellen.

4. *Direkter Zugriff auf die Originaldaten*

   Bislang dient der UDK primär als Auskunftssystem, mit dem ermittelt werden kann, wo welche Daten wie verfügbar sind. Ein direkter Zugriff auf die Originaldaten, wie er in Rechnernetzen möglich ist, wird hingegen nicht unterstützt. Ein solcher Zugriff ist entweder auf entsprechenden HTML-Seiten oder aber auf Datenbanken denkbar.

5.6 Installation

Bei der Entwicklung der Mosaic-Schnittstelle wurde darauf geachtet, daß das System möglichst einfach auf andere Betriebssystem-Welten und andere HTTP-Server übertragbar ist. In diesem Abschnitt werden die notwendigen Schritte für eine Installation beschrieben. Das System besteht aus zwei Teilen:

- einer HTML-Seite als Einstiegspunkt sowie

- dem Programm `udk_zugriff`, das die Benutzeranfragen in Anfragen an den UDK umwandelt und die Ergebnisse als dynamisch erzeugte HTLM-Seiten produziert.

Die Einstiegsseite muß in ein Verzeichnis kopiert werden, auf das der HTTP-Server zugreifen kann. Des weiteren muß die Zeile

```
<form method="POST" action="http://wwwipd.ira.uka.de/htbin/udk_zugriff">
```

angepaßt werden. Mit `method` gibt man an, welche Art des Aufrufes an das Programm `udk_zugriff` benutzt werden soll. Abhängig vom installierten HTTP-Server sind dies `POST` oder `GET`. Außerdem muß der Pfad eingetragen werden, unter dem das Programm zugegriffen werden soll.

Das Programm `udk_zugriff` ist als Präcompiler-Applikation geschrieben. Der Code befindet sich in der Datei `udk_zugriff.pc`. Vor dem Übersetzen müssen einige Einstellungen im Code vorgenommen werden. Dies betreffen

- die Aufrufart,
- der Pfad des udk_zugriff Programms und
- die Initialisierung der Datenbank.

Hierbei werden für die ersten beiden Punkte die gleichen Einstellungen gewählt, wie bei der Einstigsseite:

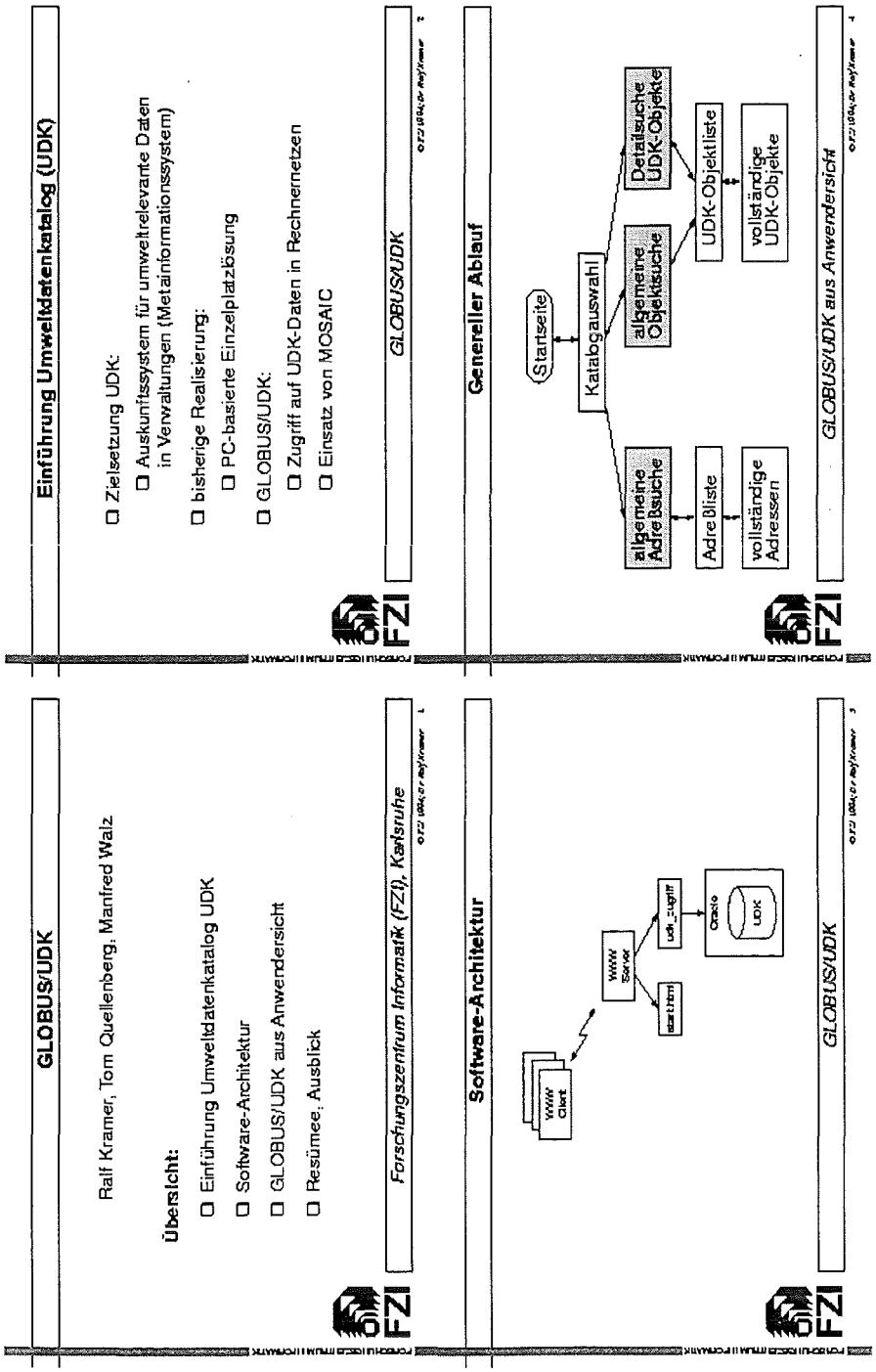
```
char * _METHOD_= "POST";
char * _URL_ = "wwwipd.ira.uka.de/htbin/udk_zugriff";
```

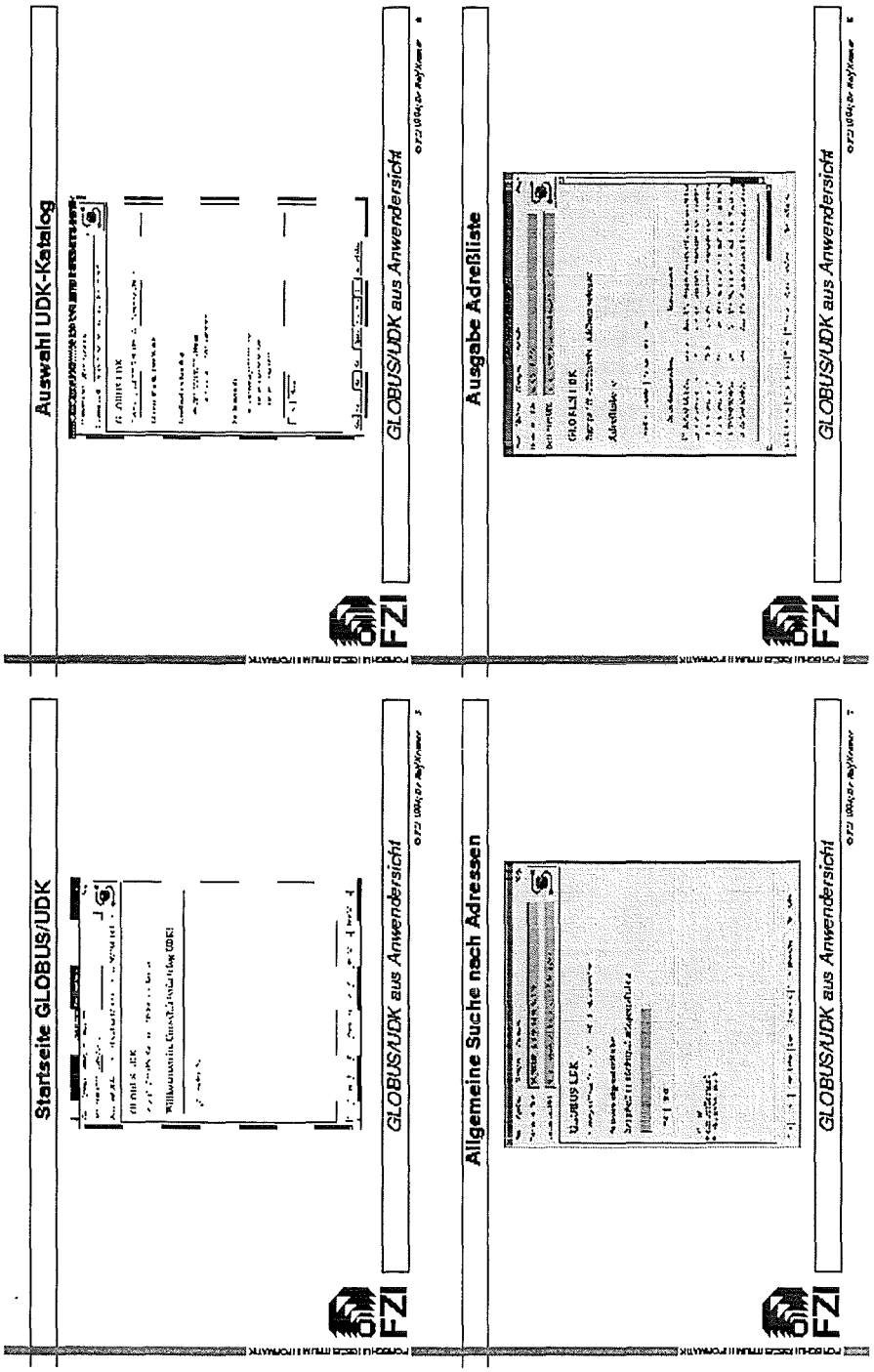
Die Initialisierung der Datenbank erfolgt in der Prozedur `oracle_init`. Dort wird festgelegt, auf welchem Rechner der Datenbankserver läuft, als welcher User mit welchen Password das Programm auf die Datenbank zugreift, usw..

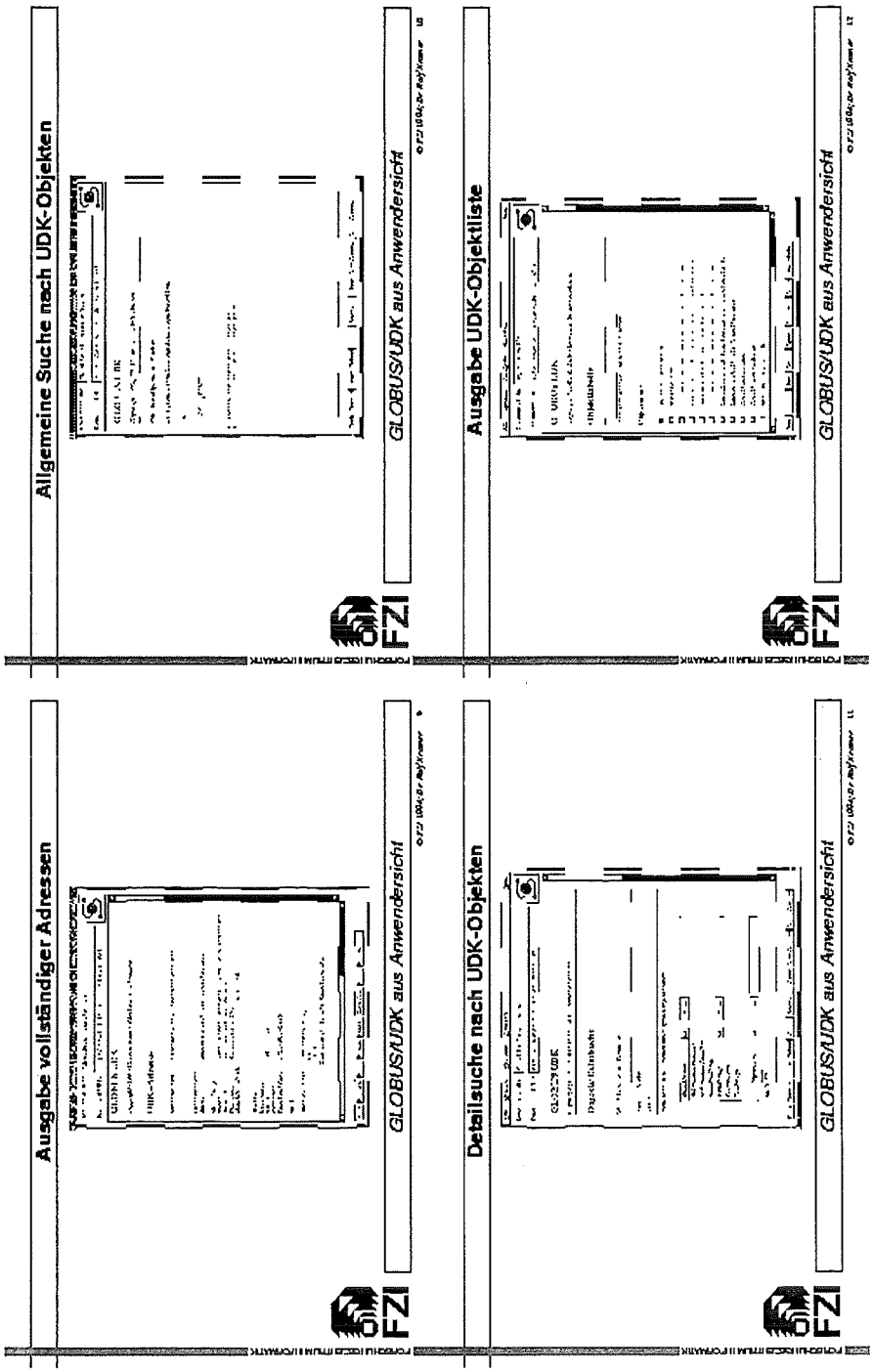
Neben der Datei `udk_zugriff.pc` gibt es noch die Dateien `util.c` und `util.h`. Dieses Modul enthält Routinen für die Auswertung von Parametern, die in den HTML-Formularen zur Spezifizierung der Anfrage an den UDK eingegeben werden. Da nur übliche Standard-Funktionen benutzt wurden, sollte unter Beachtung der Präcompiler-bedingten Restriktionen eine Übersetzung des Systems mit unterschiedlichen C-Compilern möglich sein. Das fertige Programm muß anschließend in das angegebene Verzeichnis kopiert werden. Der UDK kann nun von der Startseite aus zugegriffen werden.

Zu beachten ist ferner, daß andere Seiten als die Einstiegsseite von anderen HTML-Seiten aus nicht direkt zugegriffen werden können, da diese alle dynamisch generiert werden. Ebenfalls ist ein 'reload' einzelner Seiten nicht möglich.

5.7 Vortragsfolien




Auswahl UDK-Katalog
GLOBUS/UDK aus Anwendersicht
FZI
Ausgabe Adreßliste
GLOBUS/UDK aus Anwendersicht
FZI
Startseite GLOBUS/UDK
GLOBUS/UDK aus Anwendersicht
FZI
Allgemeine Suche nach Adressen
GLOBUS/UDK aus Anwendersicht
FZI


Allgemeine Suche nach UDK-Objekten
GLOBUS/UDK aus Anwendersicht
Ausgabe UDK-Objektliste
GLOBUS/UDK aus Anwendersicht
Ausgabe vollständiger Adressen
GLOBUS/UDK aus Anwendersicht
Detailsuche nach UDK-Objekten
GLOBUS/UDK aus Anwendersicht

Ausgabe vollständiger UDK-Objekte

FZI

GLOBUS/UDK aus Anwendersicht

Resümee, Ausblick

sinnvolle weitere Schritte:

- aktives, benutzerkonfigurierbares Auskunftssystem: automatische, gezielte Information
- direkter Zugriff auf die Orignaldaten/-anwendungen in Rechnernetzen
- Überarbeitung UDK-Datenbankentwurf und UDK-Anwendung
- dezentraler UDK: höhere Aktualität, lokale Administration

FZI

GLOBUS/UDK

Resümee, Ausblick

Ergebnisse:

- Realisierung Datenbankzugriff in Weitverkehrsnetzen mittels MOSAIC
- problemadäquate Anfragemöglichkeiten: navigierend, deklarativ

praktische Erfahrungen:

- Anwendungsrealisierung abhängig vom Kommunikationsmodell der Software-Werkzeuge
- Realisierungsaufwand abhängig von Qualität des Datenbankentwurfs

FZI

GLOBUS/UDK

Kapitel 6

Installation der WWW-Basissoftware

6.1 Aufgabenstellung

Die Aufgabe des IKE im Rahmen des Globus-Projektes war die Bereitstellung:

- eines WWW-Servers unter VMS (Version 5.5-2)
- eines Mosaic-Klienten unter VMS und DECwindows-Motif (Version 1.1)

auf dem TN-Cluster des Umweltministeriums und dem Cluster bei der LfU in Karlsruhe.

