Experiences in Document Conversions Performed during the MeDoc Project

Hubert Feyrer <hubert.feyrer@rz.fh-regensburg.de>

March 17, 1998

Abstract

This document summarizes the experiences made in converting electronically available documents in various formats for use in the MeDoc project's electronic library. After giving an overiew of available formats and tools to manipulate them, choices are outlined for design questions that will arise during document format conversion. Guidelines are given to straighten the conversion process described next as much as possible. Further ways of improvement are discussed at the end of this document.

Contents

1 Introduction: Value of document format conversions for electronic libraries **4**
- 1.1 Need for a Common Document Format . . . 4
- 1.2 Possible Common Formats . . . 4
- 1.3 Drawbacks of Using a Common Format . . . 4
- 1.4 The Way to Go . . . 4

2 Document Formats **5**
- 2.1 Overview . . . 5
- 2.2 Target Formats . . . 6
  - 2.2.1 HTML . . . 6
  - 2.2.2 PDF . . . 7
- 2.3 Intermediate Formats . . . 7
  - 2.3.1 PostScript . . . 8
  - 2.3.2 DVI . . . 8
  - 2.3.3 RTF . . . 9
- 2.4 Source Formats . . . 9
  - 2.4.1 LaTeX . . . 9
  - 2.4.2 Microsoft Word for Windows . . . 10
  - 2.4.3 FrameMaker . . . 11

3 Tools: converters and others **11**
- 3.1 Intro . . . 11
- 3.2 Word Internet Assistant . . . 12
- 3.3 latex2html . . . 13
- 3.4 frame2html . . . 14
- 3.5 WebWorks HTML Lite . . . 15
- 3.6 rtf2html . . . 16
- 3.7 TeX . . . 17
- 3.8 dvips . . . 18
- 3.9 Adobe Distiller . . . 19
- 3.10 Adobe Exchange . . . 20
- 3.11 ghostscript . . . 23
- 3.12 HotDog Pro . . . 24
- 3.13 FrontPage . . . 25

4 Guidelines for Document Conversion **26**
- 4.1 Target Format . . . 26
- 4.2 Language . . . 27
- 4.3 Fonts . . . 27
- 4.4 Colors . . . 28
- 4.5 Use of Images . . . 29
- 4.6 Document Segmentation . . . 30
- 4.7 Design of the Navigation Bar . . . 30

4.8 Links . . . 31
4.9 Table of Contents . . . 34
4.10 References . . . 37
4.11 Index . . . 38
4.12 Titlepage . . . 39

5 Communication and Exchange of Experiences 39

6 Workflow of a Conversion 39
6.1 Obtaining Sources . . . 39
6.2 Specifying the Goal . . . 41
6.3 Intercepting the Converter . . . 41
6.4 Feedback . . . 42
6.5 Document delivery . . . 42

7 Conclusion 43

1 Introduction: Value of document format conversions for electronic libraries

1.1 Need for a Common Document Format

Libraries have to keep their contents in a uniform format to facilitate reading, indexing and searching for their users. Also, the choice has to be done depending on these users' needs. And with a library as open to its customers as the MeDoc library, a whole variety of platforms have to be adressed as clients to the library. So the format has to be something that can be easily read on any of those platforms, with enforcing no or only small restrictions on these clients. Only one format has to be chosen, to avoid redundancy and inconsistencies.

The format(s) used for the library have to be open so implementing services for reading and searching the texts are possible for the users on any platforms, and noone is forced to use (and pay!) any proprietary product ported to his platform due to the marketing strategies of a single vendor.

1.2 Possible Common Formats

Open standards that are in common use today and that are available on a wide range of platforms are SGML (ISO 8879) and HTML (URL: http://www.w3-.org/) as document description languages as well as PostScript and its successor, the Portable Document Format (PDF), both of which belong to the category of page description languages/formats.

For the MeDoc library, HTML and PDF have been chosen as the formats to be used, see [1].

1.3 Drawbacks of Using a Common Format

But hardly any book or text today is written in HTML, not to speak of hand-coded PDF. Rather, text-processors ranging from rather archaic to high-tech are used. Some of these text-processors do offer converting into HTML and/or PDF, and some don't, so one probably has to go a more complicated way in reaching the wanted target format. And even if there's a straight way for conversion, today's tools aren't always 100% perfect. Bugs and immaturity often prevent correct conversion into the target format in a single step, so it's often necessary to help these tools, sometimes even fixing what they broke.

1.4 The Way to Go

As well as there's a whole lot of document formats to convert from, via and to, there are several tools and techniques that can be used to convert a document into the given format. Besides that, often the author of the document can help to ease the conversion by paying attention to the fact that the book isn't (only) going to be printed on paper.

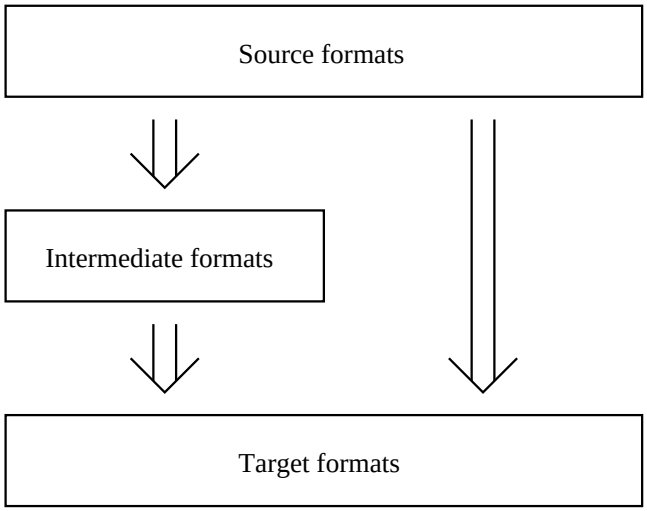

Figure 1: The three groups of document formats and their relationship

This report is about all these formats, techniques, their interchangeability and application.

2 Document Formats

2.1 Overview

There are quite a lot of different electronic document formats, and to make it easier to describe their common properties they can be divided into three groups, as shown in Figure 1

Target formats are the ones which existing documents are converted to and which are used in keeping documents for reading and other usage. Existing documents are produced in one of several source formats, usually by the author of a book, using some text processor. Some of these source formats can be brought into the desired target format either directly or via some intermediate format that can then be converted into the final (target) format in a second step.

I will explain the formats addressed in Figure 1 from bottom up. Starting with the target formats, via some intermediate document formats to the formats in which the document source is most often available today. The next section will go and describe the converters used to transform the various formats in greater detail.

For every single document format, I will discuss the following attributes:

- *General description*

- *Type of standardisation:* Proprietary, vendor dependent or open (independent) standard?
- *Advantages / Disadvantages:* Given under the special aspect of document conversion
- *Structure:* Are the contents of the document human readable or is special software required for handling it? For documents of the first type it's usually easier to manipulate them if some tool breaks something. Does the document format support structured text so that parts of it can be extracted easily? Is the document self-contained, or does it require any additional data not contained it itself, e.g. for fonts?
- *Tools:* A list of programs which can read, write and manipulate this document format.

2.2 Target Formats

This section describes document formats that can be read by many users as they are available on a wide range of differenc platforms.

2.2.1 HTML

General description: The Hypertext Markup Language was introduced by CERN. Designed as a "markup" language derived from SGML, HTML didn't have many layout capabilities and features for graphical design. Many of these features have been added in the mean time, making it also a suitable document format not only for documents that consist of text mainly, but also for fully designed, optically very appealing publications directed toward commercial advertizement.

Efforts have been taken to include not only layout improvements but also support a wider range of documents, such as foreign (non-latin) charsets, and especially mathematical formulas used in scientific publications. Unfortunately, these efforts were only written down in form of a standard, but they are not implemented by any browser commonly used today.

Type of standardisation: Open standard by the W3-Consortium.

Advantages: HTML is a plain text format that can easily be processed even with handmade programs, no special set of tools has to be used to manipulate any HTML file. Often a simple text editor does the job.

Disadvantages: When someone wants to display something not supported by the HTML charset such as formulas, this has to be included as an image, resulting in *many* images together with all their disadvantages: quite big, slow to load and can't be resized easily.

Structure: HTML is manifested as tagged ASCII text, derived after a special Document Type Description as noted by the SGML standard. The single text files do not include any information on embedded images other as references to these files. It's a HTML-browser's job to retrieve these files as well.