- Sowie eines Mosaic-Klienten unter MS-Windows (Version 3.1).

Nach Rücksprache mit dem Umweltministerium soll dabei ein PC des IKE eingesetzt werden.

6.2 Softwarekomponenten

Die Software für die VMS-Installation besteht aus folgenden Komonenten:

WWW-Server :

- Version 3.0 (prerelease fuer VMS)
- Bezugsquelle: info.cern.ch

Mosaic : Browser für WWW-Dokumente

- Version 2.4

- Bezugsquelle: info.cern.ch

XV : Viewer für Bilder im GIF, JPEG und TIFF Format

- Version 3.01
- Bezugsquelle: ftp.rus.uni-stuttgart.de

Ghostview/Ghostscript : Viewer für Postscript Bilder und Dokumente

- Version 2.61
- Bezugsquelle: gatekeeper.dec.com

wobei für die Installation von Ghostview/Ghostscript noch der Andrew-Toolkit (Ghostview Widgetset basierend auf X11/R5) installiert werden mußte.

Andrew-Toolkit :

- Version: keine Angabe
- Bezugsquelle: gatekeeper.dec.com

Weiter wurden für die Installation folgende public domain Programme benötigt:

- tar für VMS
- unzip für VMS, bezw. GNU unzip

Die Software für die MS-Windows-Installation besteht aus den Komponenten:

Mosaic :

- Version 2.0A7
- Bezugsquelle: ftp.ncsa.uiuc.edu

Lview : Viewer für Bilder im GIF und JPEG Format

- Version 3.1
- Bezugsquelle: ftp.ncsa.uiuc.edu

Ghostview/Ghostscript :

- Version 1.0/2.61
- Bezugsquelle:

Vorraussetzung für die Installation der oben genannten Mosaic Version war die Installation von:

Win32S : Windows-Erweiterung (32-bit-Version)

- Version 3.2
- Bezugsquelle: ftp.microsoft.com

6.3 Installation

6.3.1 Softwareinstallation unter VMS

Vorgehensweise

Die Installation der VMS spezifischen Software erfolgte zuerst auf dem Entwicklungsrechner an der LfU in Karlsruhe. Die dort übersetzte und zusammengelinkte Software wurde danach auf das TN-Cluster des Umweltministerium überspielt.

6.3.1.1 Installation des WWW-Servers

Die Installtaion des Servers erfolgte in folgenden Schritten:

- Bereitstellung der WWW-Objektbibliothek
- Bereitstellung des WWW-Servers und der benötigten CGI-Programme

Hierzu wurden die vom CERN bereitgestellten Installationsprozeduren verwendet.

Desweiteren mussten die zum Betrieb des WWW-Servers benötigten logischen Namen

WWW_Root , der als „concealed device "auf die Wurzel des Dokumentenverzeichnisses verweist und

HTTP_Dir , der auf das Verzeichnis verweist in sich die Startup- Prozeduren und der Server selbst befinden,

definiert werden. Beide logische Namen müssen systemweit definiert sein. Ferner wurden die in den Startup-Prozeduren enthaltenen Definitionen der sogenannten „foreign commands "bezüglich Plattenname und Directory-Bezeichnung angepaßt.

Die Definition der logischen Namen sollte in der Datei SYLOGICALS.COM, die Startup-Prozedur des Servers in der Datei SYSTARTUP_V5.COM enthalten sein.

Unter WWW_Root:[000000] wurde noch das Verzeichnis SCRATCH angelegt. In ihm legt der Server seine temporären Dateien ab.

6.3.1.2 Installation des Mosaic-Browsers und der „Viewer "

Sowohl für Mosaic, als auch für XV konnten die mitgelieferten Installationsprozeduren verwendet werden. Anpassungen waren nur bezüglich Plattenname und Directory-Bezeichnungen nötig. Einzige Ausnahme war die Installation von Ghostview/Ghostscript. Hier musste, da bei der LfU kein POSIX für VMS verfügbar war, auf diese Programme in aufführbarer Form am IKE erstellt werden.

An den Standardvorgaben wurden KEINE Änderungen vorgenommen, da zum Installationszeitpunkt nicht klar war,

- welche HomePage auf den jeweiligen Servern zur Verfügung steht
- über welche Drucker-Queue gedruckt wird
- und welches Mail-Protokoll verwendet werden wird

Dies stellt jedoch keine Beeinträchtigung der Funktion dar und läßt sich leicht durch Einträge in den jeweiligen Initialiserungs-Dateien anpassen. Die entsprechenden Initialisierungsdateien MOSAIC.DAT_COLOR, bzw. MOSAIC.DAT_MONO finden sich im Installationsverzeichnis von Mosaic.

6.3.1.3 Hinweise für den Betrieb

Neben den zuvor beschriebenen Schritten bei der Installation des Servers müssen für alle im Rahmen von Mosaic verwendeten Programme entsprechende sogenannte „foreign commands "definiert werden. Dies erfolgt durch die Kommandoprozeduren LFU_LOGICALS.COM an der LfU, beziehungsweise durch UM_LOGICALS.COM am UM. Um sicherzustellen, daß diese Definition der „foreign commands "immer erfolgt, wird vorgeschlagen, den Aufruf der entsprechenden Prozdur in der Datei SYLOGIN.COM durchzuführen.

Bemerkungen:

„foreign commands "erlauben es Programme, ähnlich wie bei UNIX,

- nur durch die Angabe des Namens zu starten und
- dem Programm beim Aufruf entsprechende Parameter mitzugeben.

Die Definition eines eines „foreign commands "unter VMS erfolgt durch Angabe der Bezeichnung des Kommandos, gefolgt von der Zeichenkombination :==$ und der Angabe des Pfades und des Programmnamens. So wir beispielweise das „foreign command "für Mosaic am UM wie folgt definiert:

```
$MOSAIC:==$DATEN$DISK:[TOOLS.IKE.MOSAIC]MOSAIC.EXE
```

Der Aufruf von Mosaic dann durch da Kommando:

```
$MOSAIC http://umssv1
```

erfolgen, wobei der Paramter:

```
http://umssv1
```

angibt, daß die HOMEPAGE des Servers „umssv1 "im Verzeichnis WWW_Root:[000000] geladen wird.

6.3.2 Softwareinstallation unter MS-Windows

Da für MS-Windows Mosaic, als WWW-Browser, sowie die Programme zur Visualisierung der verschiedenen Bildformate (Viewer) nur als auführbare Programme vorliegen, war die Installation sehr einfach durchzuführen. Für jedes der einzelnen Pgroamme wurde ein eigenes Verzeichnis erstellt, die Installationsdateien dort entpackt und unter Windows die Programm-Icons in einer eigenen Programmgruppe zur Verfügung gestellt.

Für die Installation der 32-Bit Erweiterung von Windows wurde die von Microsoft mitgelieferte Setup-Prozedur verwendewt.

Die Netzwerkeinbindung erfolgte nicht prototypisch im Rahmen eines zur Verfügung PC-Netzes sondern als einzelner TCP/IP-Knoten. Dabei war zu breücksichtigen, daß die Anzahl der „public domain "zur Verfügung stehenden paketorientierten Netzwerkkartentreiber gering ist und nur spezielle Ethernet-Karten verwendet werden konnten.

Für einen generellen Einsatz von Mosaic unter MS-Windows wird daher empfohlen, die TCP/IP-Netzwerkfunktionalität im Rahmen eines bestehenden PC-Netzwerks zu realisieren.

Hierbei sei auf Pathworks als PC-Netz verwiesen, insbesonders deshalb, da bei Version V5.1 sowohl Mosaic, als auch die 32-Bit Erweiterung von Windows im Lieferumfang von Pathworks enthalten sind.

6.4 Einbindung der Globus-Dokumente

Im Rahmen der Installation wird vom IKE eine sogenannte HomePage für das Umweltministerium erstellt, die es erlaubt über die Integral-Oberfläche auf die Globusdokumente zuzugreifen.

Kapitel 7

Zusammenfassung und Ausblick

Als Ergebnis des Projekts GLOBUS konnte gezeigt werden, daß unter einer einheitlichen Benutzeroberfläche Umweltberichte und Sachdaten verschiedenster Provenienz zugänglich sind. Als Werkzeugkasten hat sich die WWW-Architektur mit netztransparenten Hyperlinkverbindungen bewährt.

Mosaic, httpd und andere Werkzeuge im Umfeld von WWW bieten ein sehr mächtige, kostengünstige und leicht benutzbare Plattform zur Integration verschiedenster Dokumente und Dienste in einem Umweltinformationssystem.

Aufbauend auf die erreichten Ergebnisse sind weitere Punkte wünschenswert:

1. *Grundlegende Strukturierung des Umweltberichtswesens*

   Dokumente sollten in herstellerunabhängigen Markup-Sprachen strukturiert werden, sodaß ein leichterer Übergang nach HTML gegeben ist. Struktur und grafisches Layout müssen strikt getrennt werden. Das Potential von SGML und eventuell anderer Konzepte (RTF, TeX, Adobe Destiller usw.) sollte untersucht werden.

2. *Dezentrale Metadatenbestände*

   WWW bietet explizit den netztransparenten Zugriff auf Dienste und Daten. Es sollte geprüft werden, ob nicht verteilte Dokumente und Kataloge eine zweckmäßige Alternative zu zentralen Beständen sind. Die Administration wäre vereinfacht, die Aktualität verbessert, die Verantwortung verteilt.

3. *Direkter Zugang zu Originaldaten*

   WWW bietet über ein Gatewaykonzept Zugang zu unterschiedlichsten Daten und Diensten (Relationale Datenbanken, Texte, Dateitransfer, Karten). Neben dem Zugriff auf Metadaten ist auch ein Zugang zu Originaldaten möglich.

Anhang A

Mitarbeiterinnen und Mitarbeiter

An diesem Projekt haben mitgearbeitet:

- Joachim Wiesel, IPF, Projektleitung
- Werner Weisbrich, IPF
- Claus Hofmann, IPF
- Michael Böse, IPF
- Klaus-Jürgen Schilling, IPF
- Ralf Kramer, FZI
- Tom Quellenberg, FZI
- Khaldoun Ateyeh, FZI
- Oliver Scheffczyk, FZI
- Manfred Walz, FZI
- Fritz Schmidt, IKE
- Hr. Scheuermann, IKE
- Frau Kübler, IKE
- Fritz Riekert, FAW
- Frau Gaul, FAW
- Manfred Müller, LfU
- Horst Spandl, LfU
- Inge Henning, UM

Anhang B

HTML-Primer

B.1 A Beginner's Guide to HTML

This is a primer for producing documents in HTML, the markup language used by the World Wide Web.

- Acronym Expansion
- What This Primer Doesn't Cover
- Creating HTML Documents
  - The Minimal HTML Document
  - Basic Markup Tags
    - Titles
    - Headings
    - Paragraphs
  - Linking to Other Documents
    - Relative Links Versus Absolute Pathnames
    - Uniform Resource Locator
    - Anchors to Specific Sections in Other Documents
    - Anchors to Specific Sections Within the Current Document
- Additional Markup Tags
  - Lists
    - Unnumbered Lists
    - Numbered Lists
    - Definition Lists
    - Nested Lists
  - Preformatted Text

  - Extended Quotes
  - Addresses
- Character Formatting
  - Physical Versus Logical: Use Logical Tags When Possible
    * Logical Styles
    * Physical Styles
  - Using Character Tags
  - Special Characters
    * Escape Sequences
    * Forced Line Breaks
    * Horizontal Rules
- In-line Images
  - Alternate Text for Viewers That Can't Display Images
- External Images, Sounds, and Animations
- Troubleshooting
  - Avoid Overlapping Tags
  - Embed Anchors and Character Tags, But Not Anything Else
  - Check Your Links
- A Longer Example
- For More Information
  - Fill-out Forms
  - Style Guides
  - Other Introductory Documents
  - Additional References

B.1.1 Acronym Expansion

WWW World Wide Web (or Web, for short).

SGML Standard Generalized Markup Language – this is a standard for describing markup languages.

DTD

Document Type Definition – this is a specific markup language, written using SGML.

HTML

HyperText Markup Language – HTML is a SGML DTD. In practical terms, HTML is a collection of styles (indicated by markup tags) that define the various components of a World Wide Web document.

B.1.2 What This Primer Doesn't Cover

This primer assumes that you have:

- at least a passing knowledge of how to use NCSA Mosaic or some other Web browser
- a general understanding of how Web servers and client browsers work
- access to a Web server for which you would like to produce HTML documents, or that you wish to produce HTML documents for personal use

B.1.3 Creating HTML Documents

HTML documents are in plain (also known as ASCII) text format and can be created using any text editor (e.g., Emacs or vi on UNIX machines). A couple of Web browsers (tkWWW for X Window System machines and CERN's Web browser for NeXT computers) include rudimentary HTML editors in a WYSIWYG environment. There are also some WYSIWIG editors available now (e.g. HotMetal for Sun Sparcstations, HTML Edit for Macintoshes). You may wish to try one of them first before delving into the details of HTML. *You can preview a document in progress with NCSA Mosaic (and some other Web browsers). Open it with the* **Open Local** *command under the* **File***menu.*

After you edit the source HTML file, save the changes. Return to NCSA Mosaic and **Reload***the document. The changes are reflected in the on-screen display.*

B.1.3.1 The Minimal HTML Document

Here is a bare-bones example of HTML: <TITLE>The simplest HTML example</TITLE> <H1>This is a level-one heading</H1> Welcome to the world of HTML. This is one paragraph.<P> And this is a second.<P> Click here to see the formatted version of the example.

HTML uses markup tags to tell the Web browser how to display the text. The above example uses:

- the <TITLE> tag (and corresponding </TITLE> tag), which specifies the title of the document
- the <H1> header tag (and corresponding </H1>)

- the <P> paragraph-separator tag

HTML tags consist of a left angle bracket (<), (a "less than" symbol to mathematicians), followed by name of the tag and closed by a right angular bracket (>). Tags are usually paired, e.g. <H1> and </H1>. The ending tag looks just like the starting tag except a slash (/) precedes the text within the brackets. In the example, <H1> tells the Web browser to start formatting a level-one heading; </H1> tells the browser that the heading is complete.

The primary exception to the pairing rule is the <P> tag. There is no such thing as </P>.

NOTE:*HTML is not case sensitive.* <title>*is equivalent to* <TITLE>*or* <TiTlE>.

Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it just ignores it.

B.1.3.2 Basic Markup Tags

Title Every HTML document should have a title. A title is generally displayed separately from the document and is used primarily for document identification in other contexts (e.g., a WAIS search). Choose about half a dozen words that describe the document's purpose. *In the X Window System and Microsoft Windows versions of NCSA Mosaic, the* **Document Title***field is at the top of the screen just below the pulldown menus. In NCSA Mosaic for Macintosh, text tagged as* <TITLE>*appears as the window title.*

Headings HTML has six levels of headings, numbered 1 through 6, with 1 being the most prominent. Headings are displayed in larger and/or bolder fonts than normal body text. The first heading in each document should be tagged <H1>. The syntax of the heading tag is:

<H*y*>*Text of heading*</H*y*>

where y is a number between 1 and 6 specifying the level of the heading.

For example, the coding for the "Headings" section heading above is <H3>Headings</H3>

Title versus first heading In many documents, the first heading is identical to the title. For multipart documents, the text of the first heading should be suitable for a reader who is already browsing related information (e.g., a chapter title), while the title tag should identify the document in a wider context (e.g., include both the book title and the chapter title, although this can sometimes become overly long).

Paragraphs Unlike documents in most word processors, carriage returns in HTML files aren't significant. Word wrapping can occur at any point in your source file, and multiple spaces are collapsed into a single space. (There are couple of exceptions; space following a <P> or <H*y*> tag, for example, is

ignored.) Notice that in the bare-bones example, the first paragraph is coded as Welcome to HTML. This is the first paragraph. <P>

In the source file, there is a line break between the sentences. A Web browser ignores this line break and starts a new paragraph only when it reaches a <P> tag.

Important: You must separate paragraphs with <P>. The browser ignores any indentations or blank lines in the source text. HTML relies almost entirely on the tags for formatting instructions, and without the <P> tags, the document becomes one large paragraph. (The exception is text tagged as "preformatted," which is explained below.) For instance, the following would produce identical output as the first bare-bones HTML example: <TITLE>The simplest HTML example</TITLE><H1>This is a level one heading</H1>Welcome to the world of HTML. This is one paragraph.<P>And this is a second.<P>

However, to preserve readability in HTML files, headings should be on separate lines, and paragraphs should be separated by blank lines (in addition to the <P> tags). *NCSA Mosaic handles <P> by ending the current paragraph and inserting a blank line.*

In HTML+, a successor to HTML currently in development, <P> becomes a "container" of text, just as the text of a level-one heading is "contained" within<H1> ... </H1>: <P> This is a paragraph in HTML+. </P>

The difference is that the </P> closing tag can always be omitted. (That is, if a browser sees a <P>, it knows that there must be an implied </P> to end the previous paragraph.) In other words, in HTML+, <P> is a beginning-of-paragraph marker.

The advantage of this change is that you will be able to specify formatting options for a paragraph. For example, in HTML+, you will be able to center a paragraph by coding <P ALIGN=CENTER> This is a centered paragraph. This is HTML+, so you can't do it yet.

This change won't effect any documents you write now, and they will continue to look just the same with HTML+ browsers.

B.1.3.3 Linking to Other Documents

The chief power of HTML comes from its ability to link regions of text (and also images) to another document. The browser highlights these regions (usually with color and/or underlines) to indicate that they are hypertext links (often shortened to *hyperlinks* or simply *links*).

HTML's single hypertext-related tag is <A>, which stands for *anchor*. To include an anchor in your document:

1. Start the anchor with <A . (There's a space after the A.)

2. Specify the document that's being pointed to by entering the parameter HREF="*filename*" followed by a closing right angle bracket: >

3. Enter the text that will serve as the hypertext link in the current document.

4. Enter the ending anchor tag: `</A>`.

Here is an sample hypertext reference: <A HREF=EF="MaineStats.html">Maine</A>

This entry makes the word "Maine" the hyperlink to the document `MaineStats.html`, which is in the same directory as the first document. You can link to documents in other directories by specifying the *relative path* from the current document to the linked document. For example, a link to a file `NJStats.html` located in the subdirectory `AtlanticStates` would be: <A HREF=EF="AtlanticStates/NJStats.html">New Jersey</A>

These are called *relative links.* You can also use the absolute pathname of the file if you wish. Pathnames use the standard UNIX syntax.

Relative Links Versus Absolute Pathnames In general, you should use relative links, because

1. You have less to type.

2. It's easier to move a group of documents to another location, because the relative path names will still be valid.

However, use absolute pathnames when linking to documents that are not directly related. For example, consider a group of documents that comprise a user manual. Links within this group should be relative links. Links to other documents (perhaps a reference to related software) should use full path names. This way, if you move the user manual to a different directory, none of the links would have to be updated.

Uniform Resource Locator The World Wide Web uses Uniform Resource Locators (URLs) to specify the location of files on other servers. A URL includes the type of resource being accessed (e.g., gopher, WAIS), the address of the server, and the location of the file. The syntax is:

scheme`://`*host.domain*`[:`*port*`]/`*path/filename*

where *scheme* is one of

`file`

a file on your local system, or a file on an anonymous FTP server

`http` a file on a World Wide Web server

`gopher` a file on a Gopher server

`WAIS` a file on a WAIS server

`news` an Usenet newsgroup

`telnet` a connection to a Telnet-based service

The *port* number can generally be omitted. (That means unless someone tells you otherwise, leave it out.)

For example, to include a link to this primer in your document, you would use <A HREF = = "http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html"> NCSA's Beginner's Guide to HTML</A>

This would make the text "NCSA's Beginner's Guide to HTML" a hyperlink to this document.

For more information on URLs, look at

- WWW NAMES AND ADDRESSES, URIs, URLs, URNs, written by people at CERN
- A BEGINNER'S GUIDE TO URLs, located on the NCSA Mosaic **Help** menu

Links to Specific Sections in Other Documents Anchors can also be used to move to a particular section in a document. Suppose you wish to set a link from document A and a specific section in document B. (Call this file `documentB.html`.) First you need to set up a *named anchor* in document B. For example, to set up an anchor named "Jabberwocky" to document B, enter Here's <A NAME = = "Jabberwocky">some text

Now when you create the link in document A, include not only the filename, but also the named anchor, separated by a hash mark (#). This is my <A HREF = = "documentB.html#Jabberwocky">link</A> to document B.

Now clicking on the word "link" in document A sends the reader directly to the words "some text" in document B.

Links to Specific Sections Within the Current Document The technique is exactly the same except the filename is omitted.

For example, to link to the Jabberwocky anchor from within the same file (Document B), use This is <A HREF = = "#Jabberwocky">Jabberwocky link</A> from within Document B.

B.1.4 Additional Markup Tags

The preceding is sufficient to produce simple HTML documents. For more complex documents, HTML has tags for several types of lists, preformatted sections, extended quotations, character formatting, and other items.

B.1.4.1 Lists

HTML supports unnumbered, numbered, and definition lists.

Unnumbered Lists To make an unnumbered list,

1. Start with an opening list <UL> tag.
2. Enter the <LI> tag followed by the individual item. (No closing </LI> tag is needed.)
3. End with a closing list </UL> tag.

Below an example two-item list: <UL> <LI> apples <LI> bananas </UL>
The output is:

- apples
- bananas

The <LI> items can contain multiple paragraphs. Just separate the paragraphs with the <P> paragraph tags.

Numbered Lists A numbered list (also called an ordered list, from which the tag name derives) is identical to an unnumbered list, except it uses <OL> instead of <UL>. The items are tagged using the same <LI> tag. The following HTML code <OL> <LI> oranges <LI> peaches <LI> grapes </OL>
produces this formatted output:

1. oranges
2. peaches
3. grapes

Definition Lists A definition list usually consists of alternating a term (abbreviated as DT) and a definition (abbreviated as DD). Web browsers generally format the definition on a new line.

The following is an example of a definition list: <DL> <DT> NCSA <DD> NCSA, the National Center for Supercomputing Applications, is located on the campus of the University of Illinois at Urbana-Champaign. NCSA is one of the participants in the National MetaCenter for Computational Science and Engineering. <DT> Cornell Theory Center <DD> CTC is located on the campus of Cornell University in Ithaca, New York. CTC is another participant in the National MetaCenter for Computational Science and Engineering. </DL>
The output looks like:

NCSA NCSA, the National Center for Supercomputing Applications, is located on the campus of the University of Illinois at Urbana-Champaign. NCSA is one of the participants in the National MetaCenter for Computational Science and Engineering.

Cornell Theory Center CTC is located on the campus of Cornell University in Ithaca, New York. CTC is another participant in the National MetaCenter for Computational Science and Engineering.

The <DT> and<DD> entries can contain multiple paragraphs (separated by <P> paragraph tags), lists, or other definition information.

Nested Lists Lists can be arbitrarily nested, although in practice you probably should limit the nesting to three levels. You can also have a number of paragraphs, each containing a nested list, in a single list item.

An example nested list: <UL> <LI> A few New England states: <UL> <LI> Vermont <LI> New Hampshire </UL> <LI> One Midwestern state: <UL> <LI> Michigan </UL> </UL>

The nested list is displayed as

- A few New England states:
  - Vermont
  - New Hampshire
- One Midwestern state:
  - Michigan

B.1.4.2 Preformatted Text

Use the<PRE> tag (which stands for "preformatted") to generate text in a fixed-width font and cause spaces, new lines, and tabs to be significant. (That is, multiple spaces are displayed as multiple spaces, and lines break in the same locations as in the source HTML file.) This is useful for program listings. For example, the following lines <PRE> #!/bin/csh cd $SCR cfs get mysrc.f:mycfsdir/mysrc.f cfs get myinfile:mycfsdir/myinfile fc -02 -o mya.out mysrc.f mya.out cfs save myoutfile:mycfsdir/myoutfile rm * </PRE>

display as #!/bin/csh cd $SCR cfs get mysrc.f:mycfsdir/mysrc.f cfs get myinfile:mycfsdir/myinfile fc -02 -o mya.out mysrc.f mya.out cfs save myoutfile:mycfsdir/myoutfile rm *

Hyperlinks can be used within <PRE> sections. You should avoid using other HTML tags within <PRE> sections, however.

Note that because <, >, and & have special meaning in HTML, you have to use their escape sequences (`<`, `>`, and `&`, respectively) to enter these characters. See the section Special Characters for more information.

B.1.4.3 Extended Quotations

Use the <BLOCKQUOTE> tag to include quotations in a separate block on the screen. Most browsers generally indent to separate it from surrounding text.

An example: <BLOCKQUOTE> I still have a dream. It is a dream deeply rooted in the American dream. <P> I have a dream that one day this nation will rise up and live out the true meaning of its creed. We hold these truths to be self-evident that all men are created equal. <P> </BLOCKQUOTE>

The result is:

I still have a dream. It is a dream deeply rooted in the American dream.

I have a dream that one day this nation will rise up and live out the true meaning of its creed. We hold these truths to be self-evident that all men are created equal.

B.1.4.4 Addresses

The <ADDRESS> tag is generally used to specify the author of a document and a means of contacting the author (e.g., an email address). This is usually the last item in a file.

For example, the last line of the online version of this guide is <ADDRESS> A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu </ADDRESS>

The result is A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu

NOTE:<ADDRESS> is *not* used for postal addresses. See "Forced Line Breaks" on page 10 to see how to format postal addresses.

B.1.5 Character Formatting

You can code individual words or sentences with special styles. There are two types of styles: logical and physical. *Logical styles* tag text according to its meaning, while *physical styles* specify the specific appearance of a section. For example, in the preceding sentence, the words "logical styles" was tagged as a "definition." The same effect (formatting those words in italics), could have been achieved via a different tag that specifies merely "put these words in italics."

B.1.5.1 Physical Versus Logical: Use Logical Styles When Possible

If physical and logical styles produce the same result on the screen, why are there both? We devolve, for a couple of paragraphs, into the philosophy of SGML, which can be summed in a Zen-like mantra: "Trust your browser."

In the ideal SGML universe, content is divorced from presentation. Thus, SGML tags a level-one heading as a level-one heading, but does not specify that the level-one heading should be, for instance, 24-point bold Times centered on the top of a page. The advantage of this approach (it's similar in concept to style sheets in many word processors) is that if you decide to

change level-one headings to be 20-point left-justified Helvetica, all you have to do is change the definition of the level-one heading in the presentation device (i.e., your World Wide Web browser).

The other advantage of logical tags is that they help enforce consistency in your documents. It's easier to tag something as <H1> than to remember that level-one headings are 24-point bold Times or whatever. The same is true for character styles. For example, consider the <STRONG> tag. Most browsers render it in bold text. However, it is possible that a reader would prefer that these sections be displayed in red instead. Logical styles offer this flexibility.

Logical Styles

<DFN> for a word being defined. Typically displayed in italics. (*NCSA Mosaic* is a World Wide Web browser.)

<EM> for emphasis. Typically displayed in italics. (*Watch out for pickpockets.*)

<CITE> for titles of books, films, etc. Typically displayed in italics. (A BEGINNER'S GUIDE TO HTML)

<CODE> for snippets of computer code. Displayed in a fixed-width font. (The <stdio.h> header file)

<KBD> for user keyboard entry. Should be displayed in a bold fixed-width font, but many browsers render it in the plain fixed-width font. (Enter **passwd** to change your password.)

<SAMP> for computer status messages. Displayed in a fixed-width font. (`Segmentation fault: Core dumped.`)

<STRONG> for strong emphasis. Typically displayed in bold. (**Important**)

<VAR> for a "metasyntactic" variable, where the user is to replace the variable with a specific instance. Typically displayed in italics. (**rm***filename* deletes the file.)

Physical Styles

<B> bold text

<I> italic text

<TT> typewriter text, e.g. fixed-width font.

B.1.5.2 Using Character Tags

To apply a character style,

1. Start with <*tag*>, where*tag* is the desired character formatting tag, to indicate the beginning of the tagged text.
2. Enter the tagged text.
3. End the passage with </*tag*>.

B.1.5.3 Special Characters

Escape Sequences Four characters of the ASCII character set – the left angle bracket (<), the right angle bracket (>), the ampersand (&) and the double quote (le quote (") – have special meaning within HTML and therefore cannot be used "as is" in text. (The angle brackets are used to indicate the beginning and end of HTML tags, and the ampersand is used to indicate the beginning of an escape sequence.)

To use one of these characters in an HTML document, you must enter its *escape sequence* instead:

`<` the escape sequence for <

`>` the escape sequence for >

`&` the escape sequence for &

`&quot;` the escape sequence for "

Additional escape sequences support accented characters. For example:

`&ouml;` the escape sequence for a lowercase o with an umlaut: ö

`&ntilde;` the escape sequence for a lowercase n with an tilde: ñ

`&Egrave;` the escape sequence for an uppercase E with a grave accent: È

A full list of supported characters can be found at CERN.

NOTE: Unlike the rest of HTML, the escape sequences are case sensitive. You cannot, for instance, use < instead of <.

Forced Line Breaks The <BR> tag forces a line break with no extra space between lines. (By contrast, most browsers format the <P> paragraph tag with an additional blank line to more clearly indicate the beginning the new paragraph.)

One use of <BR> is in formatting addresses: National Center for Supercomputing Applications
 605 East Springfield Avenue
 Champaign, Illinois 61820-5518

Horizontal Rules The <HR> tag produces a horizontal line the width of the browser window.

B.1.6 In-line Images

Most Web browsers can display in-line images (that is, images next to text) that are in X Bitmap (XBM) or GIF format. Each image takes time to process and slows down the initial display of the document, so generally you should not include too many or overly large images.

To include an in-line image, use <IMG SRC=*image_URL*>

where *image_URL* is the URL of the image file. The syntax for IMG SRC URLs is identical to that used in an anchor HREF. If the image file is a GIF file, then the filename part of *image_URL* **must** end with .gif. Filenames of X Bitmap images must end with .xbm.

By default the bottom of an image is aligned with the text as shown in this paragraph.

Add the ALIGN=TOP option if you want the browser to align adjacent text with the top of the image as shown in this paragraph. The full in-line image tag with the top alignment is: <IMG ALIGN=top SRC=*image_URL*>

ALIGN=MIDDLE aligns the text with the center of the image.

B.1.6.1 Alternate Text for Browsers That Can't Display Images

Some World Wide Web browsers, primarily those that run on VT100 terminals, cannot display images. The ALT option allows you to specify text to be displayed when an image cannot be. For example: <IMG SRC = = "UpArrow.gif" ALT = "Up">

where UpArrow.gif is the picture of an upward pointing arrow. With NCSA Mosaic and other graphics-capable viewers, the user sees the up arrow graphic. With a VT100 browser, such as lynx, the user sees the word "Up."

B.1.7 External Images, Sounds, and Animations

You may want to have an image open as a separate document when a user activates a link on either a word or a smaller, in-line version of the image included in your document. This is considered an external image and is useful if you do not wish to slow down the loading of the main document with large in-line images.

To include a reference to an external image, use <A HREF = *image_URL*>link anchor</A>

Use the same syntax is for links to external animations and sounds. The only difference is the file extension of the linked file. For example,

```
<A HREF = = &amp;quot;QuickTimeMovie.mov&amp;quot;&amp;gt;link
anchor&amp;lt;/A&amp;gt;
```

specifies a link to a QuickTime movie. Some common file types and their extensions are:

File Type Extension

Plain text `.txt`

HTML document `.html`

GIF image `.gif`

TIFF image `.tiff`

XBM bitmap image `.xbm`

JPEG image `.jpg` or `.jpeg`

PostScript file `.ps`

AIFF sound `.aiff`

AU sound `.au`

QuickTime movie `.mov`

MPEG movie `.mpeg` or `.mpg`

Make sure your intended audience has the necessary viewers. Most UNIX workstations, for instance, cannot view QuickTime movies.

B.1.8 Troubleshooting

B.1.8.1 Avoid Overlapping Tags

Consider this snippet of HTML: <B>This is an example of <DFN>overlapping</B> HTML tags.</DFN>

The word "overlapping" is contained within both the <B> and <DFN> tags. How does the browser format it? You won't know until you look, and different browsers will likely react differently. In general, avoid overlapping tags.

B.1.8.2 Embed Anchors and Character Tags, But Nothing Else

It is acceptable to embed anchors within another HTML element: <H1><A HREF = A HREF = "Destination.html">My heading</A></H1>

Do not embed a heading or another HTML element within an anchor: <A HREF = = "Destination.html"> <H1>My heading</H1> </A>

Although most browsers currently handle this example, it is forbidden by the official HTML and HTML+ specifications, and will not work with future browsers.

Character tags modify the appearance of other tags: <UL><LI><B>A bold list item</B> <UL> <LI><I>An italic list item</I> </UL>

However, avoid embedding other types of HTML element tags. For example, it is tempting to embed a heading within a list, in order to make the font size larger: <UL><LI><H1>A large heading</H1> <UL> <LI><H2>Something slightly smaller</H2> </UL>

Although some browsers, such as NCSA Mosaic for the X Window System, format this construct quite nicely, it is unpredictable (because it is undefined) what other browsers will do. For compatibility with all browsers, avoid these kinds of constructs.

What's the difference between embedding a <B> within a <LI> tag as opposed to embedding a <H1> within a <LI>? This is again a question of SGML. The semantic meaning of <H1> is that it's the main heading of a document and that it should be followed by the content of the document.Thus it doesn't make sense to find a <H1> within a list.

Character formatting tags also are generally not additive. You might expect that <B><I>some text</I></B>

would produce bold-italic text. On some browsers it does; other browsers interpret only the innermost tag (here, the italics).

B.1.8.3 Check Your Links

When an <IMG> tag points at an image that does not exist, a dummy image is substituted. When this happens, make sure that the referenced image does in fact exist, that the hyperlink has the correct information in the URL, and that the file permission is set appropriately (world-readable).