Tools: Many freely and commercially available editors (basically every text editor; HotDog Pro; Microsoft FrontPage), converters (Word Internet Assistant, frame2html, latex2html, Quadralay WebWorks Publisher, ...) and browsers (mainly Netscape's Navigator/Communicator and Microsoft's Internet Explorer).

2.2.2 PDF

General description: The Portable Document Format is the successor of the PostScript page description and printer language, both invented by Adobe.

Type of standardisation: PDF is an open standard by Adobe, the Portable Document Format Reference Manual Version, 1.2 is freely available, see [12].

Advantages: Besides the very good mechanisms regarding page description, graphical and color operations taken from PostScript, PDF provides document navigation within a document and to outside documents as well as to the WWW, links to texts, pictures, animations and external programs as well as defining fill-out-forms. Saved documents can be secured by a password for reading, printing, editing, etc..

Disadvantages: Very few tools exist to manipulate documents available in PDF.

Structure: Although the PDF specification allows documents to be in plain ASCII format as well, PDF documents are usually saved in the much smaller binary representation, making it impossible to modity them without the proper tools. Any PDF document includes all image information needed to display the document. Common fonts are not included (by default) in the document, but can be requested to do so as well.

Tools: Mandatory tools for working with PDF are Adobe's Distiller (for converting PostScript to PDF), Exchange (for later manupulating PDF files) and Reader (PDF-browser). Freely available PDF-viewers are xpdf and GhostScript, with the latter also being capable of generating PDF documents from PostScript to a certain extent.

2.3 Intermediate Formats

The formats described here are used if no straight conversion from a source to a target format can be made.

2.3.1 PostScript

General description: PostScript as itself is a very powerful graphical programming language for describing objects on a page, and manipulating them through various filter operations, scaling, rotating, clipping, etc. Documents in PostScript are resolution independent as long as no pixel-images or -fonts are included, resulting in very high quality print output.

Type of standardisation: The PostScript language is described in the three books by Adobe, known as the "blue book"[2], "big red book"[3], and "green book"[4] (called so according to their jacket colors).

Advantages: Excellent graphical capabilities, with PostScript Level 2 even in color, resolution indepent.

Disadvantages: No textural structure is retained in the documents, thus making it difficult to extract text from PostScript documents.

Structure: PostScript itself is ASCII readable, but to manipulate an existing document one needs a very good working knowledge of the PostScript language to manipulate documents by hand.

Tools: Adobe Distiller, GhostScript, "print to file" together with a PostScript-printer-driver for any text processor, pstools.

2.3.2 DVI

General description: DVI-files are "DeVice Independent", i.e. the contents they contain are independent of the resolution of the intended output device. They are usually[1] produced by TeX and subsequently converted into PostScript.

Type of standardisation: No official standard, the format is defined by the original Web code by Donald Knuth's TeX distribution, which is unique on all systems.

Advantages: As the name says, DVI-files contain no information depending on the resolution of the output device and thus may be converted further without loss in display quality.

Features not implemented in TEX itself can still be embedded in the form of `\special`-commands, which are then processed by some device driver. Hypertext links are carried from LATEX to PDF with this mechanism.

[1] The GNU project's roff implementation, groff, can output files in DVI format, too.

Disadvantages: This is only output by TEX, and it should be really only be viewed as a format is subsequently converted to PostScript, as there are no editors available for DVI files.

Structure: The internal structure of a DVI file is in binary format. Fonts are required to be kept external as they are resolution dependent as used in the TEX world.

Tools: TEX, dvips, groff.

2.3.3 RTF

General description: The Rich Text Format was created by Microsoft to provide a document format that can be exchanged with other platforms and word processors easily. Today, few word processors support this document format, although it's the native document format for documents created by tools of the NeXTstep operating systems.

Type of standardisation: Open standard by Microsoft, see [11].

Advantages: RTF is very simple and intended mainly for textual information, although it provides capabilities for embedded images.

Disadvantages: No extended formating facilities are supported, formulas etc. are not supported.

Structure: RTF is an ASCII format, which also embeds any images that are contained in the document.

Tools: MicroSoft's FrontPage, rtf2html, word-processors with RTF-output (e.g. Word for Windows).

2.4 Source Formats

The source formats described here represent a small choice of what's in common use today, but the ones listed here are the ones most commonly used during the document conversions of the MeDoc project.

2.4.1 LATEX

General description: LATEX is a complex typesetting system invented in the 70s by Donald E. Knuth. It has very few limitations, especially designed for producing high quality printouts of mathematical and scientific documents. With the aid of so-called style-files, many additional types of contents can be described such as syntax diagrams, musical notes, etc.

Type of standardisation: The TEX core system is described in Knuth's "TEX Book"[9], which still is the authoritative source for any modifications made to the sources.

Further development of TEX, LATEX and web2c is coordinated by the TEX User Group. See http://www.tug.org/ for more information.

Advantages: TEX has very strong capabilities for typesetting mathematical formulas and non-standard fonts. The resulting output file (DVI-file, see above) is independent of the resolution of the output device, thus resulting in a very high quality of the document when printed or displayed on screen. A further advantage is the virtually unlimited extensibility through style files.

Disadvantages: The core system of TEX is basically a programming language, and therefore a lot of dirty things can be made by a document's author. Bad luck for the converter of such a document if he can't figure out what's going on. ;-)

Structure: TEX's input file is plain (7bit) ASCII text, with references to some external style descriptions that define the actual layout. The metrics of the fonts (with information on kerning, ligatures, ...) are included from external files (TFM-files), that only contain information on the metrics of the fonts, not their actual data, therefore being resolution independent.

Tools: Basically any text editor can be used to edit TEX-files, but also dedicated typesetting systems that base on TEX, such as Lyx and the commercial tool The Publisher.

2.4.2 Microsoft Word for Windows

General description: The WinWord document format supports (of course) everything that its creator Microsoft's "Word for Windows" provides, including images, formulas, tables, etc.

Type of standardisation: Proprietary, changing with every release of the software.

Advantages: Mostly complete for every possible type of document-contents known to mankind.

Disadvantages: Proprietary format very likely incompatible to itself in the next version.

Structure: Documents usually contain everything they need, but references to all kind of external links can be embedded as well. The documents themselves are in a binary representation that makes it impossible to decode them easily with self-written scripts.

Tools: Many, free and commercial. Mostly Microsoft's "Word for Windows". Importing Word-documents into FrameMaker is possible with some loss of information.

2.4.3 FrameMaker

General description: Powerful text processor especially aimed towards technical documentation, i.e. it has support for formulas, tables, technical drawings, images, as well as sophisticated tools for creating indices of all kinds.

Type of standardisation: Proprietary, document format description can be purchased via the Frame developer program.

Advantages: The document format has been extended between releases, but backward compatibility has been addressed properly, making it possible for documents created with the old version (V4) to be processed with the new version (V5) without any loss of data. Printing output is done in PostScript, if requested it can also generate PDF-marks for further conversion into PDF, so that cross references, table of contents and index are hyperlinked properly.

Disadvantages: The program is very expensive.

Structure: FrameMaker can save its documents in two formats. The native FrameMaker format is small and compact, because of its binary representation. Better for any conversion jobs is the "Maker Interchange Format" (MIF), a tagged ASCII format similar to SGML.

Tools: FrameMaker, frame2html, Quadralay's WebWorks Publisher and HTML Lite.

3 Tools: converters and others

3.1 Intro

Having that many different document formats asks for a whole lot of programs for converting one into another, and doing other manipulations on documents in any of these formats. The availability and quality of such tools for a given format also make it more or less easy to deal with that format.

This section tries to give a list of the most commonly used tools that are available for manipulating the document formats described in the previous chapter, concentrating on the following aspects for each tool:

- *Where to get:* Postal and/or email address, URL of web page and/or FTP-site, if freely available
- *Supported formats:* List of which formats the described tool can read and write.
- *Pros and Cons:* Description of the tool's strenghts and flaws
- *Tips & Tricks:* Often some features are non-obvious; if there are some such for the given tool, they are listed here.
- *Fixes:* Some of the publically available tools can (and have) to be tuned to produce proper output. Sometimes it's even necessary to modify the source code. If so, references to these patches are listed here.
- *Other tools:* List of other tools which can be used together with the program described here.

3.2 Word Internet Assistant

Word Internet Assistant is an add-on for Microsoft's Word for Windows (WinWord) which allows export of a Word-document to HTML from within WinWord.