B.1.9 A Longer Example

Here is a longer example of an HTML document: <HEAD> <TITLE>A Longer Example</TITLE> </HEAD> <BODY> <H1>A Longer Example</H1> This is a simple HTML document. This is the first paragraph. <P> This is the second paragraph, which shows special effects. This is a word in <I>italics</I>. This is a word in <B>bold</B>. Here is an in-lined GIF image: <IMG SRC = <B>bold</B>. Here is an in-lined GIF image: <IMG SRC = "myimage.gif">. <P> This is the third paragraph, which demonstrates links. Here is a hypertext link from the word <A HREF = "subdir/myfile.html">foo</A> to a document called "subdir/myfile.html". (If you try to follow this link, you will get an error screen.) <P> <H2>A second-level header</H2> Here is a section of text that should display as a

fixed-width font: <P> <PRE> On the stiff twig up there Hunches a wet black rook Arranging and rearranging its feathers in the rain ... </PRE> This is a unordered list with two items: <P> <UL> <LI> cranberries <LI> blueberries </UL> This is the end of my example document. <P> <ADDRESS>Me (me@mycomputer.univ.edu)</ADDRESS> </BODY> Click here to see the formatted version.

In addition to tags already discussed, this example also uses the <HEAD> ... </HEAD> and <BODY> ... </BODY> tags, which separate the document into introductory information about the document and the main text of the document. These tags don't change the appearance of the formatted document at all, but are useful for several purposes (for example, NCSA Mosaic for Macintosh 2.0, for example, allows you to browse just the header portion of document before deciding whether to download the rest), and it is recommended that you use these tags.

B.1.10 For More Information

This guide is only an introduction to HTML and not a comprehensive reference. Below are additional sources of information.

B.1.10.1 Fill-out Forms

One major feature not discussed here is fill-out forms, which allows users to return information to the World Wide Web server. For information on fill-out forms, look at this Fill-out Forms Overview

B.1.10.2 Style Guides

The following offer advice on how to write "good" HTML:

- COMPOSING GOOD HTML
- CERN's style guide for online hypert

B.1.10.3 Other Introductory Documents

These cover similar information as this guide:

- HOW TO WRITE HTML FILES
- INTRODUCTION TO HTML

B.1.10.4 Additional References

- THE HTML QUICK REFERENCE GUIDE, which provides a comprehensive listing of HTML codes

- The official HTML specification
- A description of SGML, the Standard Generalized Markup Language
- Dan Connolly's HTML DESIGN NOTEBOOK. Dan Connolly is one of the originators of HTML.

National Center for Supercomputing Applications / pubs@ncsa.uiuc.edu

Anhang C

TCL-Programmcode cvp2html

```
#!/usr/local/bin/tcl
#
# Konvertierungsprogramm, dass Corel Ventura Publisher Kapitel
# nach HTML uebersetzt
#-------------------------------------------------------------------------
#
# globale Variablen
#
# param()     : Array, in dem die Eingabevariablen stehen
#   cfname          : Dateiname der Steuerdatei
#   htmldir         : Basisdirectory der HTML Dateien
#   sourcedir       : Basisdirectory der CVP Dateien
#   destdir         : Basisdirectory der konvertierten Dateien
#   showdeltag      : Tags, die entfernt werden, anzeigen
#   showdelchar     : Zeichen, die entfernt werden, anzeigen
#
# control()   : Array, in dem die Steuerinformation gespeichert
wird
#    docfname       : Liste von filenamen der zu konvertierenden
Textdateien
#    pictfname      : Liste von Listen von Filenamen der
Abbildungen
#    subtitletag    : Liste von Kurzbeschreibungen fuer
Abbildungen;
#    blocknr        : Liste die angibt, wie haeufig die
zugehoerige
#                     Kurzbeschreibung fuer Bilduntertitel
vorkommt
#    pictnum        : Liste der Anzahl der Abbildungen pro
Kapitel
#
# status()    : Array, in dem Parserzustaende gespeichert werden
```

```
#    line          : aktuelle Zeile im sourceText
#    attag         : automat im Modus "@tag"
#    text          : automat im Modus "text"
#    file          : aktuelles zu uebersetzendes TextFile
#    chapternum    : aktuelle (reale) Kapitelnummer
#    immagepage    : aktuelle Grafikseitenummer
#    depictnum     : aktuelle Anzahl von Abbildungen
#
# statussub(): Array, wie status, fuer Untertitel der Bilder
#
# textbuf1        : Text-Puffervariable
# textbuf2        :       "
# body            : Variable in der der Uebersetzte HTML text
geschrieben wird
# html()      : Array, in dem HTML-Definitionen gespeichert
sind
# char()      : Array, in dem Zeichen uebersetzungs Konstanten
gespeichert sind

#-------------------------------------------------------------------
# InitHtmlDefs {} Initialisiert HTML Tag Definitionen
#-------------------------------------------------------------------
#
proc InitHtmlDefs {} {

  global html

  set html(title_start)        "\n<H1>"
  set html(title_stop)         "</H1>\n"
  set html(subhead1_start)     "\n<h2>"
  set html(subhead1_stop)      "</h2>\n"
  set html(bold_start)         "<b>"
  set html(bold_stop)          "</b>"
  set html(italic_start)       "<i>"
  set html(italic_stop)        "</i>"
  set html(quote_start)        "\n<blockquote>\n"
  set html(quote_stop)         "\n</blockquote>\n"
  set html(paragraph)          "\n<p>\n"
  set html(reference_start)    "<cite>"
  set html(reference_stop)     "</cite>"
}

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

```
# InitCharTrans {} Initialisiert einige Zeichen fuer
Zeichenanpassung
#-------------------------------------------------------------------------
#
proc InitCharTrans {} {

  global char

  # HTML-ESCAPE Sequenzen
----------------------------------------
  #
#  set char(132)   "&#228;"        ; # kleines a"
#  set char(142)   "&#196;"        ; # grosses A"
#  set char(129)   "&#252;"        ; # kleines u"
#  set char(154)   "&#220;"        ; # groesses U"
#  set char(148)   "&#246;"        ; # kleines o"
#  set char(153)   "&#214;"        ; # grosses O"
#  set char(217)   "&#223;"        ; # scharfes s (sz)
#  set char(169)   "'"             ; # Anfuehrungszeichen
oeffnen
#  set char(170)   "`"             ; # Anfuehrungszeichen
schliessen
#  set char(M^)    "^"             ; # Sequenz fuer "naechstes
Zeichen
                                    # hochstellen " simmulieren

  # 8-Bit ISO Zeichensatz
  #
  set char(132)   ""              ; # kleines a"
  set char(142)   ""              ; # grosses A"
  set char(129)   ""              ; # kleines u"
  set char(154)   ""              ; # groesses U"
  set char(148)   ""              ; # kleines o"
  set char(153)   ""              ; # grosses O"
  set char(217)   ""              ; # scharfes s (sz)
  set char(169)   "'"             ; # Anfuehrungszeichen
oeffnen
  set char(170)   "`"             ; # Anfuehrungszeichen
schliessen
  set char(M^)    "^"             ; # Sequenz fuer "naechstes
Zeichen
                                    # hochstellen " simmulieren
}
```

```
#-------------------------------------------------------------------
# PrintHelp {} Routine, die einen Hilfetext fuer den Aufruf
dieses Programms
#              ausgibt und die Programmausfuehrung beendet
#-------------------------------------------------------------------
#
proc PrintHelp {} {

  puts stderr "\ncvp2html - converts Corel Ventura Publisher
documents to HTML"
  puts stderr {Usage: cvp2html -cf <fname> [-hd <dir>] [-bd
<dir>] [-dd <dir>] [-debug <opt>]}
  puts stderr " -cf <filename>   : filepath/name of control
file"
  puts stderr " -hd <htmldir>    : base directory of HTML
files"
  puts stderr " -sd <sourcedir>  : base directory of CVP files"
  puts stderr " -dd <destdir>    : base directory of converted
files"
  puts stderr " -debug <option>  : debug-mode : option:"
  puts stderr "                          @tag - show removed
@tags"
  puts stderr "                          char - show removed
chars"
  puts stderr "                          all  - show both"
  puts stderr "\n"
  exit
}

#-------------------------------------------------------------------
# InitParam {} Routine, die die Eingabeparameter initialisiert
#              para : Name des Arrays, in dem die
Kommandozeilenvariablen
#                     geschrieben werden
#-------------------------------------------------------------------
#
proc InitParam {para} {

  global $para

  set ${para}(sourcedir)   "."
  set ${para}(destdir)     "."
  set ${para}(htmldir)     [pwd]
  set ${para}(showdeltag)   0
```

```
  set ${para}(showdelchar)  0
}

#-------------------------------------------------------------------------
# GetCommandParam {} Routine, die die Kommandozeilenparameter
liesst
#                     para  : Name des Arrays, in dem die
Kommandozeilenvariablen
#                             geschrieben werden
#-------------------------------------------------------------------------
#
proc GetCommandParam {para} {

   global argc argv $para

   while {$argv != ""} {
     set cpara [lvarpop argv]
     switch -- $cpara {
        -cf {
              if {$argv == ""} {
                PrintHelp
              } {
              set ${para}(cfname) [lvarpop argv]
              }
        }
        -hd {
              if {$argv == ""} {
                PrintHelp
              } {
              set ${para}(htmldir) [lvarpop argv]
              }
        }
        -sd {
             if {$argv == ""} {
                PrintHelp
             } {
             set ${para}(sourcedir) [lvarpop argv]
             }
        }
        -dd {
             if {$argv == ""} {
             PrintHelp
             } {
             set ${para}(destdir) [lvarpop argv]
```

```
            }
        }
        -debug {
            if {$argv == ""} {
            PrintHelp
            } {
                set option [lvarpop argv]
                if {$option == "@tag"} {
                   set ${para}(showdeltag)  1
                } elseif {$option == "char"} {
                   set ${para}(showdelchar) 1
                } elseif {$option == "all"} {
                   set ${para}(showdeltag)  1
                   set ${para}(showdelchar) 1
                } else {
                  PrintHelp
                }
            }
       }
       default {PrintHelp}
     } ; # switch
   } ; # while

   if {![info exists ${para}(cfname)]} {
      PrintHelp
   }
}

#-------------------------------------------------------------------------
# ConvertFname {} Routine, die einen Dos-Pfadnamen auf einen
Unix-Pfadnamen
#                 convertiert
#                 fname : Pfadname
#                 base  : (Optional) ersetzt Laufwerksangabe
#                 ext   : (Optional) neue Erweiterung des
Pfadnamens
#                 return     konvertierter Pfadname
#-------------------------------------------------------------------------
#
proc ConvertFname {fname {base nochange} {ext nochange}} {

   # Backslash in slash und Grosse Lateinische Buchstaben auf
kleine umwandeln
   #
```

```
   set nname [translit \\A-Z \/a-z $fname]

   # fuehrende und anschliessende Leerzeichen loeschen
   #
   set nname [string trim $nname]

   # Laufwerksangabe ersetzen
   #
   if {$base != "nochange"} {
      regsub {.:} $nname $base nname
   }

   # Erweiterung ersetzen
   #
   if {$ext != "nochange"} {
      regsub {\.[^\.]*$} $nname $ext nname
   }
   return $nname
}

#-------------------------------------------------------------------------
# getsdos {} wie gets, passt aber die Zeile-ende Sequenz
gegebenenfalls an
#              f   : Dateibezeichnung (wird von open erzeugt)
#              var : Variable, in die geschrieben werden soll
#-------------------------------------------------------------------------
#
proc getsdos {f var} {

   upvar $var unixline

   # Zeile einlesen
   #
   set s [gets $f dosline]

   # Zeile-ende Sequenz anpassen
   #
   regsub \r $dosline "" unixline

   if {$s == -1} {
      return -1
   } {
      return [clength $unixline]
   }
```

```
}

#-------------------------------------------------------------------------
# GetGroup {} Hilfsprozedur, die eine Gruppe von Bilddateien
einliesst
#             cfname    : Steuerdateiname
#             f         : Identifikator fuer geoeffnete
Steuerdatei
#             ii        : aktuelle Zeile in der Steuerdatei
#             sourcedir : Basisdirectory der CVP Dateien
#             number    : anzahl zu lesender Bilddateien
#             Return    Liste von Bilddateinamen
#-------------------------------------------------------------------------
#
proc GetGroup {cfname f ii sourcedir number} {

  upvar $ii i
  set pictpath ""
  while {$number > 0} {
    incr i
    set l [getsdos $f line]
    if {$l == -1 || $line == ""} {
       puts stderr "error: unexpected end of group"
       puts stderr "invoked by control file \"$cfname\" at line
$i"
       exit
    }

    if {[cindex $line 0] == "#"} {continue} ; # Kommentar
ueberlesen

    # Zeile verarbeiten
    #
    set line [translit \\ \/ $line]
    set gftok [lvarpop line]
    set gfname [ConvertFname $gftok $sourcedir .gif]

    if {![file exists $gfname]} {
       # GrafikFile ist nicht vorhanden
       #
       puts stderr "error: grafic file \"$gfname\" not found"
       puts stderr "invoked by control file \"$cfname\" at line
$i"
       exit ;# Programm abbrechen
```

```
    }

    # Grafikfile vorhanden
    #
    if {[clength $line] > 0} {
       puts stderr "warning: unexpected tag \"$line\" in group"
       puts stderr "invoked by control file \"$cfname\" at line
$i"
    }
    lappend pictpath [ConvertFname $gftok]
    incr number -1
  } ;# while
  return $pictpath
}

#-------------------------------------------------------------------------
# AppendBlockNr {} stellt fest, wie haeufig die letze
Kurzbeschreibung fuer
#               einen Untertitel in  ${contr}(subtitletag)
vorkommt und traegt
#               die Anzahl in ${contr}(blocknr) ein
#               contr : Name des Arrays, in dem die
Steuerinformationen
#                       gespeichert werden
#-------------------------------------------------------------------------
#
proc AppendBlockNr {contr} {

   global $contr

   # letzte Kurzbeschreibung lesen
   #
   set ll [llength [set ${contr}(subtitletag)]]
   incr ll -1
   set lasttag [lindex [set ${contr}(subtitletag)] $ll]

   # finde alle Kurzbeschreibungen namens $lasttag
   #
   set taglist [lmatch [set ${contr}(subtitletag)] $lasttag]

   # Anzahl in ${contr}(blocknr) abspeichern
   #
   lappend ${contr}(blocknr) [llength $taglist]
}
```

```
#-------------------------------------------------------------------
# GetControlFile {} routine, die die Steuerdatei einliesst und
die Existens
#                   der angebebenen files ueberprueft
#                   cfname : filename der steuerdatei
#                   basedir: basisadresse der cvp dateien
#                   contr  : Name des arrays, in dem die
Steuerinformation
#                            geschrieben wird
#-------------------------------------------------------------------
#
proc GetControlFile {contr cfname sourcedir} {

   global $contr

   # Steuerdatei vorhanden?
   #
   if {![file exists $cfname]} {
      puts stderr "error: cannot find control file \"$cfname\""
      exit
   }

   puts stderr "checking control file \"$cfname\"..."

   set f [open $cfname  r]
   set i 0

   # Pfadnamen der Textdateien
einlesen-----------------------------------
   #
   while {[getsdos $f line] > 0} {
     incr i
     set line [string trim $line]          ; # umschliessenden
Leerraum loeschen
     if {[cindex $line 0] == "#"} {continue} ; # Kommentar
ueberlesen

     # Zeile verarbeiten
     #
     set tfname [ConvertFname $line $sourcedir]
     if {[file exists $tfname]} {
         # textfile ist vorhanden
         #
```

```
        lappend ${contr}(docfname) [ConvertFname $line]
    } {
        # file nicht vorhanden
        #
        puts stderr "error: textfile \"$tfname\" not found"
        puts stderr "invoked by controlfile \"$cfname\" at
line $i"
        exit ; # Programm beenden
    }
  } ; # while

  # Informationen fuer Abbildungsbloecke
einlesen-------------------------
  #
  incr i                        ; # letzte Leerzeile mitzaehlen
  set ${contr}(subtitletag) ""
  set ${contr}(blocknr)     ""
  set ${contr}(pictnum)     ""
  set subtitletag   1
  set subtabletag   1
  set nosubtitle    1
  set pictnum       0

  set wasempty 1 ; # flag fuer Leerzeile gefunden setzen

  while {[getsdos $f line] >= 0} {
    incr i
    set line [string trim $line]          ;# umschliessenden
Leerraum loeschen
    if {[cindex $line 0] == "#"} {continue} ;# Kommentar
ueberlesen
    if {$line == ""} {
       # Leerzeile gefunden
       #
       if {!$wasempty} {
          # Die letzte Zeile war nicht leer => neues Kapitel
anfangen

          # Kapitelnummer Speichern
          #
          lappend ${contr}(pictnum) $pictnum
          set pictnum 0

          # Nummerierung neu beginnen
```

```
            #
            set subtitletag 1
            set subtabletag 1
            set nosubtitle  1

            set wasempty 1 ; # flag fuer Leerzeile gefunden
setzen
         }
         continue ; # naechste Zeile lesen
      } ;# Leerzeile gefunden

      # Zeile verarbeiten
      #
      set wasempty 0 ; # flag fuer Leerzeile gefunden auf 0
setzen
      set line [translit \\ \/ $line]
      set gftok [lvarpop line]
      set gfname [ConvertFname $gftok $sourcedir .gif]
      if {$gfname == "noimages"} {
         # Keine Grafiken in diesem Kapitel
         #
         continue
      }
      incr pictnum    ; # Abbildungsanzahl erhoehen

      if {![file exists $gfname]} {

          # GrafikFile ist nicht vorhanden
          #
          puts stderr "error: grafic file \"$gfname\" not found"
          puts stderr "invoked by control file \"$cfname\" at
line $i"
          exit ;# Programm abbrechen
       } {
          # Grafikfile ist vorhanden => tags parsen
          #
          # Standartzustand fuer Bildunterschrift initialisieren

          set pictpath [ConvertFname $gftok]
          set prefix A
          set suffix $subtitletag
          set newnum 0 ; # Zaehlvariable noch nicht geaendert

          foreach tag $line {
```

```
            if {[regexp -nocase {^G=([0-9]+)$} $tag all
number]} {
               # tag fuer Gruppierung gefunden =>
               # die naechsten number-1 Zeilen als Gruppe
einlesen
               #
               lvarcat pictpath\
                        [GetGroup $cfname $f i $sourcedir
[expr $number-1]]

            } elseif {[regexp -nocase {^U=([0-9]+)$} $tag all
number]} {
               # tag fuer "UnterTitelnummer aendern"
               # Zaehler auf number setzen
               #
               set suffix $number
               set newnum 1 ; # Zaehlvariable aendern !

            } elseif {[regexp -nocase {^U=/(.*)$} $tag all
text]} {
               # tag fuer "Regulaerstring" gefunden"
               #
               set prefix S
               set suffix $text

            } elseif {[regexp -nocase {^U=(.*)$} $tag all
text]} {
               # tag fuer "Untertitelstring setzen" gefunden"
               #
               set suffix $text

            } elseif {[regexp -nocase {^T$} $tag]} {
               # tag fuer "Abbildung ist Tabelle" gefunden
               #
               set prefix T

            } elseif {[regexp -nocase {^K$} $tag]} {
               # tag fuer "Abbildung hat keinen Untertitel"
gefunden
               #
               set prefix K
               set suffix $nosubtitle
               incr nosubtitle
```

```
            } else {
               # tag nicht bekannt
               #
               puts stderr "warning: unknown tag \"$tag\""
               puts stderr "invoked by control file
\"$cfname\" at line $i"
            }
         } ; # foreach tag
      } ; # if file exists

      # Bilddateinamen abspeichern
      #
      lappend ${contr}(pictfname) $pictpath

      # Untertiteltag merken
      #
      if {$prefix == "T" && !$newnum} {
         # Tabelle gefunden und Zaehlvariable nicht geaendern
         # => laufende Tabellennummerierung benutzen
         set suffix $subtabletag
      }
      lappend ${contr}(subtitletag) "$prefix$suffix"

      # Blocknummer fuer Kurzbeschreibung anhaengen
      #
      AppendBlockNr $contr

      # Zaehler fuer Kurzbeschreibungen anpassen
-----------------------
      #
      if {$prefix == "A"} {
         # Grafik ist Abbildung
         #
         if {[regexp {^[0-9]+$} $suffix]} {
            # Suffix ist numerisch
            #
            set subtitletag $suffix
            incr subtitletag
         }

      } elseif {$prefix == "T"} {
         # Grafik ist Tabelle
         #
```

```
          if {[regexp {^[0-9]+$} $suffix]} {
             # Suffix ist numerisch
             #
             set subtabletag $suffix
             incr subtabletag
          }
        }
      } ; # while

      if {!$wasempty} {
        # Anzahl der Abbildung des Letzten Kapitels speichern
        #
        lappend ${contr}(pictnum) $pictnum
      }

    close $f
}


#
-------------------------------------------------------------------------
# open_cap{}   Diese Funtion bekommt den Pfadnamen der *.cap
Datei
#              uebergeben. Sie legt eine globale Variable
<caplist>
#              an, in der die Bildunterschriften als
Listenelemente
#              gespeichert sind.
#
#              Eine Bildunterschrift wird erkannt, wenn die
Ziele mit
#              @BILDUNTERSCHR beginnt. Wobei vor dem
Schluesselwort
#              bel. viele Leerzeichen sein duerfen. Eine
Bildunterschrift
#              wird durch eine Leerzeile oder eine neue
Bildunterschrift
#              die in einer Zeile beginnt begrenzt.
Bildunterschriften
#              die keinen Text enthalten werden nicht in die
Liste
#              uebernommen. Alle sonstigen Schluesselworter,
wie @TAB
#              usw. werden ueberlesen.
```

```
#
# RETURN  0 OK
#
-------------------------------------------------------------------------
#
proc open_cap { path } {

  global caplist

  set caplist ""

  # *.cap Datei zum lesen oeffnen
  #
  if {[catch {set fd [open $path r]}]} {
     # Kann Datei nicht oeffnen
     #
     lappend caplist "Kann_cap_datei_nicht_oeffnen"
     return 0
  }

  set line ""
  set titel 0

  # Datei Zeilenweise einlesen und parsen
  #
  while {-1 != [gets $fd line]} {
        # ev. DOS \r rausmachen
        #
        set dosline ""
        regsub "\r" $line "" dosline

        # Beginnt diese Zeile mit einem @BILD (es duerfen
Leerzeichen
        # davor stehen) und hat nach dem = noch Text ?
        #
        if {1 == [regexp {^[ ]*@BILDUNTERSCHR[ ]+=.*[^ ]+}
$dosline]} {

           if {$titel == 1} {
              # Diese Ueberschrift beendet die momentan noch
geoeffnete
              # Ueberschrift und tragt diese in die Liste ein.
              #
              lappend caplist $elem
```

```
            }

            # Neue Zeile merken
            #
            set elem "$dosline <NewLine>"
            set titel 1

         } else {
            # Normale Zeile ohne beginnende @BILD
            #
            if { [regexp {^[ ]*$} $dosline] == 1 || \
                 [regexp {^$} $dosline] == 1 } {
               # Das ist eine Leerzeile !
               # Die Unterschrift ist fertig und kann in die
Liste
               # eingetragen werden.
               #
               if {$titel == 1} {
                  lappend caplist $elem
                  set titel 0
               }
            } elseif {$titel == 1} {
               # Diese Zeile gehoert noch zu der derzeit offenen
Unterschrift.
               # <NewLine>
               set elem "$elem $dosline <NewLine>"
            }
             # end if titel
         }
          # end else @BILDUNTER
   }
    # end while

   # Kontrollausgabe:
   #
   # foreach i $caplist {
   #    puts $i
   #    puts --------
   # }

   # Datei schliessen und OK zurueckmelden
   #
   close $fd
   return 0
```

```
}
 # end open_cap{}


#
-------------------------------------------------------------------------
# gettitel{}  Diese Funktion gibt den Bildtitel zurueck, der zu
einem
#             Vergleichsstring und der Blocknummer passt.
#
#             Die Funktion open_cap{} muss zuvor aufgerufen
werden!
#
#             Der Uebergabeparameter <search> enthaet den
Suchtyp sowie
#             den Suchstring. Folgende Moeglichkeinten sind
definiert:
#
#             * Kx wobei x eine natuerliche Zahl ist. Es wird
ein Text
#               zurueckgegeben, der eine fehlende Unterschrift
beschreibt.
#               Die Zahl x wird eingesetzt, um zu beschreiben
wieviele
#               Unterschriften in dem Dokument schon gefehlt
haben.
#               Ein cap Datei muss zuvor nicht erfolgreich
geoeffnent
#               worden sein.
#
#             * Ax wobei x String ist der nach dem "Abb"
Stichwort folgt.
#               Bsp: 1 2a Hier wird eine Abbildungsunterschrift
gesucht,
#               die zu dem regulaeren Ausdruck
#                           <pre(A)><x><post(A)>
#               passt. In der *.ord Datei durch ungequotete U=x
zu erkennen.
#
#             * Tx wobei x eine natuerliche Zahl ist. Hier wird
eine
#               Tabellenuntershrift gesucht, die zu dem
regulaeren
```

```
#                Ausdruck
#                              <pre(T)><x><post(T)>
#                passst. In der *.ord Datei durch ungequotete U=x
zu erkennen.
#
#              * Se wobei e ein regulaerer Ausdruck ist. Hier
wird nur
#                mithilfe des uebergebenen regulaeren Ausdrucks
#                              <e>
#                gesucht. In der *.ord Datei durch gequotete U=x
zu erkennen.
#
#              Im Zweifelsfall wird der zuletzt gefundene Titel
gewaehlt.
#
#              In dem Uebergabeparameter <blocknr> wird
bestimmt, der
#              wievielte passende String genommen wird, da es ja
mehrere
#              Titel mit der gleichen Nummer geben kann. Die
<blocknummer>
#              soll beim parsen der *.ord Datei immer dann
incrementiert
#              werden, wenn die Abbildungen wieder mit einer
Nummer beginnen
#              die zuvor schon man da war.
#
# RETURN  4  Koennte *.cap Datei nicht oeffnen.
#         3  groessere blocknr als passende Strings
#         2  Kein passender Titel gefunden
#         1  Fehlerhafter search String oder blocknr
#         0  OK
#
-------------------------------------------------------------------------
#
proc gettitel { search blocknr varname } {

  global caplist

  upvar $varname ergtitel

  # Pre- und Poststrings der Surchstrings (regulaere
Ausdruecke)
  # und Rueckgabetexte.
```

```
  #
  # Abbildungsuntertitel:
  # Ein Abbildungtitel oder Tabellentitel beginnt immer mit dem
  # Schluesselwort "Abb" bzw. "Tab" darauf duerfen maximal 6
oder
  # kein nicht Zahlenzeichen folgen.
  # Damit wird die Lokalitaet der Suche erreicht. Wenn ein
Titel
  # mit "Abb. 5   bla bla 2" beginnt und die 2 gesucht wird,
wuerde
  # dieser sonst auch gefunden werden.
  # Dann kommt die gesucht Suchnummer und danach darf eine Zahl
mehr folgen.
  # Die gesuchte Zeile endet mit <NewLine>, somit wird nur die
erste
  # Zeile der Bildunterschrift beachtet.
  #    {Abb[^0-9]\{0,6\}}  beim grep geht das so
  #
  set pre(K)  "@BILDUNTERSCHR = Dies ist die "
  set post(K) "te Abbildung ohne Untertitel."

  set pre(A)  {Abb[^0-9]*[^0-9]*[^0-9]*[^0-9]*[^0-9]*[^0-9]*}
  set post(A) {[^0-9].*<NewLine>}

  set pre(T)  {Tab[^0-9]*[^0-9]*[^0-9]*[^0-9]*[^0-9]*[^0-9]*}
  set post(T) {[^0-9].*<NewLine>}

  set pre(S)  {}
  set post(S) {.*<NewLine>}

  # Mittlerer Suchausdruck bestimmen
  #
  set mitte ""
  regsub "^." "$search" "" mitte


  # Wenn K als Tag wird ein Untertitel generiert. Auch wenn
keine
  # cap Datei zuvor geoeffnet wird.
  #
  if {[regexp {^K[0-9]+$} $search]} {
     # Kein Untertitel
     #
     set ergtitel "$pre(K)$mitte$post(K)"
```

```
    return 0
  }

  # Ist die Untertitelliste leer ?
  #
  if {0 == [llength $caplist]} {
     # Liste leer
     #
     return 2
  }

  # Pruefen ob die open_cap Funktion auch die *.cap Datei
oeffnen konnte
  #
  if {1 == [string match "Kann_cap_datei_nicht_oeffnen" [lindex
$caplist 0]]} {
     return 4
  }

  # Bestimme den Suchtyp und bilde den regulaeren Ausdruck
  #
  if {[regexp {^A[0-9]+} $search]} {
     # Abbildung
     #
     set re "$pre(A)$mitte$post(A)"

  } elseif {[regexp {^T[0-9]+} $search]} {
     # Tabelle
     #
     set re "$pre(T)$mitte$post(T)"

  } elseif {[regexp {^S.+} $search]} {
     # Regulaerer Ausdruck
     #
     set re "$pre(S)$mitte$post(S)"

  } else {
     # Felerhafter search String !
     #
     return 1
  }

  if {0 == [regexp {[0-9]+} $blocknr]} {
     # Falsche Blocknummer
```

```
      #
       return 1
  }


  # Alle passenden Titel suchen und in die matchlist schreiben
...
  #
  # Damit kann spaeter moeglicherweise bei einem uneindeutigen
Suchstring
  # eine Auswahl angeboten werden.
  #
  set count 0
  set matchlist ""
  foreach i $caplist {
     if {[regexp $re $i]} {
        #  Titel in die gefunden Liste uebernehmen
        #
        lappend matchlist $i
        incr count

        # puts "---\[$count\]----------------------"
        # puts $i
     }
      # end if
  }
   # end foreach

  # Feststellen, ob ueberhaupt was gefunden wurde
  #
  if {$count == 0} {
     # Nichts gefunden
     #
     return 2
  }

  # Feststellen, ob weniger Strings gefunden wurden als die
blocknr
  # verlangt.
  #
  if {$count < $blocknr} {
     # Oh, je .. da stimmt was nicht
     #
     return 3
```

```
  }

  # Titel auswaehlen (Listen beginnen mit 0)
  #
  incr blocknr -1
  set ergtitel [lindex $matchlist $blocknr]

  return 0
}
 # end gettitel{}


#
-------------------------------------------------------------------------
# mkimgpage{}   Diese Funktion erstellt eine HTML-Seite in der
die Bilder
#               der in dem Uebergabeparameter <imglist> und
danach der
#               in <utitel> stehende Untertiteltext zu sehen
ist. <pfad>
#               bestimmt den Dateinamen des Dokumentes. Eine
bereits
#               existerende Datei wird ohne Warnung
ueberschrieben.
#
# RETURN  0 OK
#         1 Kann File nicht erstellen
#
-------------------------------------------------------------------------
#
proc mkimgpage {pfad imglist utitel} {

  set html(absatz)  "<P>"
  set html(imgpre)  "<IMG SRC=\""
  set html(imgpost) "\">"

  # Datei zum schreiben oeffnen
  #
  if {[catch {set fpw [open $pfad w+]}]} {
     # Kann Datei nicht oeffnen
     #
     return 1
  }
```

```
  # HTML Document Praefix
  #
  puts $fpw "<HTML>\n"
  puts $fpw "<BODY>\n"

  # Bildertags schreiben
  #
  foreach i $imglist {
     puts $fpw $html(absatz)
     puts $fpw "$html(imgpre)$i$html(imgpost)"
     puts $fpw ""
  }

  # Untertitel schreiben
  #
  puts $fpw $html(absatz)
  puts $fpw $utitel

  # HTML Document Suffix
  #
  puts $fpw "</BODY>\n"
  puts $fpw "</HTML>\n"

  close $fpw
  return 0
}
 # end mkimgtitel{}


#
-------------------------------------------------------------------------
# cp{}    Diese Funktion kopiert ein File. Dabei werden nicht
vorhandene
#         Zielverzeichnisse erzeugt. Wenn das Zielfile schon
existiert
#         wird eine Fehlermedung zurueckgegeben und das
Kopieren abgebrochen.
#         Pfadangaben koennen mit / beginnen also relativ zum
root Verzeichnis
#         sein oder relativ zum working directory sein und
beginnen somit
#         nicht mit /.
#
#         source  Die zu kopierende Datei mit kompletten Pfad
```

```
#
#         dest    Zieldatei mit Pfadangabe. Der Filename kann
sich von
#                 dem source Filenamen unterscheiden.
#
#
# RETURN  0  Ok, alles gemacht.
#         1  source Datei nicht gefunden oder gelesen werden.
#         2  dest Filename fehlt im Pfad.
#         3  dest File existiert schon.
#         4  Kann Verzeichnis nicht erstellen.
#         5  Kann source nicht zum lesen oeffnen.
#         6  Kann dest nicht zum schreiben oeffnen.
#         7  Konnte Datei nicht vollstaendig kopieren.
#
-------------------------------------------------------------------------
#
proc cp {source dest} {

  # Pruefen ob es das source File gibt und auch lesbar ist.
  #
  if {![file exists $source] || ![file readable $source]} {
     # nicht lesbar
     #
     return 1
  }

  # Pruefen ob auch ein dest Filename angegeben wurde
  #
  if {[string match "*/" $dest]} {
     # dest Filename fehlt
     #
     return 2
  }

  # Pruefen ob es das dest File schon gibt.
  #
  if {[file exists $dest]} {
     # gibt's schon
     #
     return 3
  }

  # Gibt es schon alle Zielverzeichnisse ?
```

```
  #
  set destpath [file dirname $dest]
  if {![file isdirectory $destpath]} {
     # Nein, gibt's noch nicht!
     # Alle nicht vorhandenen Verzeichnisse des Zielepfades
erstellen
     #
     if {0 != [catch {mkdir -path $destpath}]} {
        # Konnte die Verzeichnisse nicht erstellen
        #
        return 4
     }
  }