Where to get:

- WWW: http://www.microsoft.com/word/internet/ia/
- FTP: ftp://ftp.htwk-leipzig.de/fachbereiche/imn/software/inetAssi/iaword6.exe
- FTP: ftp://ftp.htwk-leipzig.de/fachbereiche/imn/software/inetAssi/iaword7.exe

Supported formats:
Input:

- Every format that can be imported into Word for Windows.

Output:

- HTML

Pros and Cons: The output generated by this converter module depends a great deal on the input present in the word document. If the markup tags aren't mapped properly, headings are just marked with a big, bold font, but not with the appropriate tags for headings in HTML (<H1>, <H2>, ...), making it difficult if any post-processing of the output is necessary. It can be said in general that the conversion is rather poor, because mostly only paragraph boundaries are kept during the conversion. Changes in the font size, sub- and superscript are dropped, which is especially unpleasant for scientific documents. Images imbedded in the document are not converted, neither are tables and formulas.

No document segmentation is performed by the Word-Internet-Assistant, this is left to the converter doing this by hand. No need to mention what this does to any existing links within the document.

Some of be bugs described above are fixed in the Assistant V7.0, but by far not all.

Tips & Tricks: Double-check with the paper version after conversion.

Fixes: None possible.

Other tools: -

3.3 latex2html

latex2html converts LaTeX documents into HTML, paying special attention to images, mathematical formulas, etc.

Where to get:

- FTP: http://www-dsed.llnl.gov/files/programs/unix/latex2html/sources/
- WWW: http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/

Supported formats:
Input:

- About every valid LaTeX text, independent of any styles used.

Output:

- HTML

Pros and Cons: The converter supports document segmentation, automatically adds a navigation bar, a (hyperlinked) table of contents and index as well as cross-references within the document. Formulas, and images that can't be converted into HTML are automatically converted into GIF images. Tables can be included in HTML-style or as images.

Minor probles such as only small images as anchors in links can be fixed with not too much effort by an external script, see [7].

However, when the converter needs to deal with many images e.g. when converting documents with mathematic contents, it starts mixing up these images and doesn't do it's job properly any more. Fixing this wasn't possible when attempting to, too, due to the chaotic internal structure of the converter (a set of perl scripts).

Tips & Tricks: Lots of useful tips for tuning latex2html can be found in Juergen Mayerhofer's conversion report on the first volume of the Kopka books, see [7].

Fixes: Fixes are available for various problems, see mentioned above in [7].

Other tools: latex2html requires TEX, dvips, ghostscript and NetPBM-tools to be installed.

3.4 frame2html

frame2html is a free converter from FrameMaker to HTML.

Where to get:

- FTP: ftp://ftp.nta.no/pub/fm2html/fm2html.v.0.8.9a.tar.Z

Supported formats:
Input:

- Maker Interchange Format (MIF)
- Native FrameMaker text is first converted to MIF automatically
- Frame-"Books" consisting of several files

Output:

- HTML

Pros and Cons: By using one fixed configuration file, this program is very well suited for being used in batch processing. Once the mapping of which Frame-paragraph-tag is becoming which HTML-tag is done in that config file, the number of files to convert doesn't matter. And if there isn't a precise HTML-tag that corresponds to one of the Frame-paragraph-tags, arbitrary HTML-code can be put into that place. If whole books (consisting of several Frame input files) are converted, links between these files are maintained properly. Graphics are automatically converted to GIF (with links to PostScript files for higher-resolution output), tables are converted into preformatted ASCII-tables with a link to a PostScript file displaying the table in higher quality.

One of the drawbacks is that document seperation has to be done manually, the converter generates one HTML file from each Frame-input-file. Generation of automatic indices is not supported, the converter just warns about unused index points (see below for a patch). Sub- and superscript are not supported.

About the biggest drawback of frame2html however is that it is no longer supported by the author.

Tips & Tricks: Some tips on support for index creation can be found in the conversion report of the Raasch-book[6].

Fixes: A patch to help in generating an index is available from Hubert Feyrer <hubert.feyrer@rz.fh-regensburg.de>.

Other tools: To run frame2html, ghostscript and NetPBM as well as perl V5 have to be installed.

3.5 WebWorks HTML Lite

The lite version of this converter for FrameMaker documents to HTML is part of FrameMaker as of version 5. It's available on any platform that FrameMaker runs on, i.e. Windows, Macintosh, Unix.

Where to get: WebWorks HTML Lite is a part of FrameMaker, which is available from:

Frame Technology Corporation
333 West San Carlos Street
San Jose, CA 95110-2711
USA
Tel: 408-975-6000 and 800-U4-FRAME
Fax: 408-975-6799

EMail: comments@frame.com
WWW: http://www.frame.com/

Supported formats:
Input:

- Maker Interchange Format (MIF)
- Native FrameMaker text is first converted to MIF automatically
- Frame-"Books" consisting of several files

Output:

- HTML

Pros and Cons: The Lite-version of Frame's own HTML-converter comes free with version 5, it's fully integrated into FrameMaker's menues.

The main drawback of the HTML-Lite converter is that the tag-mapping has to be done for every single file, and that this whole process is 100% GUI-driven, so one has to repeat the mapping-steps (too) many times. This makes it unusable for bigger documents. The full version of Frame's HTML-exporter does not have this disadvantage.

Tips & Tricks: -

Other tools: Quadralay WebWorks Publisher (full version of HTML Lite, send mail to info@quadralay.com for more information), frame2html, FrameMaker.

3.6 rtf2html

For converting text in RTF format into HTML, the converter rtf2html is available in two version. Version prior to 3.x are freely available in source, version 3.x and later are available as precompiled binaries only. Source is available under a Non Disclosure Agreement as well.

Where to get:

- WWW: http://www.sunpack.com/RTF/latest/

Supported formats:
Input:

- RTF

Output:

- HTML

Pros and Cons: Documents are converted as a whole or are being devided into small parts automatically (document segmentation), if requested. The conversion doesn't need any corrections, as long as no lists are used. These aren't recognized properly and must be fixed by hand. Images are embedded in the native format of the platform the RTF file was created on, i.e. Windows MetaFile if the RTF text is from Word. They must be converted into HTML manually later. However, it is possible to only place thumbnails on the HTML page and create links to the big images automatically.

Tips & Tricks: -

Fixes: None possible.

Other tools: Word for Windows, Internet Assistant for Word for Windows.

3.7 TEX

TEX is a typesetting system, transforming ASCII input with special formating commands into a device independent (DVI) file, which can then be transformed into PostScript or any other print- and display-format.

Where to get:
The core TEX system with few style files and fonts is (for Unix systems) available via anonymous FTP from ftp://ftp.tug.org/tex/. There are also complete packages available with more fonts and styles:

- teTeX is for Unix-systems, it's available from ftp://ftp.dante.de/pub/tex/systems/unix/teTeX
- emTeX is for PC-based systems, available from ftp://ftp.dante.de/pub/tex/systems/msdos/emtex

Supported formats:
Input:

- plain TEX
- LATEX 2.09
- LATEX 2e
- any of the above with a lot of different style files

Output:

- DVI

Pros and Cons: TEX is a very mature typesetting system, although the user interface is a bit archaic. Through the use of various style files, many things not implemented initially can be archieved, e.g. including hypertext-links in the resulting DVI-file.

About the only annoying problem for which there's no known solution is that TEX only produces one big output file. If one wants to segment a big documents into small files, this has to be done by hand, probably breaking things like links within the file.

Tips & Tricks: The following styles and packages have been useful when producing a DVI-file that will be carried on to PDF later:

- ulem.sty: TEX usually doesn't hyphenate words or break up lines that are `\underline`'d. This style defines a new `\uline` command (plus some others), with which correct hypenation and line breaks are implemented.
- mathptm.sty: Replaces math-fonts with PostScript-versions
- times.sty: Replaces regular fonts with PostScript-versions
- hyperref.sty: inserts commands that are translated into PDF-links with a decent dvips. Useful options to this style are:
  - `pdfmark`: Needed for generating bookmarks
  - `hyperindex`: Create hyperlinks for index
  - `backref`: In the bibliography, provide links back to the places where a certain book was references.

Fixes: None needed.

Other tools: dvips

3.8 dvips

Dvips is used to convert DVI-files created by TEX into PostScript-code. If embedded hyperlink-information is to be transformed into link-information for further conversion to PDF, only one of the latest dvips' can be used.