  # Files oeffnen
  #
  if {[catch {set fpr [open $source r]}]} {
     # kann eigendlicht nicht sein.
     #
     return 5
  }

  if {[catch {set fpw [open $dest w+]}]} {
     # kann nicht zum schreiben oeffnen
     #
     return 6
  }

  # Groesse des Files feststellen
  #
  set size [file size $source]

  # Da eingendliche Kopieren
  #
  if {$size != [copyfile $fpr $fpw]} {
     # Konnte Datei nicht vollstaendig kopieren
     #
     return 7
  }

  close $fpr
  close $fpw

  return 0
```

```
}
 # end cp{}

#-------------------------------------------------------------------------
# LoadText {} laedt ein Textfile als string in eine globale
#             Variable ein; spezialTags <NewLine> <EmptyLine> -
ZeilenEnde
#             fname    : Textfile Name
#             textbuf  : Name der globalen Variablen, in der der
Text geladen
#                        wird
#-------------------------------------------------------------------------
#
proc LoadText {fname textbuf} {

   global $textbuf

   set f [open $fname  r]
   while {[getsdos $f line] >= 0} {
      if {$line == ""} {
         append $textbuf " <EmptyLine> "
      } {
      append $textbuf $line " <NewLine> "
      }
   }
}

#-------------------------------------------------------------------------
#  SaveText {} speichert einen Text aus einer Variablen
#               textpath : pfad der zu speichernden Datei
#               dest     : globale variable, in der der Text
steht
#-------------------------------------------------------------------------
#
proc SaveText {textpath dest} {

    global $dest

    if {[file isdirectory [file dirname $textpath]] == 0} {
       # Pfad ist kein momentanes Verzeichnis =>
Subverzeichnisse anlegen
       #
       mkdir -path [file dirname $textpath]
    }
```

```
    set f [open $textpath w+]
    puts $f [set $dest]
    close $f
}

#-------------------------------------------------------------------------
# GetChapterFname     ermittelt den Filenamen des Kapitels
(*.ord-DateiName)
#                     ohne die .ord Erweiterung
#                     para  : Array, in dem die
Kommandozeilenparameter stehen
#-------------------------------------------------------------------------
#
proc GetChapterFname {para} {

   global $para

   set ordpath  [set ${para}(cfname)]        ; #
OrdnungsdateiPfad
   set ordfname [file tail $ordpath]         ; #
OrdnungsdateiName

   # Erweiterung streichen, falls vorhanden
   #
   set chapterfname $ordfname
   if {[file ext $ordfname] != ""} {
       regexp {(.*)\.[^\.]*} $ordfname all chapterfname
   }
   return $chapterfname
}

#-------------------------------------------------------------------------
# ConvertTpname {} wie ConvertFname, nur ohne Groessenaenderung
der
#                  Zeichen
#-------------------------------------------------------------------------
#
proc ConvertTpname {fname {base nochange} {ext nochange}} {

   # Backslash in slash umwandeln
   #
   set nname [translit \\ \/ $fname]
```

```
   # fuehrende und anschliessende Leerzeichen loeschen
   #
   set nname [string trim $nname]

   # Laufwerksangabe ersetzen
   #
   if {$base != "nochange"} {
      regsub {.:} $nname $base nname
   }

   # Erweiterung ersetzen
   #
   if {$ext != "nochange"} {
      regsub {\.[^\.]*$} $nname $ext nname
   }
   return $nname
}

#-------------------------------------------------------------------------
# GetTextPagePath {} Produziert den Pfad/Filenamen fuer eine
Textseite
#
#                    para   : Array, in dem die
Kommandozeilenparameter stehen
#                    stat   : Array, in dem die Parserzustaende
stehen
#                    prefix : Prefix, der das "c:" des
Orginalnamens ersetzt
#
#                     return    kommpletter Pfadname der
Textseite
# Pfadnamenaufbau:
#
<prefix/TextfileDir/KontrollfileName_Textfilename_CNkapitelnr>
#-------------------------------------------------------------------------
#
proc GetTextPagePath {para stat prefix} {

  global $para $stat

  set chapternum [set ${stat}(chapternum)]   ; # Kapitelnummer

  set fpath [set ${stat}(file)]              ; # Pfadname des
Textes
```

```
  set fname [file tail $fpath]                ; # Filename des
Textes
  set fdir  [file dirname $fpath]             ; # Verzeichniss
des Textes

  set cfname [GetChapterFname $para]          ; # Filename der
Kontrolldatei

  set tfpage [ConvertTpname $fdir $prefix]    ; # Verzeichniss
der TextSeite

  # Pfad zusammenbauen
  #
  append textpagepath $tfpage / $cfname _ $fname _ CN
$chapternum

  return $textpagepath
}

#-------------------------------------------------------------------------
# GetImmagePagePath {} Produziert den Pfad/Filenamen fuer eine
Grafikseite
#
#                      contr  : Array, in dem die
Steuerinformationen stehen
#                      para   : Array, in dem die
Kommandozeilenparamter stehen
#                      stat   : Array, in dem die
Parserzustaende stehen
#                      prefix : Prefix, der das "c:" des
Orginalnamens ersetzt
#                      index  : Index fuer Listenzugriffe auf
Dateinamen
#
#                      return    kommpletter Pfadname der
Grafikseite
# Pfadnamenaufbau:
#
<TextSeitenPfad_GrafikFilename1_GrafikFilename2..._GPgseitennummer>
#-------------------------------------------------------------------------
#
proc GetImmagePagePath {contr para stat prefix index} {

   global $contr $stat $para
```

```
   # Filepfade der Grafikseite ermitteln
   #
   set immagenum [set ${stat}(immagepage)]
   set immagepaths [lindex [set ${contr}(pictfname)] $index]

   # Grafikfilenamen ermitteln
   #
   set immagenames ""
   foreach gpath $immagepaths {
     append immagenames [file tail $gpath] _
   }

   # TextSeitenPfad ermitteln
   #
   set textpagepath [GetTextPagePath $para $stat $prefix]

   # GrafikSeitenPfad zusammenbauen
   #
   append immagepagepath $textpagepath _ $immagenames GP
$immagenum

   return $immagepagepath
}

#-------------------------------------------------------------------------
# GetImmagePath {} Produziert den Pfad/Filenamen fuer eine
Grafik
#
#                  contr  : Array, in dem die
Steuerinformationen stehen
#                  para   : Array, in dem die
Kommandozeilenparamter stehen
#                  stat   : Array, in dem die Parserzustaende
stehen
#                  prefix : Prefix, der das "c:" des
Orginalnamens ersetzt
#                  index  : Index fuer Listenzugriffe auf
Dateinamen
#                  gn     : Grafiknummer dieser Abbildung
(siehe unten)
#
# Pfadnamenaufbau:
```

```
#
<GrafikSeitenPfad_GNgrafiknummer_INAbbildungsnummer.gif>
#
#         Grafiknummer     : diese Grafik ist Grafik i der
Abbildungsseite
#         Abbildungsnummer : laufende Abbildungsnummer
innerhalb eines
#                            Publisherkapitels
#
#-------------------------------------------------------------------
#
proc GetImmagePath {contr para stat prefix index gn} {

   global $stat

   # akueller Grafikseitennummer
   #
   set ipn [GetImmagePagePath $contr $para $stat $prefix
$index]

   # laufende Abbildungsnummer
   #
   set depictnum [set ${stat}(depictnum)]

   # Pfad zusammenbauen
   #
   append immagepath  $ipn _ GN $gn _ IN $depictnum .gif

   return $immagepath
}

#-------------------------------------------------------------------
# proc gettitelerr {} produziert eine Klartextfehlermeldung
fuer gettitel{}
#                    und bricht das Programm eventuell ab
#                    para   : Array, in dem die
Kommandozeilenparameter stehen
#                    error  : Fehlernummer
#                    search : aktueller Suchtag
#                    blocknr: Blocknummer des Suchtags
#-------------------------------------------------------------------
#
proc gettitelerr {para error search blocknr} {
```

```
   global $para

   switch $error {
    3 {puts stderr "error: gettitel : too big blocknumber
\"$blocknr\" for tag \"$search\""}
    2 {puts stderr "error: gettitel : cannot find title with
tag \"$search\""}
    1 {puts stderr "error: gettitel : don't know tag
\"$search\""}
   }

   set cfname [set ${para}(cfname)]                ; # name der
Kontrolldatei
   if {$error == 4} {
      set hd [set ${para}(sourcedir)]              ; # CVP
Basisdir.
      set capname [ConvertFname $cfname $hd .cap]  ; # Name der
.cap Datei
      puts stderr "warning: was not able to open file
\"$capname\""
      puts stderr "invoked by control file \"$cfname\""
      return
   }

   puts stderr "invoked by control file \"$cfname\""
   exit
}

#-------------------------------------------------------------------------
# InsertLinks  {} erzeugt die Links (Aufrufstartpunkte) und
Grafiken
#                  fuer ein (reales) Kapitel; wird
normalerweise
#                  am Ende eines realen Kapitels aufgerufen
#                  contr  : Array, in dem Steuerinformationen
stehen
#                           geschrieben wird
#                  para   : Array, in dem die
Kommandozeilenparameter stehen
#                  stat   : Array, in dem der Parsersatus
gespeichert wird
#                  source : Name der globalen Variablen, in dem
der Text steht
```

```
#                   dest    : Name der globalen Variablen, in der
das Ergebnis
#-------------------------------------------------------------------------
#
proc InsertLinks {contr para stat source dest} {

   global $contr $para $stat $source $dest html subtitle
subtbuffer graficsubt

   # subtitle   : Untertiteltext (ungeparst)
   # subtbuffer : Kopie von subtitle
   # graficsubt : geparster Untertiteltext fuer Grafikseiten

   set cfname [set ${para}(cfname)]          ; # Name des
Kontrollfiles

   # ermittle Anzahl der Abbildungen in diesem Kapitel
   #
   set pictnums [set ${contr}(pictnum)]      ; # Liste der
Anzahl der Abb.
   set chapternum  [set ${stat}(chapternum)] ; # Laufende
Kapitelnummer kopieren
   set index [expr $chapternum-1]            ; # index auf
Listen ermitteln

   set num [lindex $pictnums $index]         ; # Anzahl der
Abb. dieses Kap.

   if {$pictnums == ""} {
      # gar keine Abbildungen in diesem Publisher Kapitel
      #
      set num 0
   }

   if {$num == ""} {
     # keine passenden (realen) Kapitelbeschreibugen in der
     # .ord -Datei gefunden
     puts stderr "warning: too much chapter in textfile \"[set
${stat}(file)]\""
     puts stderr "invoked by control file \"$cfname\""
     return
   }

   # aktuelle Zeile fuer Parser merken
```

```
   #
   set pline [set ${stat}(line)]

   # Parser fuer Untertitel initialisieren
   #
   set ${stat}(line) 0

   set sprefix [set ${para}(sourcedir)]         ; #
Quell-Pfadprefix
   set dprefix [set ${para}(destdir)]           ; #
Ziel-Pfadprefix
   set hprefix [set ${para}(htmldir)]           ; #
html-Pfadprefix

   # Alle Links zu diesem (realen) Kapitel einbinden
   #
   set from [set ${stat}(immagepage)]
   set to   [expr $from + $num]
   loop index $from $to {

      # passenden Untertitel holen
----------------------------------------
      #
      set search [lindex [set ${contr}(subtitletag)] $index]
      set blocknr [lindex [set ${contr}(blocknr)] $index]
      set error [gettitel $search $blocknr subtitle]
      if {$error != 0} {
       gettitelerr $para $error $search $blocknr
       exit
      }

      # Link einfuegen
------------------------------------------------------
      #
      set ippath [GetImmagePagePath $contr $para $stat $hprefix
$index]
                                                   ; #
GrafikseitenPfad
      set subtbuffer $subtitle                     ; #
Bildbeschreibung kopieren
      lvarpop subtitle                             ; # @tag
entfernen
      lvarpop subtitle                             ; # "="
entfernen
```

```
      set linktext [lvarpop subtitle]              ; # erstes
Wort als Ankertext
      set linkinsert [TranslateChars $para $stat "$linktext "]
      set ${stat}(readpara)    0            ; #
Paragraphenzeichen nicht lesbar
      set ${stat}(setpara)     0            ; #
Paragraphenzeichen nicht setzbar
      append $dest "$html(paragraph)<A HREF=$ippath>
$linkinsert </A>"
      TranslateText $contr $para $stat subtitle $dest

      # zugehoerige GrafikFiles kopieren und Html-Zielpfade
erzeugen----------
      #
      set pictures [set ${contr}(pictfname)]    ; # Liste aller
Bilder
      set pictfnames [lindex $pictures $index]  ; # Liste
zugehoeriger Bilder
      set  imglist ""                           ; # Liste der
html-Zielpfade
      set gn 0                                  ; # Position
auf der G.seite
      foreach picture $pictfnames {
       set sp [ConvertFname $picture $sprefix .gif]         ; #
Quel-Pfad
       set dp [GetImmagePath $contr $para $stat $dprefix $index
$gn]
                                                            #
Ziel-Pfad
       set hp [GetImmagePath $contr $para $stat $hprefix $index
$gn]
                                                            #
html-ZielPfad

       lappend imglist $hp                      ; # Liste der
html-Zielpfade
       set err  [cp $sp $dp]                    ; # Grafic
kopieren
       if {$err != 0} {
          puts stderr "warning: cp : #$err"
       }
       incr gn
       incr ${stat}(depictnum) ; # aktuelle Grafiknummer
erhoehen
```

```
      }

      # Grafikseite erzeugen und
abspeichern----------------------------------
      #
      set ippath [GetImmagePagePath $contr $para $stat $dprefix
$index]
                                                                  #
GrafikseitenPfad
      InitParser statussub
      set graficsubt ""
      TranslateText $contr $para statussub subtbuffer
graficsubt
      set err [mkimgpage $ippath $imglist $graficsubt]
      if {$err != 0} {
         puts stderr "Warning: mkimgpage : #$err"
      }


      incr ${stat}(immagepage) ; # aktuelle Grafikseitennummer
erhoehen
   } ; # loop

   # Textseite
abspeichern-------------------------------------------------
   #
   set tppath [GetTextPagePath $para $stat $dprefix]

   set $dest "<HTML>\n <BODY> \n[set $dest]" ; # HTML Document
Praefix
   append $dest "</BODY>\n </HTML>"          ; # HTML Document
Suffix

   SaveText $tppath $dest                    ; # Kapitel
abspeichern
   set $dest ""                              ; # Zielpuffer
loeschen

   puts stdout $tppath ;# Ausgabe fuer log-Datei der erzeugten
files

   # laufende Zeile fuer Parser wieder setzen
   #
   set ${stat}(line) $pline
```

```
}

#-------------------------------------------------------------------------
# TranslateChars {} Uebersetzt 7-Bit ASCII Text nach Html
#                   para  : Array, das die
Kommandzeilenparameter enthaelt
#                   stat  : Array, in dem Parserzustaende
stehen
#                   word  : zu uebersetzendes Wort
#                   return angepasstes Wort
#-------------------------------------------------------------------------
#
proc TranslateChars {para stat word} {

   global char $para $stat

   # spezielle HTML-Zeichen ersetzen
   #
   regsub {&}  $word {\&amp;} word
   regsub {<<} $word {\&lt;}  word
   regsub {>>} $word {\&gt;}  word

   # <> Sonderzeichen ersetzen,
   #
   while {[regexp -indices {<[^>]*>} $word trange]} {

         # noch mindestens 1 <> token vorhanden

         # Token Anfang und Ende ermitteln
         #
         set from [lindex $trange 0]
         set to [lindex $trange 1]

         # Token ermitteln
         #
         set token [string range $word [incr from] [incr to
-1]]

         # Wort bis zum Tokenanfang ans Ergebnis haengen
         #
         append result [string range $word 0 [incr from -2]]

         # Wort bis zum Tokenende loeschen
```

```
        #
        set word [string range $word [incr to 2] end]

        if {[info exist char($token)]} {

            # token bekannt => html-Zeichencode einsetzen
            #
            append result $char($token)
        } {
            # token unbekannt

            if {[set ${para}(showdelchar)]} {

              # Programm im zugehoerigen Debugmodus => Warnung
ausgeben
              #
              set fname [ConvertFname [set ${stat}(file)] [set
${para}(sourcedir)]]
              puts stderr "warning: unknown charakter
\"<$token>\""
              puts stderr "invoked by textfile \"$fname\" at
line [set ${stat}(line)]"
            }
        }
   }
   append result $word
   return $result
}

#-------------------------------------------------------------------------
# Translate@Text {} Hilfsparser, der @-tags uebersetzt:
#                   normalerweise wird der Text bis zum
Zeilenende gelesen
#                   und ohne spezielle Attribute uebersetzt;
#                   Wird in diesem Text ein <R> gefunden, so
wird zurueck-
#                   gesprungen und das flag ${para}(attag)
gesetzt, damit
#                   der Aufrufer eventuelle "ende"-tags setzen
kann.
#                   Wird diese Routine mit gesetztem flag
${para}(attag) auf-
#                   gerufen, so gilt nur eine Leerzeile als
ende-Symbol, es
```