Where to get:

- WWW: http://www.radicaleye.com/dvips.html
- FTP: ftp://labrea.stanford.edu/pub/

Supported formats:
Input:

- DVI

Output:

- PostScript

Pros and Cons: None - what can one say about a tool that just does the job it's intended to, without any big bells and whistles, but also without any bugs. It just works. :)

Tips & Tricks: If you want to create PDF-documents from the output generated by dvips, you have to specify the "-z"-switch to do so. This switch isn't available in older dvips', and thus it's a way to check if your dvips is up to date.

Fixes: None needed.

Other tools: TEX, Adobe Distiller, ps2pdf (part of ghostscript)

3.9 Adobe Distiller

Adobe's "Distiller" is *the* tool to convert PostScript into PDF. Being made by the creators of the PDF and PostScript formats, it's the best utility for creating PDF from PostScript available today.

Where to get: Adobe Distiller is available for Windows, Macintosh and some Unix platforms. Use the following address for forther information:

Adobe Shop
Adobe Systems Direct Ltd.
Abteilung R.M.I.
Postfach 10 29 65
D-50469 Köln
Tel: (+49)130.81 74 73
Fax: (+44)131.451 6999

WWW: http://www.adobe.com/prodindex/acrobat/main.html

Supported formats:
Input:

- PostScript Level 1 & 2

Output:

- Portable Document Format

Pros and Cons: Distiller provides many options that can be tuned in the conversion process, amongst them are setting if fonts are to be embedded, if the output is ASCII or binary, choosable default resolution, if and how to compress color, greyscale and monochrome bitmap images (available compression modes are JPEG, ZIP and CCITT (FAX) in various qualities), as well as instructions on whether colors in the generated PDF file are device independend (and thus more accurate) or device dependent (which are faster to display).

For use in batch mode, a list of directories can be given to watch, and if a file is copied into one of these directories, that file will be converted into PDF. It is also possible to use different conversion options for different directories.

Tips & Tricks: Switching on all available methods for image-compression can reduce the output file to a huge extent.

Fixes: None possible.

Other tools: Any program generating PostScript output; dvips; ghostscript; Adobe Exchange.

3.10 Adobe Exchange

Adobe Exchange is to editing PDF documents what Adobe Distiller is to conversion from PostScript to PDF - the ultimate tool. It facilitates all features of PDF, and through it's graphical user interface, it provides easy access to all features it offer.

Where to get: Adobe Distiller is available for Windwows, Macintosh and some Unix platforms. Use the following address for forther information:

Adobe Shop
Adobe Systems Direct Ltd.
Abteilung R.M.I.
Postfach 10 29 65
D-50469 Köln
Tel: (+49)130.81 74 73
Fax: (+44)131.451 6999

WWW: http://www.adobe.com/prodindex/acrobat/main.html

Supported formats:
Input:

- PDF

Output:

- PDF

Pros and Cons All of the features available in PDF documents are made available through this program. Linking to various types of objects (PDF-documents, images, movies and sounds, intra-document-links, web-links), generating forms for user input, creating PDF documents from scanned images, making whole documents searchable through an index, etc.

About the only drawback is that there is no alternative to this program, so one has to use it for editing PDF documents. And if one's platform of choice tends to be Unix, this is a very expensive choice.

Tips & Tricks:

- Save: better "Save as"

  When changing a PDF file, Exchange just appends this changes to the existing PDF file, thus growing it even if you delete something. To avoid this, always use "Save as" to create a new file, that only contains what's necessary.

- Don't forget optimizing

  As of version 3.0, Exchange provides the facility to "byteserve" only few pages requested by a web client, if the webserver in use supports this. (The client also needs a special plug-in). To make a document ready for byteserving, the "optimize" switch has to be checked when saving a document. See http://www.adobe.com/prodindex/acrobat/byteserve.html for more details on byteserving.

- Impact of screen-resolution on centering

  When creating links within a document by hand, one should keep in mind that Exchange saves the position of the first line visible on the screen, and then displays everything down as far as resolution permits. Thus, centering e.g. headings and images may result in surprises when one views a document created on a high-resolution display on a low-resolution one, as illustrated by Figure 2.

  A better solution is to set links so that the object linked to does appear in the upper left corner (as far as possible), as this position is the same on all displays.

- View-Menue: Continuous

  A problem similar to the above arises, when a link is set referencing an object near the bottom of a page. Doing so might also result in the problem described above. Solving this problem by scrolling the view of the page further down isn't possible (with the default settings), as this would switch over to the next page entirely. Instead, setting scrolling from "Single Page" to "Continuous" in the "View"-menu makes the next page scroll in instead of switching to it, as displayed by Figure 4.

Fixes: None possible.

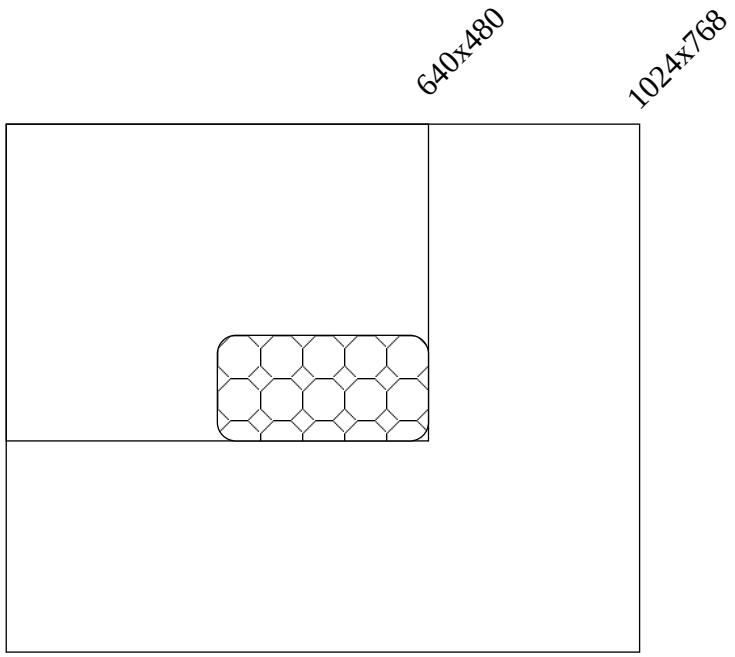

Figure 2: Bad: impact of screen resolution on a centered image

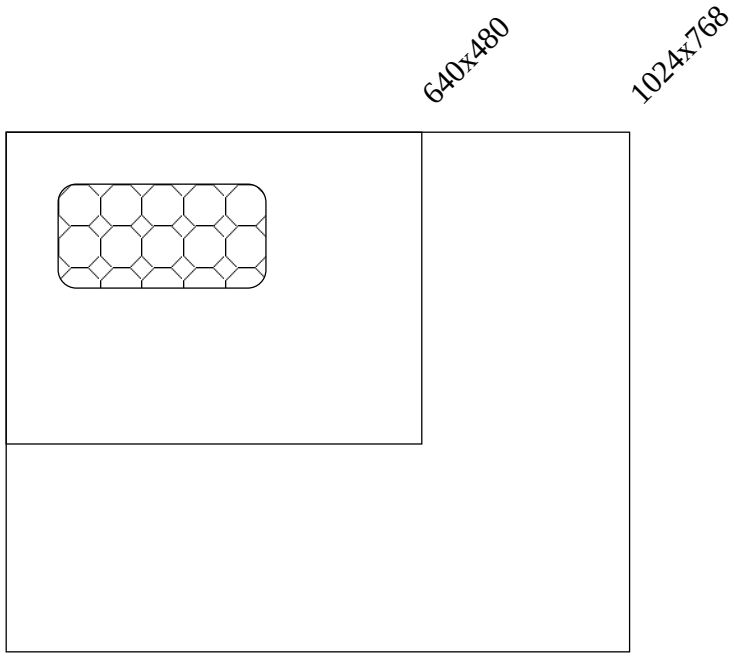

Figure 3: Good: resolution-independent positioning of an image

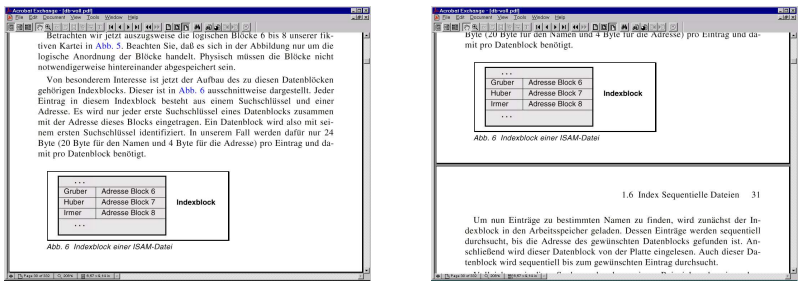

Figure 4: Link to bottom of page with View→Single Page (left) and View→Continuous (right)

Other tools: Adobe Distiller, ghostscript, Acrobat Reader, xpdf

3.11 ghostscript

Where to get: Ghostscript is freely available for non-commercial use. Support can be requested from the address below:

Aladdin Enterprises
P.O. box 60264
Palo Alto, CA 94306
Phone: +1-415-322-0103 (8:30 AM - noon, Pacific time)
Fax: +1-415-322-1734
EMail: ghost@aladdin.com

WWW: http://www.cs.wisc.edu/ ghost/
FTP: ftp://ftp.cs.wisc.edu/ghost/aladdin

Supported formats:
Input:

- PostScript Level 1 & 2
- Display PostScript

Output:

- Various pixel image formats (PCX, TIFF, PNG, PBM, ...)
- Various non-PostScript printer formats (PCL for HP diskjet and laserjet, Epson, ...)
- display output for Windows, Macintosh and X11
- Portable Document Format V1.2

Pros and Cons: Through its complete implementation of all PostScript commands and also the availability of lots of output formats, ghostscript is not only used to display PostScript, but also by many programs working with PostScript in one way or another, e.g. it's used by frame2html and latex2html to extract figures and symbols unknown to HTML.

Together with a decent frontend (such as ghostview or gv) it's also a very handy tool to preview PostScript documents on screen.

When using ghostscript to produce PDF output, the output differs from Adobe's Distiller output in that it is not compressed (no image downsampling is performed) and links are displayed different. It's also one of the know problems that ps2pdf sometimes produces output with missing links.

Tips & Tricks: -

Fixes: None needed.

Other tools: Adobe Distiller, TEX, pstools, as well as every word processor capable of producing PostScript output

3.12 HotDog Pro

HotDog is a time-limited shareware HTML editor for Windows 95, which can be upgraded to full version by receiving a license key.

Where to get:

- WWW: http://www.sausage.com/store/indexST.htm
- WWW: http://tucows.rz.fh-hannover.de/files/hdallinst.exe

Supported formats:
Input:

- HTML
- ASCII

Output:

- HTML

Pros and Cons: The HTML editor provides operations on several files, which can be quite handy when one has to correct some mistake a converter has introduced into several files at once. Utilities for project management include checking for broken links ("linkcheck") and also HTML syntax checking ("htmlcheck"). HTML syntax hilighting is done when editing HTML documents.

A drawback when working a lot with this editor is that it is fully GUI driven, and does *not* provide any key-shortcuts.

Tips & Tricks: -

Fixes: -

Other tools: FrontPage, every text editor and converter producing HTML.

3.13 FrontPage

FrontPage is an integrated suite of web-server, HTML-editor and site-administration toolkit with support for virtual web servers and access-administration for several users, enabling different permissions for different sub-trees of a web site.

Where to get: Frontpage is only available for Windows-based PCs.

Microsoft Corp.
x Redmont
WWW: http://microsoft.com/ (if available by accident)

Supported formats
Input:

- HTML
- ASCII
- RTF
- Word for Windows

Output:

- HTML with MicroSoft extensions

Pros and Cons: FrontPage is an toolkit for web site generation. The graphical overview of complex document structures (texts, images, ...) helps as much as the facility to search for any broken links. Moving of whole directories is supported, with links being corrected automatically then. The integrated web server speeds development of CGI scripts as well as the frame-designer does for frames.

Disadvantages are that the resulting documents use many Micosoft extensions to HTML, reducing the usability of these pages on a wider range of platforms. Created documents often contain lots of meta-information not useful for the user, such as information on image-maps.

See [10] for a conversion report that utilized MS FrontPage.

Tips & Tricks: -

Fixes: None possible.

Other tools: Word for Windows, Internet Information Server

Supported formats:

Pros and Cons:

Tips & Tricks:

Fixes:

Other tools:

4 Guidelines for Document Conversion

Having a set of tools available is like an instrument to a musician, but there's more than just knowing how to play this instrument. Without the knowledge on how to play to the notes, disharmonies will come out, and the result will look (sound) bad[2]. And just as a musician has more than one way to interpret a certain piece of music, the converter of a document has to decide by which stylistic means he arranges the given fragments.

The following list of keypoints is intended to ease the process of identifying points which need attention, and to aid in deciding on how to solve them.

4.1 Target Format

The choice of the target document format, into which to convert a given electronic document depends mainly on the contents of this document. Given the impact of the usage pattern of a document, HTML is usually preferrable, but there are reasons which speak for the PDF format in some cases.

With HTML, some things are easy to do, some are difficult to do, and others aren't possible at all, given the current state of the available browsers commonly used, and the definitions on HTML they implement. E. g. documents with lots of formulas end up in including lots of small images into the HTML text, as there's currently no HTML reader in widespread use, that does support formulaes and mathematical symbols. In addition, available converters as latex2html do not work properly in translating all those formulas into images, so PDF is the way to go here.

Other features such as comments and notes on the margin that the inventors of HTML didn't have in mind but that are common on printed media can be realized in HTML with some effort with tables, frames, and other advanced

[2]Of course it's also possible the composer caused some disharmonies - during document conversion, this usually means more work for the converter. But that's another story.

features. If not, one should investigate if switching over to PDF can solve the problem.

Besides visible aspects, the the way in which the document will be used has some impact on what target format to use. While PDF documents are usually downloaded in one piece, documents in the HTML format are downloaded page by page. So if a document is expected to be read from front to back, PDF might be the choice. If it's more likely that only few parts of a document are accessed frequently (like e.g. table of contents and index) and the text is accessed on a on-demand basis, choosing HTML will support this usage pattern in a less network saturating way.

4.2 Language

Sometimes a book exists in more than one language, probably the author's native language and english. In such a case, the decision has to be made, which version of the document to convert for further use.

In some cases the answer is given by the fact which version of the document is more up to date. In any other case, the language should be used that the intended user community is most likely to accept preferrably. However, if the audience is international, english is the language of choice.

4.3 Fonts

When reading text on the screen, the choice of the font used is important. If it's too small, people may have to wear glasses to read it. Furthermore, with a small font, a lot of information can be put on a single page, which can reduce readability, too. On the other hand, choosing a font too big may need the user to scroll too much, which will be a burden when reading long texts, too. Therefore, one should be careful to use any extreme, and not use a certain size just because it fits the converter or it's default.

The main goal to maintain and remember is readbility, taking the most common settings into consideration that the user of a document viewer can tune.

Besides giving general rules, some document formats need special attention for fonts.

When converting TEX-documents into PDF (via DVI and PostScript), one should try to replace the pixel-fonts TEX uses into PostScript fonts, as these are clearer to read and faster to display on screen (see discussion on TEX above on how to achive this). Furthermore, when printing, the document is more resolution independent of the output device.

When converting into PDF, one should also choose a font size that would be ok for printing, too. Just because a PDF reader provides the facility to zoom into every word, the page might still get too crowded with a small font.

When converting into HTML, this problem isn't really there, since the user of the document can set the font size via his browser, so he can adjust it so it fits his reading abilities. However, even here there are two points that should be

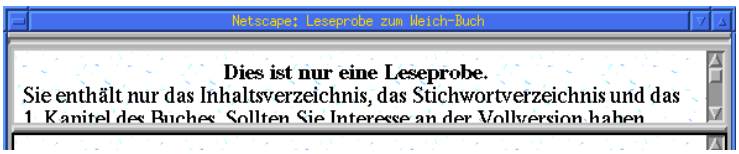

Figure 5: Example of a bad designed (font-insensitive) Frame. Even worse ones don't even provide a scroll bar.

Figure 6: Table and horizontal rule on white and grey background.

remembered. When using images with text in them, e.g. as in many technical drawings etc., the size of that font can not be chanced any more, as it's part of the images. Care must be taken so that this font is displayed ok, too. Second, when using frames, these are usually specified in fixed relations relative to the window size. When very small frames are used, e.g. for a navigation bar or page header/footer, it's possible text (or even worse, links) get scrolled outside of the visible area of the frame when the user has chosen a very big font. It's usually better to use some kind of tables for these purposes as they are font sensitive, in contrast to frames.

4.4 Colors

When using color, a few aspects must be take into account as well, else the result may not be easy to read or even nice to look at.