```
#                   wird aber trotzdem bei jedem <R>
zurueckgesprungen
#
#                   contr  : Array, in dem Steuerinformationen
stehen
#                   para   : Array, in dem die
Kommandozeilenparameter stehen
#                   stat   : Array, in dem der Parsersatus
gespeichert wird
#                   source : Name der globalen Variablen, in
der der Text steht
#                   dest   : Name der globalen Variablen, in
der das Ergebnis
#                            geschrieben wird
#-------------------------------------------------------------------------
#
proc Translate@Text {contr para stat source dest} {

  global $contr $source $dest $para $stat html char

    while {[set $source] != ""} {
      set token [ctoken $source " "]

      if {[regexp {<R>} $token]} {

         # @-Zeile aufgeteilt =>

         # Parser in den Zustand "@tag" setzen
         #
         set ${stat}(attag) 1

         # Wort koennte noch weitere Zeichen enthalten => Wort
abspeichern
         #
         append $dest [TranslateChars $para $stat "$token "]

         # Zurueckspringen, damit vom Hauptparser eventuell
         # End-tags gesetzt werden koennen
         #
         return

      } elseif { $token == "<NewLine>"} {

         incr ${stat}(line)
```

```
        if {![set ${stat}(attag)]} {
            # Automat nicht im attag-Modus => Zuruecksprungen
            #
            return
        }

    } elseif { $token == "<EmptyLine>"} {

            # Automat aus dem attag-Modus holen
            #
            incr ${stat}(line)
            set ${stat}(attag) 0

            return

    }  else {

      # normales Wort uebersetzen
      #
      append $dest [TranslateChars $para $stat "$token "]
    }
  } ; # while
}

#-------------------------------------------------------------------------
# TranslateText {} Parser, der einen CVP Text uebersetzt;
Paragraphen
#                   (Absaetze) duerfen nur zwischen Texten
uebersetzt werden
#
#                   contr  : Array, in dem Steuerinformationen
stehen
#                   para   : Array, in dem die
Kommandozeilenparameter stehen
#                   stat   : Array, in dem der Parsersatus
gespeichert wird
#                   source : Name der globalen Variablen, in
dem der Text steht
#                   dest   : Name der globalen Variablen, in
der das Ergebnis
#                            geschrieben wird
#-------------------------------------------------------------------------
#
```

```
proc TranslateText {contr para stat source dest} {

  global $contr $source $dest $para $stat html char

  while {[set $source] != ""} {
    set token [ctoken $source " "]

    # @-tokens
----------------------------------------------------
    #
    if {[cindex $token 0] == "@"} {

       # @tag gefunden
       #
       if {$token == "@KAPITEL<154>BER" || $token ==
"@KAP_<154>BERSCHR"} {

          # Kapiteluebersch rift
gefunden-------------------------------------
          #
          set ${stat}(readpara) 0 ; # Paragraphen nach Titeln
nicht lesbar
          set ${stat}(setpara)  0 ; # Paragraphen nach Titeln
nicht setzbar
          set ${stat}(attag) 1

          set token [ctoken $source " "]
          if {$token == "="} {
             # Hauptkapiteluebersch rift gefunden
             #
             set token [ctoken $source " "]
             if {$token == "<NewLine>"} {

               # Leere Kapiteluebersch riften ueberlesen
               #
               incr ${stat}(line)
               continue

             } else {

               # Kapiteluebersch rift nicht leer =>

               # letztes gelesenes Token wieder in den
Quellcode einsetzen
```

```
                #
                set $source "$token [set $source]"


                if {[set ${stat}(chapternum)] > 0} {
                    # (reales) Kapitelende gefunden =>
Hypertextanker setzen
                    #
                    InsertLinks $contr $para $stat $source $dest
                }
                incr ${stat}(chapternum) ; # Anzahl von Kapiteln
erhoehen

                # Hauptkapitelueberschrift setzen----------
                #
                append $dest $html(title_start)
                Translate@Text $contr $para $stat $source $dest
                append $dest $html(title_stop)

                # Fals Ueberschrift mehrzeilig => weitere Titel
generieren
                #
                while {[set ${stat}(attag)]} {
                  append $dest $html(title_start)
                  Translate@Text $contr $para $stat $source
$dest
                  append $dest $html(title_stop)
                }
              } ; # Titel nicht leer
            } else {

               # Text bis zum Gleichheitszeichen ueberlesen
               #
               ctoken $source "="
               ctoken $source " "

               # Unterkapitelueberschrift setzen -------------
               #
               append $dest $html(subhead1_start)
               Translate@Text $contr $para $stat $source $dest
               append $dest $html(subhead1_stop)

               # Falls Ueberschrift mehrzeilig wietere Titel
generieren
```

```
            #
            while {[set ${stat}(attag)]} {
              append $dest $html(subhead1_start)
              Translate@Text $contr $para $stat $source $dest
              append $dest $html(subhead1_stop)
            }
         }
      } else {

         # unbekanntes @-token gefunden

         if {[set ${para}(showdeltag)]} {

            # Programm im zugehoerigen Debug-Modus => Warnung
ausgeben:
            #
            set fname [ConvertFname [set ${stat}(file)] [set
${para}(sourcedir)]]
            puts stderr "warning: unknown @tag \"$token\""
            puts stderr "invoked by textfile \"$fname\" at
line [set ${stat}(line)]"
         }

         # Rest des @tags ueberlesen
         #
         set token [ctoken $source =]
         ctoken $source " "
      }

   # <>-tokens
-------------------------------------------------
   #
   } elseif {$token == "<EmptyLine>"} {

      # Leerzeile gefunden
      #
      incr ${stat}(line)

      if {[set ${stat}(readpara)]} {
            set ${stat}(setpara) 1 ; # Hinweis fuer Paragraph
setzen
      }

   }  elseif {$token == "<NewLine>"} {
```

```
        # neue source-Zeile gefunden
        #
        incr ${stat}(line)
        append $dest \n

    } else {

        # Token ist Wort

        if {[set ${stat}(setpara)]} {

           # letztes token war Leerzeile gefunden => Paragraph
einfuegen
           #
           append $dest $html(paragraph)
           set ${stat}(setpara) 0 ; # Vermerk fuer Paragraph
eingefuegt
        }
        # Zeichen uebersetzen
        #
        set translated [TranslateChars $para $stat "$token "]
        append $dest $translated
        if {$translated != ""} {
          set ${stat}(readpara) 1    ; # Paragraphen lesbar
machen
        }
    }
  } ; # while
}

#-------------------------------------------------------------------------
# InitParser {} Prozedur, die einen InterpreterStatus
initialisiert
#               stat    : Array, in dem Parserzustaende
gespeichert sind
#-------------------------------------------------------------------------
#
proc InitParser {stat} {

   global $stat

   set ${stat}(line)       1 ; # aktuelle Textzeile
   set ${stat}(chapternum) 0 ; # aktuelle Kapitelnummer
```

```
   set ${stat}(immagepage) 0 ; # akutelle Grafikseitennummer
   set ${stat}(depictnum)  0 ; # aktuelle Abbildungsnummer
   set ${stat}(attag)      0 ; # attag-Modus ausgeschaltet
   set ${stat}(readpara)   0 ; # Paragraphenzeichen nicht
lesbar
   set ${stat}(setpara)    0 ; # Paragraphenzeichen nicht
setzbar
}

#-------------------------------------------------------------------
# TranslateDoc {} Prozedur, die das CVP Document uebersetzt
#              contr    : Name der globalen Variablen, in der
die
#                         Steuerinformationen gespeichert sind
#              dest     : Name der globalen Variablen, in der
das Ergebnis
#                         geschrieben wird
#              para     : Array, in dem die
Kommandozeilenparameter stehen
#              stat     : Array, in dem der Interpreterstatus
gespeichert wird
#-------------------------------------------------------------------
#
proc TranslateDoc {contr dest para stat} {

   global $contr $dest $para $stat textbuf1


   # Parser Initialisieren
   #
   InitParser $stat

   set pictnums [set ${contr}(pictnum)] ;# Liste der Anzahl der
Abb. kopieren
   if {$pictnums != ""} {

      # es sind Abbildungen in diesem PublisherKapitel

      # Routinen zum Bildunterschrift finden initialisieren
      #
      set cfname [set ${para}(cfname)]      ; # Name der
Kontrolldatei
      set source [set ${para}(sourcedir)]   ; # Quell-directory
```

```
      set capname [ConvertFname $cfname $source .cap] ; #  name
der .cap Datei
      open_cap $capname
   }

   # Text uebersetzen
   #
   foreach file [set ${contr}(docfname)] {
     set fname [ConvertFname $file [set ${para}(sourcedir)]]
     set ${stat}(line)        1            ; # Textzeile auf 1
     set ${stat}(file)        $file        ; # aktueller
Textdatei Name
     set ${stat}(readpara)    0            ; #
Paragraphenzeichen nicht lesbar
     set ${stat}(setpara)     0            ; #
Paragraphenzeichen nicht setzbar
     LoadText $fname textbuf1              ; # Text einladen
     puts stderr "translate file \"$fname\""
     TranslateText $contr $para $stat textbuf1 $dest ; # Text
uebersetzen
   }

   # PublisherKapitelende erreicht => Hypertextlinks setzen
   #
   if {[set ${stat}(chapternum)] == 0} {
      # gar keine Hauptkapiteluberschrift gefunden
      # trotzdem links setzen, dafuer Kapitelnummer auf 1
      #
      set ${stat}(chapternum) 1
   }
   InsertLinks $contr $para $stat textbuf1 $dest
}

#--------------main----------------------------------------------------------

# Kommandozeilenparameter lesen
#
InitParam param
GetCommandParam param

# Steuerdatei einlesen
#
GetControlFile control $param(cfname) $param(sourcedir)
```

```
# Document uebersetzen
#
InitHtmlDefs
InitCharTrans
TranslateDoc control body param status

######################TESTAUSGABE##############################################
#
#  puts stderr "\nTestausgabe"
#  for_array_keys key control {
#  puts stderr "$key: $control($key)"
#  }
```

Anhang D

Fehlerliste LfU-Bericht

Die folgende Liste dokumentiert den Zustand der an das IPF übergebenen Dateien als Grundlage für die Ventura-HTML Konvertierung.

```
Hier ist die Liste der fuer den Umweltbericht benoetigten
Daten.

Das <-- Zeichen am rechten Rand kennzeichnet fehlende Daten.



Umweltdaten 91/92

Zusammenstellung der Ventura - Publikationsdateien sowie deren
zugehoerigen
Kapiteldateien.  [..] enthaelt die Seitenzahl in dem gedruckten
Dokument.

------------------------------------------------------------------------
Struktur der Ventura Publisher Dokumente:

Mit dem Ventura Publisher kann man Dokumente erzeugen, die er
Kapitel
nennt. Diese werden in Dateien mit der Endung *.chp
gespeichert.
Weiterhin hat man die Moeglichkeit mehrere solche Kapitel zu
einer
so genannten Publikation zusammenzustellen. Diese werden in
Files
mit der Endung *.pub gesichert.

------------------------------------------------------------------------
```

```
DECKBLATT  fehlt
<---

SCHMUTZBLATT (inneres Deckblatt)

  * c:\iud\1_sch_bl\schmutzb.chp

IMPRESSUM

  * c:\iud\1-impres\impress.chp
    22 c:\1ud\1-impres\impress.doc

VORWORT MINISTER

  * c:\iud\1-vw-mini\vw-min.chp
    03 c:\1ud\mini-03.tif
    03 c:\1ud\min-usch.tif
    04 c:\1ud\1-vw-min\vw-min.doc

VORWORT PRAESIDENT

  * c:\iud\1-vw-pr\vw-prae.chp
    - Das Bild der Unterschrift fehlt.
<--
    03 c:\1ud\1-vw-pr\praesid3.tif
    22 c:\1ud\1-vw-pr\vw-prae.doc

INHALTSVERZEICHNIS

  * c:\iud\1-inhalt\ges-inh.chp

------------------------------------------------------------------------
AI - ALLGEMEINE DATEN   c:\iud\ai\a-i.pub

  * c:\iud\ai\ai-ih.chp    Inhalt [AI-1]
    04 c:\1ud\ai\a-i-ih.doc

  * c:\iud\ai\a-i-01.chp   Bevoelkerungsentwicklung [AI-2 .. 5]
    - Abb. 2 fehlt im Dokument
<--
    07 c:\1ud\ai\ad_abb01.eps
    07 c:\1ud\ai\siedlew.eps
    07 c:\1ud\ai\siedlfl.eps
    22 c:\1ud\ai\flae_1.doc
```

```
  07 c:\1ud\ai\bevent-1.eps
  22 c:\1ud\ai\bevent-1.doc

* c:\iud\ai\a-i-02.chp   Verkehr [AI-6 .. 15]
  07 c:\1ud\ai\vk_ab18d.eps
  07 c:\1ud\ai\vk_ab18c.eps
  09 c:\1ud\ai\vk_ab_13.wmf
  09 c:\1ud\ai\vk_ab_14.wmf
  07 c:\1ud\ai\vk_abb17.eps
  07 c:\1ud\ai\vk_abb16.eps
  07 c:\1ud\ai\vk_abb15.eps
  07 c:\1ud\ai\vk_abb19.eps
  07 c:\1ud\ai\vk_ab18e.eps
  07 c:\1ud\ai\vk_ab18b.eps
  07 c:\1ud\ai\vk_ab18a.eps
  07 c:\1ud\ai\vk_abb11.eps
  07 c:\1ud\ai\vk_abb12.eps
  07 c:\1ud\ai\vk_abb10.eps
  07 c:\1ud\ai\vk_abb09.eps
  07 c:\1ud\ai\vk_abb07.eps
  07 c:\1ud\ai\vk_abb06.eps
  07 c:\1ud\ai\vk_abb05.eps
  07 c:\1ud\ai\vk_abb04.eps
  07 c:\1ud\ai\vk_abb03.eps
  07 c:\1ud\ai\vk_abb02.eps
  07 c:\1ud\ai\vk_abb01.eps
  07 c:\1ud\ai\t1.eps
  04 c:\1ud\ai\a-i-02.doc

* c:\iud\ai\a-i-03.chp   Energie [AI-16 .. 23]
  07 c:\1ud\ai\en_abb10.eps
  07 c:\1ud\ai\en_ab12c.eps
  07 c:\1ud\ai\en_ab12b.eps
  07 c:\1ud\ai\en_ab12a.eps
  07 c:\1ud\ai\en_abb11.eps
  07 c:\1ud\ai\en_abb09.eps
  07 c:\1ud\ai\en_abb08.eps
  07 c:\1ud\ai\en_abb06.eps
  07 c:\1ud\ai\en_abb05.eps
  07 c:\1ud\ai\en_abb04.eps
  07 c:\1ud\ai\en_abb03.eps
  07 c:\1ud\ai\en_abb02.eps
  07 c:\1ud\ai\en_abb01.eps
  04 c:\1ud\ai\a-i-03.doc
```

```
---------------------------------------------------------------------
AII - ABFALL, ALTLASTEN   c:\iud\aii\a-ii.pub

  * c:\iud\aii\a-ii-ih.chp   Inhalt [AII-1]
    04 c:\1ud\aii\a-ii-ih.doc

  * c:\iud\aii\a-ii-01n.chp  Abfallwirtschaft  [AII-2 .. 3]
    07 C:\1UD\AII\ab_abb01.eps
    01 c:\1ud\aii\a-ii-01n.txt

  * c:\iud\aii\a-ii-02n.chp  Gesamtabfallaufkommen [AII-4 .. 8]
    07 C:\1UD\AII\af_abb02.eps
    07 C:\1UD\AII\af_abb03.eps
    07 c:\1ud\aii\af_abb05.eps
    07 c:\1ud\aii\af_abb04.eps
    04 c:\1ud\aii\a-ii-02n.doc

  * c:\iud\aii\a-ii-03n.chp  Abfallbilanz [AII-9 .. 13]
    07 c:\1ud\aii\ab_abb09.eps
    07 c:\1ud\aii\ab_abb08.eps
    07 c:\1ud\aii\ab_abb07.eps
    07 c:\1ud\aii\ab_abb06.eps
    6D c:\1ud\aii\a-ii-03n.wp

  * c:\iud\aii\a-ii-04n.chp  Wertstofferfassung [AII-14 .. 17]
    07 c:\1ud\aii\ab_abb12.eps
    07 c:\1ud\aii\ab_abb11.eps
    07 c:\1ud\aii\ab_abb10.eps
    04 c:\1ud\aii\a-ii-04n.doc

  * c:\iud\aii\a-ii-05.chp   Problemstoffe [AII-18 .. 19]
    07 C:\1UD\AII\ab_abb13.eps
    22 c:\1ud\aii\a-ii-05.doc