When going to PDF, using color is usually no matter, as the underlying color models from PostScript ensure that one color displays the same everywhere, besides that, PDF documents are usually in a uniform "black on white" style.

This might not be the case with documents in HTML. The "black on grey" that many browsers offer today is often deemed as boring, and that at least a different background color has to be chosen for the pages. White is easily chosen and also pretty to look at, but one should use it with care if the document uses tables with borders (<TABLE BORDER>), horizontal rules (<HR>) and other objects which have a three-dimensional look by default, as the white background will harm this look, see Figures 6. In such a case, shades of grey and pastel may provider a better look and also keep the 3D look of the said objects.

Figure 7: Characters with and without anti-aliasing

4.5 Use of Images

When images are part of a document, effort must be taken so the result looks good in the converted document. Often, more than one step is necessary to bring an image from a source to the target document, usually GIF and/or JPEG when heading for HTML. A good set of image manipulation tools will help in making this step easier. But still, one has to know a little bit on what's important when generating images.

First and most important is, that every detail on the picture is visible. Ways to archive this are switching on anti-aliasing if the tools and image formats support it. Anti-aliasing has a positive effect not only on fonts and their readability, but also lines, circles and other graphical objects look less crude, whether they are color or black and white. See Figure 7 for an example.

When a image contains text, the notes on fontsize explained in the previous section apply.

Furthermore, when an image consists of line drawing, the width of these lines shouldn't be taken too narrow or to wide, to keep details and not stroke them out by too thick lines. This is usually no problem when going for PDF, as this format contains good mechanisms for sizing images itself, as well as anti-aliasing is usually done automatically by the document viewer.

But when converting into a pixel format such as GIF and/or JPEG for use in an HTML document, keeping details may be a problem because to have all the details visible on the picture, the dimension of the image may exceed the size of what can be viewed even on a 640x480 display. To solve this problem, it's easiest to put a small (thumbnail) image on the page itself, and then create a link to the image with a higher resolution. That way, everyone can watch the details in his own demand, and no time is wasted to download some big image

the user doesn't want to watch.

The same kind of time-saving can be archived (in addition) by reducing the number of colors an image uses. It's of little use to insert any true- and highcolor-images with millions of colors, when most of the users have displays which hardly support 256 different colors, and even these colors are shared by several images, documents and/or applications.

Beside all those visible aspects of the contents of an image, there are also some guidelines to regard on inserting them into documents.

Most important, every image should have some text outlining what it's good for, either directly above of below it. (Of course it should be only above or below throughout the whole book, not mixed!) When converting into HTML, one might also set an additional ALT-parameter to the IMG-tag (<IMG ALT="some-text" ...>) to give a textual description of the picture in case the browser used to read the document has images disabled or doesn't support them at all.

4.6 Document Segmentation

Document segmentation means dividing a large document into smaller pieces. This technique is used to reduce download time, so only the parts of a documents need to be downloaded that are actually of interest, not the whole document. Furthermore, especially with documents in HTML, having a document divided in small pages prevents that the user has to scroll a lot, and it's easier to bookmark a certain location than if that interesting place is buried somewhere in a long document.

Document segmentation in HTML is usually done one page for each (sub)section, resulting in many small pages that can be loaded quickly. Also, only the images referenced in that page are downloaded, not every one in the whole document. In sections only consisting of the section heading and no text, it's recommended to merge this heading with the next section (and it's text), so that no whole page is used for only a heading. Figure 8 illustrates this merging process.

With PDF, the story is a bit different, as the linking between different files isn't very well handled by the existing processing tools, and therefore rather big segments are recommended, usually containing a whole part or chapter of the document. If an index or working table of contents is essential for a PDF document, one might even considering not segmenting the document at all, to keep the link information.

4.7 Design of the Navigation Bar

With having the big document broken up into many little pages, it's necessary to provide a common method to skip through all these pages, going back and forth. Also, when reading some page, it might be useful to be able to go to the index and/or table of contents as well. The mechanism providing this is called a "navigation bar". In PDF, this is built into the document format and the reader itself, so nothing has to be done there, but in HTML this has to be

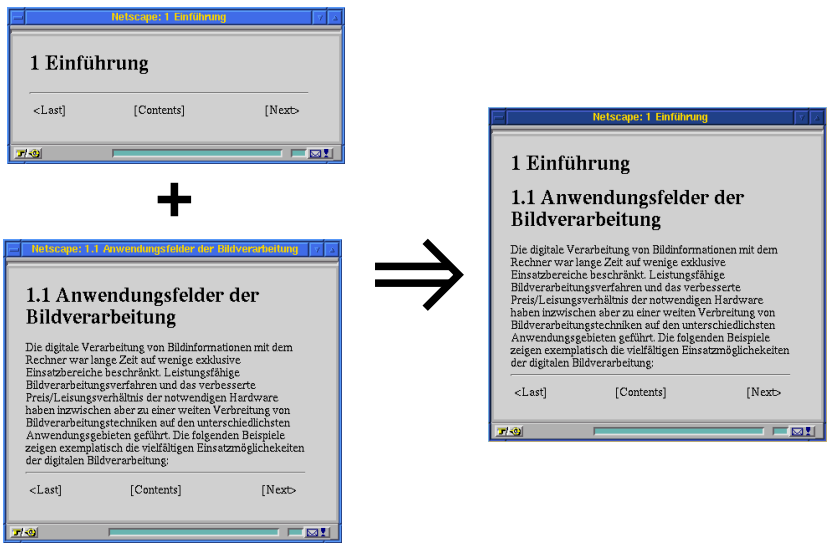

Figure 8: Example of merging two pages after segmentation

realized by hand. The navigation bar provides a doubly linked list of all the pages, for going to the next and previous page, see Figure 9.

The buttons a navigation bar should provide at the very least links to the next and last page. The "back"-link from the table of contents and the "next"-link from the last page are usually omitted. Additional links to the table of contents and index make it much more usable.

Also, a "up" link may provide a way to go up one step in the document hierarchy, as is shown by Figure 10.

The "up"-links only go *one* level up to the next hightest page, as for example the "up"-link for 1.2.1 goes to 1.2.

Aside from the logical structure of the links, the actual look of these links in the navigation bar may vary between documents (although consistency is a must within a single document!). The links may consist of pictures, chapter numbers, chapter names, or both, see Figures 11, 12, 13 and 14 for examples.

When placing navigations bars, attention should be paid that there is one at the beginning of the page, and also a second one at the end, so the reader of the page doesn't have to scroll up when he's done reading the page, but instead can go to the next page directly.

4.8 Links

Links are the essence of every hypertext document. Besides referencing related information, it can be (and is!) used to provide navigation inside a document in HTML. Therefore, it's important to make sure that every link is working properly, to ensure correct navigation, especially in the navigation bar, but also

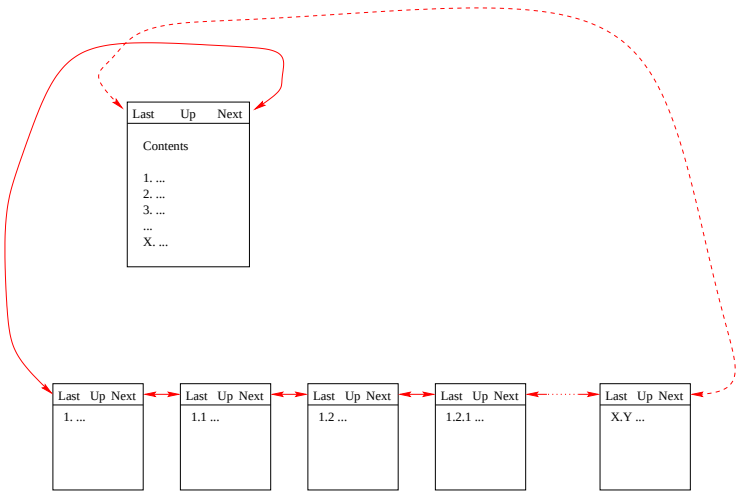

Figure 9: Connecting the document segments with a navigation bar.

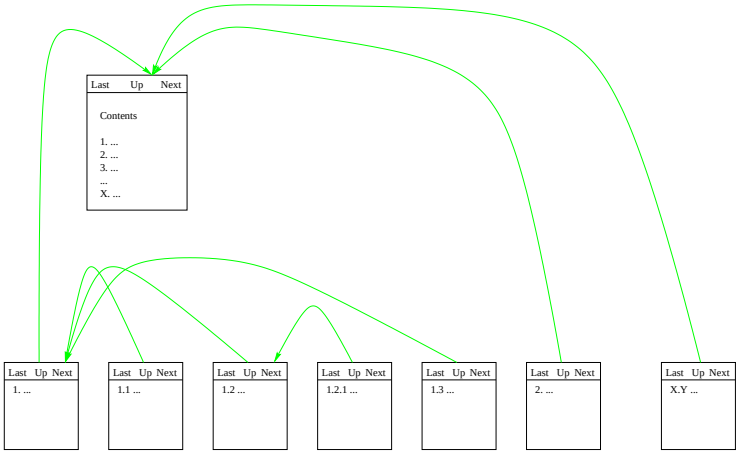

Figure 10: Navigating in the document hierarchy with a “up” link

Figure 11: Navigation bar with images.

<BACK] [CONTENTS] [INDEX] [FORWARD>

Figure 12: Navigation bar with plain text.

Figure 13: Navigation bar with chapter names

Figure 14: Navigation bar with chapter numbers and names.

in the table of contents and the index. While HTML-browsers display in detail that a link is broken when followed, PDF readers will act as if the link wasn't there if the linked destination is not available.

Also to provide smooth navigation, the navigation bar should be visible when following a link to a new page. This is not the case when converting LaTeX documents to HTML with latex2html, as the links produced by that tool rather address the first heading on a page, resulting in the navigation bar being scrolled out. When the user of the document decides he wants to go to the next page immediately, he has to use the scoll bar first, which is bad. Rather, these links of the form "xxx#yyy" need to be converted into simple links of the form "xxx", so the navigation bar's also visible.

As far as possible and supported by source format and converter tools, cross references inside a document should be converted into links as well as any available index.

For links of the form "see chapter XXX", "see section XXX", "see figure XXX", the chapter-, section- or figure-number "XXX" can be the anchor of the link, pointing at the referenced object. For references naming a page ("see page XXX"), it is hardly possible to name a page numer at all, and even in those cases it's unlikely that the named location has anything to do with a physical page. Therefore, it's ok to use some sort of button as an anchor for the link, instead of the page number, as shown in Figure 15.

When using the icon provided by latex2html and choosing a non-standard background color, the icon must be fixed so only the edges are transparent, not

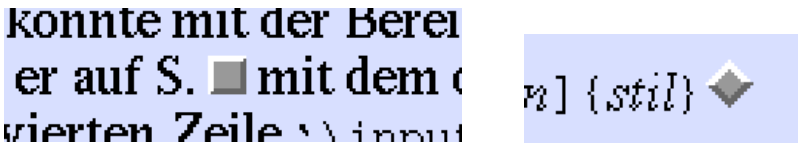

Figure 15: Page references with images as anchor

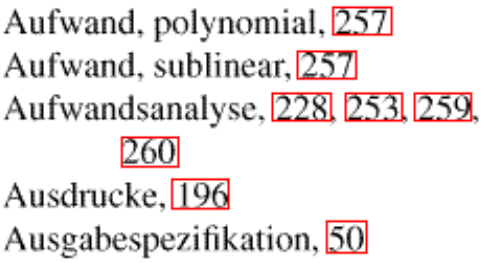

Figure 16: Virtual page numers as anchors for links in index.

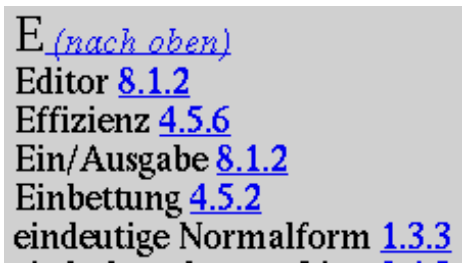

Figure 17: Section numbers as anchors for links in index.

the center.

In the index, there's a greater variety on how to display a anchor. Possible solutions are a virtual page-number that's only there to provide an anchor, an image just as described above, or listing the referenced section's number and/or heading. Figures 16, 17, 18, 19 and 20 show some examples.

When the `\ref` and `\pageref` are used in a LaTeX document that is then converted to HTML using latex2html, a small square is placed as anchor, instead of any page/section number. To turn this into plain text, information written into the .aux file can be used to gain these page/section number. An example script for this can be found in the conversion report from Graf/Eibisch [5].

4.9 Table of Contents

To get an overview of a documents contents, a table of contents should be available, from which every section can be accessed directly. This can be done with PDF and HTML, when converting LaTeXto HTML, the hyperref macros can do this job automatically.

Sometimes the table of contents gets quite long. When converting into HTML, it's recommended to divide the table of contents into two levels. The main table on the first level only displays chapter headings, linking to a table of contents only for this chapter on the second level. Figure 21 illustrates this splitting of the table of contents.

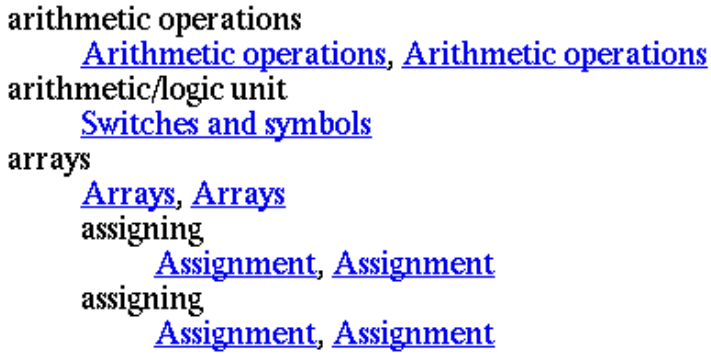

Figure 18: Section names as anchors for links in index.

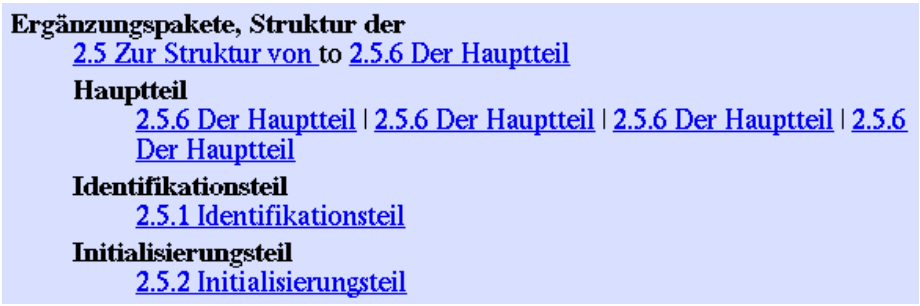

Figure 19: Section numbers and names as anchors for links in index.

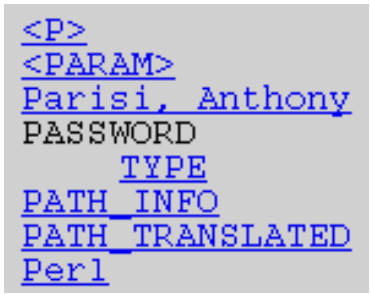

Figure 20: Keywords as anchors for links in index.

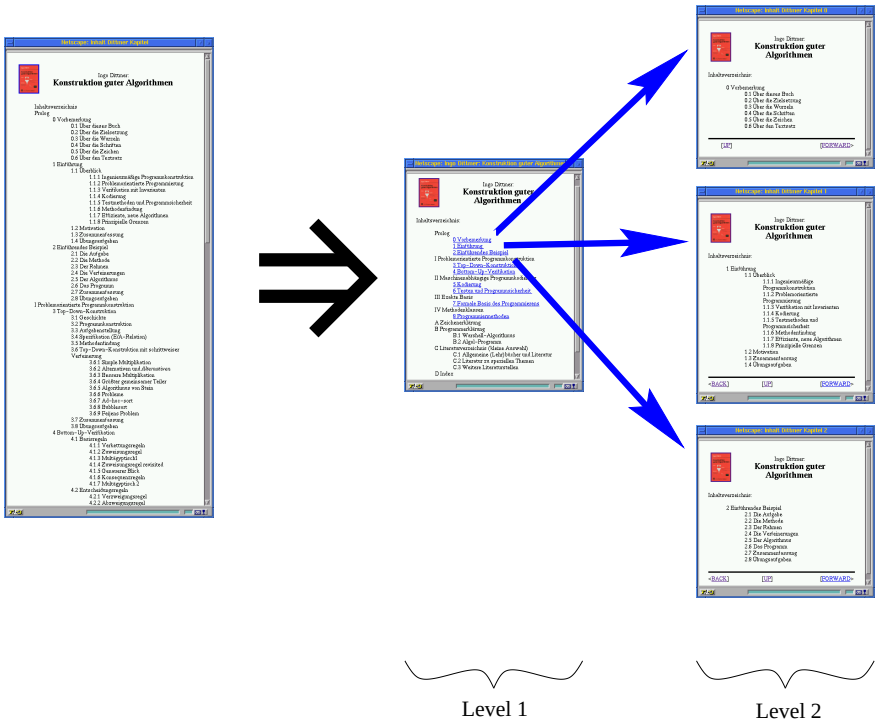

Figure 21: How to split the table of contents into two levels.