  * c:\iud\aii\a-ii-06.chp   Sonderabfall [AII-20 .. 23]
    07 C:\1UD\AII\ab_abb13.eps
    22 c:\1ud\aii\a-ii-05.doc

  * c:\iud\aii\a-ii-07.chp   Altlasten [AII-24 .. 29]
    07 c:\1ud\aii\al_abb05.eps
    07 c:\1ud\aii\al_abb04.eps
    07 c:\1ud\aii\al_abb03.eps
    07 c:\1ud\aii\al_abb02.eps
```

```
    07 c:\1ud\aii\al_abb01.eps
    04 c:\1ud\aii\a-ii-07c.doc
    01 c:\1ud\aii\a-ii-07a.txt
    01 c:\1ud\aii\a-ii-07b.txt

  * c:\iud\aii\a-ii-lit.chp  Literatur [AII-30]
    22 c:\1ud\aii\a-ii-lit.doc

------------------------------------------------------------------------
BI - LUFT   c:\iud\bi\*.pub fehlt
<--

  * c:\iud\bi\b-i-inh.chp   Inhalt [BI-1]
    - Die Datei D:\1UD\CIII\ENVIRON.WID fehlt, jedoch wird
      Das Inhaltsverzeichnis richtig angezeigt.
    04 c:\1ud\bi\b-i-inh.doc

  * [BI-2 .. 28] d.h. das ganze Kapitel fehlt !!
<--

------------------------------------------------------------------------
BII - Wasser    c:\iud\bii\b-ii.pub

  * c:\iud\bii\b-ii-inh.chp     Inhalt  [BII-1]
    04 c:\1ud\bii\b-ii-inh.doc

  * c:\iud\bii\b-ii-01a.chp     Wasser  [BII-2 .. 3]
    07 c:\1ud\bii\ws_abb01.eps
    07 c:\1ud\bii\wa_abb01.eps
    04 c:\1ud\bii\b-ii-01a.doc

  * c:\iud\bii\b-ii-02r.chp      Hoch- und Niedrigwasser ...
[BII-4 .. 7]
    09 C:\1UD\BII\wf-abb02.wmf
    22 c:\1ud\bii\b-ii-02a.doc
    07 c:\1ud\bii\wf-abb01.eps
    07 c:\1ud\bii\wf-abb03.eps


  * c:\iud\bii\b-ii-03.chp      Wasserversorgung [BII-8 .. 9]
    09 c:\1ud\bii\wv_abb03.wmf
    09 c:\1ud\bii\wv_abb01.wmf
    09 c:\1ud\bii\wv_abb02.wmf
    04 c:\1ud\bii\b-ii-03.doc
```

```
  * c:\iud\bii\b-ii-05a.chp     Grudwasserberschaffenheit
[BII-10 .. 19]
    - Abb. 3 bis 8 fehlen
<--
    09 C:\1UD\BII\gw-abb01.wmf
    07 c:\1ud\bii\gw-abb02.eps
    22 c:\1ud\bii\b-ii-05a.doc

  * c:\iud\bii\b-ii-04a.chp     Abwasserbehandlung [BII-20 ..
21]
    09 c:\1ud\bii\aw_abb01.wmf
    09 c:\1ud\bii\aw_abb02.wmf
    09 c:\1ud\bii\aw_abb03.wmf
    22 c:\1ud\bii\b-ii-04a.doc

  * c:\iud\bii\b-ii-06a.chp     Fiessgewaesser [BII 22 .. 24]
    - Abbildungen auf Seite BII-25 und BII-26 fehlen.
<--
      Es wurden auch keine Leerseiten dafuer gemacht.
    09 c:\1ud\bii\fg_abb02.wmf
    09 c:\1ud\bii\fg_abb01.wmf
    04 c:\1ud\bii\b-ii-06a.doc

  * c:\iud\bii\b-ii-07a.chp     Chemisch- Phy. Beschaffenheit
[BII-27 .. 33]
    07 c:\1ud\bii\fg_abb04.eps
    07 c:\1ud\bii\fg_abb10.eps
    07 c:\1ud\bii\fg_abb09.eps
    07 c:\1ud\bii\fg_abb08.eps
    07 c:\1ud\bii\fg_abb07.eps
    07 c:\1ud\bii\fg_abb06.eps
    07 c:\1ud\bii\fg_abb05.eps
    07 c:\1ud\bii\fg_abb03.eps
    04 c:\1ud\bii\b-ii-07a.txt

  * c:\iud\bii\b-ii-08.chp     Sedimente [BII-34 .. 35]
    09 c:\1ud\bii\fg_abb12.wmf
    09 c:\1ud\bii\fg_abb11.wmf
    22 c:\1ud\bii\b-ii-08.doc

  * c:\iud\bii\b-ii-09.chp      Naturfremde org. Stoffe [BII-36
.. 39]
    09 C:\1UD\BII\fg_abb14.wmf
```

```
    09 c:\1ud\bii\fg_abb13.wmf
    04 c:\1ud\d\fliessg2\b-ii-09.txt

  * c:\iud\bii\b-ii-10.chp      Bodensee [BII-40 .. 41]
    - Abb. 1 fehlt
<--
    07 c:\1ud\bii\bs_abb03.eps
    09 c:\1ud\bii\bs_abb02.wmf
    04 c:\1ud\bii\wag_01.doc

  * c:\iud\bii\b-ii-11.chp      Chemische Parameter [BII-42 ..
45]
    07 c:\1ud\bii\bs_abb06.eps
    07 c:\1ud\bii\bs_abb05.eps
    09 c:\1ud\bii\bs_abb04.wmf
    07 c:\1ud\bii\bs_abb07.eps
    04 c:\1ud\bii\b-ii-11.doc

  * c:\iud\bii\b-ii-12.chp      Pestizide im Bodensee [BII-46
.. 47]
    07 c:\1ud\bii\bs_abb08.eps
    22 c:\1ud\d\b-ii-12.doc

  * c:\iud\bii\b-ii-13.chp      Phytoplankton [BII-48 .. 49]
    07 c:\1ud\bii\bs_abb09.eps
    04 c:\1ud\bii\b-ii-13.doc

  * c:\iud\bii\b-ii-15.chp      Chemisch ...  [BII-50 .. 51]
    07 c:\1ud\bii\ks_abb01.eps
    22 c:\1ud\d\bodensee\kl-seen2.doc
    04 c:\1ud\bii\b-ii-00.doc

  * c:\iud\bii\b-ii-lit.chp     Literatur  [BII-52]
    c:\1ud\bii\b-ii-lit.doc

--------------------------------------------------------------------------------
BIII - BODEN    c:\iud\biii\b-iii.pub

  * c:\iud\biii\b-iii-ih.chp    Inhalt [BIII-1]
    c:\1ud\biii\b-iii-ih.doc

  * c:\iud\biii\b-iii-01.chp    Erfassung und Ueberwachung ...
[BIII-2 .. 3]
```

```
    - Abb. 2 fehlt
<--
    07 c:\1ud\biii\bo_abb01.eps
    04 c:\1ud\biii\b-iii-01.wrd

  * c:\iud\biii\b-iii-02.chp    Anorganische Schadstoffe
[BIII-4 .. 7]
    - Abb. 3 und 4 fehlen
<--
    07 c:\1ud\biii\bo_abb5b.eps
    07 c:\1ud\biii\bo_abb5a.eps
    22 c:\1ud\biii\b-iii-02.doc
    04 c:\1ud\biii\b-iii-02.wrd

  * c:\iud\biii\b-iii-03.chp    Organische Schadstoffe [BIII-8
.. 9]
    - Abbildung auf Seite BIII-9 fehlt
<--
    04 c:\1ud\biii\schoe02.doc
    04 c:\1ud\biii\b-iii-03.doc
    07 c:\1ud\biii\bo_abb06.eps

  * c:\iud\biii\b-iii-04.chp    Dioxine im Boden  [BIII-10 ..
15]
    - Abbildungen auf Seite BIII-11 und BIII-15 fehlen.
<--
    04 c:\1ud\biii\b-iii-04.txt
    04 c:\1ud\biii\dioxine.txt

  * c:\iud\biii\b-iii-05.chp    Verkehrsbedingte Immisionen
[BIII-16]
    07 c:\1ud\biii\st_abb01.eps
    07 c:\1ud\biii\strass_2.eps
    22 c:\1ud\biii\b-iii-05.doc


-------------------------------------------------------------------------
CI - LEBENSMITTEL   c:\iud\ci\c-i.pub

  * c:\iud\ci\c-i-inh.chp       Inhalt [CI-1]
    04 C:\1UD\CI\C-I-INH.DOC

  * c:\iud\ci\c-i-01.chp        Lebensmitteluebeberwachung [CI-2
.. 11]
```

```
    - Abb. 6 und 7 fehlen
<--
    07 c:\1ud\ci\lm_abb02.eps
    07 c:\1ud\ci\lm_abb12.eps
    07 c:\1ud\ci\lm_abb10.eps
    07 c:\1ud\ci\lm_abb08.eps
    07 c:\1ud\ci\lm_abb05.eps
    07 c:\1ud\ci\lm_abb03.eps
    07 c:\1ud\ci\lm_abb01.eps
    07 c:\1ud\ci\lm_abb04.eps
    08 c:\1ud\ci\c-i-01.txt

  * c:\iud\ci\c-i-02.chp        Leichtfluechtige chlorierte ...
[CI-12 .. 22]
    - Abb 14 und 15 fehlen
<--
    07 c:\1ud\ci\lm_abb13.eps
    07 c:\1ud\ci\lm_abb26.eps
    07 c:\1ud\ci\lm_abb25.eps
    07 c:\1ud\ci\lm_abb23.eps
    07 c:\1ud\ci\lm_abb22.eps
    07 c:\1ud\ci\lm_abb21.eps
    07 c:\1ud\ci\lm_abb19.eps
    07 c:\1ud\ci\lm_abb18.eps
    07 c:\1ud\ci\lm_abb17.eps
    07 c:\1ud\ci\lm_abb16.eps
    07 c:\1ud\ci\lm_abb14.eps
    07 c:\1ud\ci\lm_abb20.eps
    08 c:\1ud\ci\c-i-02.txt

-------------------------------------------------------------------------
CII - RADIOAKTIVITAET   c:\iud\cii\*.pub  fehlt
<--

  * hier fehlt alles !! von [CII-1 .. 18]
<--

-------------------------------------------------------------------------
CIII - LAERM    c:\iud\ciii\c-iii.pub

  * c:\iud\ciii\ih-ciii.chp    Inhalt  [CIII-1)
    04 c:\1ud\ciii\ih_ciii.doc
```

```
  * c:\iud\ciii\c-iii.chp      Laerm als Belastung   [CIII-2 ..
17]
    - Ich konnte nicht alle Grafiken laden, da angeblich
      nicht genug Speicher vorhanden war.
    07 c:\1ud\ciii\lr_tab04.eps
    07 c:\1ud\ciii\lr_abb24.eps
    07 c:\1ud\ciii\lr_abb23.eps
    07 c:\1ud\ciii\lr_tab03.eps
    07 c:\1ud\ciii\lr_abb22.eps
    07 c:\1ud\ciii\lr_abb21.eps
    07 c:\1ud\ciii\lr_abb20.eps
    09 c:\1ud\ciii\lr_abb19.wmf
    07 c:\1ud\ciii\lr_abb18.eps
    07 c:\1ud\ciii\lr_abb16.eps
    07 c:\1ud\ciii\lr_abb15.eps
    07 c:\1ud\ciii\lr_abb17.eps
    09 c:\1ud\ciii\lr_abb14.wmf
    09 c:\1ud\ciii\lr_abb04.wmf
    07 c:\1ud\ciii\lr_abb13.eps
    07 c:\1ud\ciii\lr_abb12.eps
    07 c:\1ud\ciii\lr_tab02.eps
    07 c:\1ud\ciii\lr_abb10.eps
    07 c:\1ud\ciii\lr_abb09.eps
    07 c:\1ud\ciii\lr_tab01.eps
    07 c:\1ud\ciii\lr_abb07.eps
    07 c:\1ud\ciii\lr_abb06.eps
    07 c:\1ud\ciii\lr_abb05.eps
    07 c:\1ud\ciii\lr_abb03.eps
    07 c:\1ud\ciii\lr_abb02.eps
    07 c:\1ud\ciii\lr_abb01.eps
    04 c:\1ud\ciii\c-iii.doc


-------------------------------------------------------------------------
CIV - WALD     c:\iud\civ\*.pub  fehlt
<--

  * c:\iud\civ\c-iv-inh.chp   Inhalt  [CIV-1]
  * c:\iud\civ\walt1.chp      Waldbesitzartenverteilung  [CIV-2
.. 20]
    - Dokument stimmt nicht mit der gedruckten Ausgabe
ueberein.         <--

-------------------------------------------------------------------------
```

```
CV - NATUR UND LANDSCHAFT   c:\iud\cv\cv.pub

  * c:\iud\cv\cv-inh.chp      Inhalt [CV-1]
    04 c:\1ud\cv\cv-inh.doc

  * c:\iud\cv\cv-01.chp       Gesamtkonzept ...  [CV-2 .. 16]
<--
    - Aus Seite CV-3 fehlt Text. Die weiteren
      Seiten stimmen nicht mit dem gedruckten
      Dokument ueberein. Es fehlt vorallen Text.
      ?* c:\iud\cv\natlan01.chp Dies hat den gleichen Inhalt
wie cv-01.chp
         und liegt auch naeher an dem gedruckten Bericht.
Jedoch
         stimmt auch dies nicht mit dem Bericht ueberein.
      ?* c:\iud\cv\natlan02.chp liegt noch naeher am Orginal,
stimmt jedoch
         auch nicht richtig ueberein. Weiterhin geht das
Dokument nur
         bis [CV-13].
      ?* c:\iud\cv\natlan03.chp hat keinen Header und beginnt
irgendwo.
    07 c:\1ud\cv\nl_abb11.eps
    07 c:\1ud\cv\nl_abb06.eps
    07 c:\1ud\cv\nl_abb09.eps
    07 c:\1ud\cv\nl_abb05.eps
    07 c:\1ud\cv\nl_abb07.eps
    07 c:\1ud\cv\nl_abb04.eps
    07 c:\1ud\cv\nl_abb03.eps
    07 c:\1ud\cv\nl_abb02.eps
    07 c:\1ud\cv\nl_abb01.eps
    22 c:\1ud\cv\cv-01.doc

--------------------------------------------------------------------------------
DI - UMWELTOEKONOMIE    c:\iud\di\d.pub

  * c:\iud\di\d-inh.chp     Inhalt  [D-1]
    04 C:\1UD\DI\D-INH.DOC

  * c:\iud\di\d-i.chp       Umweltoekonomie       [D-2 .. 13]
    - Ab Seite D-10 sind leichte Differenzen
<--
      vorhanden.
    07 C:\1UD\DI\OE_ABB14.EPS
```

```
07 C:\1UD\DI\OE_ABB17.EPS
07 C:\1UD\DI\OE_ABB16.EPS
07 C:\1UD\DI\OE_ABB15.EPS
07 C:\1UD\DI\OE_TAB01.EPS
07 C:\1UD\DI\OE_ABB12.EPS
07 C:\1UD\DI\OE_ABB13.EPS
07 C:\1UD\DI\OE_ABB11.EPS
07 C:\1UD\DI\OE_ABB10.EPS
07 C:\1UD\DI\OE_ABB09.EPS
07 C:\1UD\DI\OE_ABB07.EPS
07 C:\1UD\DI\OE_ABB08.EPS
07 C:\1UD\DI\OE_ABB06.EPS
07 C:\1UD\DI\OE_AB06A.EPS
07 C:\1UD\DI\OE_ABB03.EPS
07 C:\1UD\DI\OE_ABB05.EPS
07 C:\1UD\DI\OE_ABB04.EPS
07 C:\1UD\DI\OE_ABB01.EPS
07 C:\1UD\DI\OE_ABB02.EPS
04 C:\1UD\DI\U1.DOC
```

Literaturverzeichnis

[Dr.93] Dr. Lippke und Dr. Wagner GmbH.
Benutzerhandbuch für den Umwelt-Datenkatalog UDK - Entwurf.
Niedersächsisches Umweltministerium, Berlin, 1993.

[Ian94] Ian Graham.
HTML Documentation.
Instructional and Research Computing, Computing and Communications, University of Toronto , Juni 1994.

[Klu94a] Rainer Klute.
Gut in Form.
IX Multiuser, Multitasking Magazin Seiten 186–189, Oktober 1994.

[Klu94b] Rainer Klute.
Sensitive Bilder.
IX Multiuser, Multitasking Magazin Seiten 190–192, Oktober 1994.

[Klu94c] Rainer Klute.
Zusammengewebt.
IX Multiuser, Multitasking Magazin Seiten 150–157, Februar 1994.

[Nat94] National Center for Supercomputing Applications, University of Illinois, Urbana .
HTML Primer, Dezember 1994.

[Nie94] Niedersächsisches Umweltministerium, Bundesminsterium für Umwelt, Jugend und Familie Republik Österreich.
Umweltdatenkatalog Benutzerhandbuch, Juni 1994.