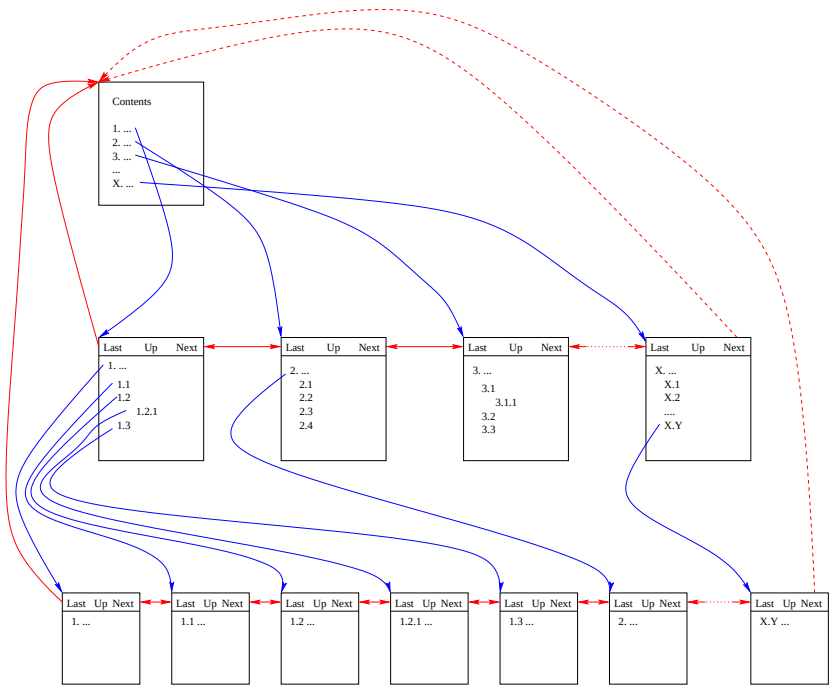

Figure 22: Impact of a second level of table of contents on navigation.

As for navigating, the chapter's tables of contents on the second level should be linked into a circular list (with references to the next/last chapter), just as regular pages are, see Figure 22.

As with the regular pages, a "back"-link from the first page and a "next"-link from the last page may be omitted. However, "up"-links should be available, pointing to the first level table of contents.

4.10 References

Documents may contain collections of references to books, papers, magazines and other type of information, that is references throughout the document. In the referenced section, it's possible to create links to other available documents, if they are available, their URL is know, *and* known to persist. Also, email-references (mailto:-links) may be inserted to contact an author of a book.

Furthermore, so-called "backlinks" can be inserted from the reference of a book back into the places in the document, where this book (or whatever) was references. This makes it easier to find out what for a certain book was being references for. When using the hyperref-macros for a LaTeX-to-PDF conversion, this can be done automatically by giving the "backref"-option when calling the hyperref macros:

1 2 3 4 7 A B C D E F G H I J K L M N O P Q R S T U V W Z

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Figure 23: Various character-jump-bars for an index

```
\RequirePackage[backref]{hyperref}
```

4.11 Index

The last part of a document is usually the index, containing keywords and references into the document, where these keywords first appear or are described. Many source formats and converters support index creation, either when converting to PDF or HTML.

When going for PDF from a LaTeXsource and using the hyperref macro package, a hyperlinked index can be generated by giving the "hyperindex" option:

```
\RequirePackage[hyperindex]{hyperref}
```

When going from FrameMaker to HTML using frame2html, this converter needs to be fixed to recognize index links at all, and then the index needs to be generated with the help of an external script. See the conversion report for Jörg Raasch's book for further details.[6]

As noted above during discussion of the look of links in the previous section, there are a number of ways to display links in the index, ranging from (virtual) page numbers, to images and chapter number/names as anchor, see Figures 16, 17, 18, 19 and 20.

Besides the look of the links (or better, their anchors) in the index, there's another point to pay attention to when generating an index, especially in HTML. When the index gets too long, it becomes slow to load and awkward to use. In such a case, at least a character-jump-bar has to be inserted, which allows jumping to the beginning of every set of index marks that begin with that character. The look of this bar may differ, but if present, it needs to be on the top of the index as well as at the end, similar to the navigation bar. Figure 23 shows some examples.

Of course, inserting a character-jump-bar just helps navigating inside the index, it doesn't make it shorter or speed up loading. If required, the index can be broken into a single file for every set of index words starting with the same letter, with a jump-table for every file as the starting point, similar to the table of contents with two levels, see 4.9

4.12 Titlepage

For every document, either converted to PDF or HTML, there needs to be a starting page in HTML. The page should contain a picture of the book's cover, and the text of the book's back. This titlepage should be designed after the guidelines of the publisher of the book, so all the books of one publisher look the same. If the conversion is into PDF, there should also be a table of contents in HTML besides the titlepage, so anyone interrested in the book doesn't have to download the whole document for having a first look on that's in there.

5 Communication and Exchange of Experiences

So far, a lot of topics have been addressed that are important when converting a document into either PDF or HTML. However, if one starts working on a conversion, it's easy to loose the overview. Therefore, it's good to have some documents available at which too look for reference, and to see how the one or other detail has been realized. If other people at the same site are working on converting documents, it can be very useful to not let them work independent of each other, but to give them a base for communications, either through a (local) mailing list or some meetings scheduled to inform everyone about the state, problems and solutions encountered.

If unsolveable problems arise, there's a wealth of information on document conversion collected in the conversion reports during the MeDoc project and hosted by Springer, see [8].

On the same site, there's also a list of email addresses of people who've worked on the various document conversions during the MeDoc project, sorted by source and target format they've converted from and to. So this may give another way of getting help when someone doesn't know how to handle a certain problem.

6 Workflow of a Conversion

After explaining the formats of electronic documents, how to convert them and by using which guidelines, by explaining what's important and where traps and pitfalls are, this section gives an overview of the actual workflow for a document conversion. Ranging from the aquisition of the document sources, ranging over to pinning the desired goal down and what two principal ways exist to realize the end product if the converter doesn't do its job 100%, until the final delivery of the produced document.

6.1 Obtaining Sources

Before you can even think about converting anything, you need to get the sources to convert. They are usually obtained after consulting with the publisher or directly from the author of the book in question. Regardless if the source comes

from the author or the publisher, it's good to have both addresses for later feedback.

Ways to get the source include traditional media such as floppy disks and tapes, or electronic ways such as email or FTP. In the first case, it's very important that the media is in a format that can be read at the converters site. E. g. if a disk is shipped in Macintosh or Amiga file format, it's highly unlikely anyone can read it. The least common nominator for this job is probably still the MS-DOS filesystem, with files compressed in some way to retain the filenames, their case and the directory structure. The same goes with tapes, where one should agree on block size and backup software used before attempting to ship anything.

Problems can arise when using an electronic transport way, too, though. When sending through FTP or email, the files should also be compressed for the reasons given above. If the network in use is too slow, has a high rate of packet loss or connections are terminated often and on a random basis, the archive files containing the source should be chopped into little pieces of 1 megabyte, 100 or even 10 kilobytes, and each of these little pieces can then be transported seperately.

When getting the sources for the Weich-book, this method had to be applied, as the author of the sources lived in Austria, and the network between austrian schools and the german education network was then routed via the EBone network, going from Regensburg to Vienna via Duesseldorf, Amsterdam and Paris, and it was impossible to transfer the 15 megabytes file via FTP in one piece. Several attempts have been taken during all day- and night times, but no chance. The transfer always terminated after some random time. The file was then cut into tiny pieces and send to Regensburg via email, using the automatic retransmit-on-failure mechanisms employed in the mail protocol (SMTP). With this method, the little packets arrived one by one, slowly, though, but surely, until all pieces were there and could be glued together to a big archive again, which then could be processed. With today's Ten34 network connection european educational networks, this problem wouldn't arise anymore, but still it was a problem at the end of 1996!

After getting the source for a document to convert, it's important to check if it's complete, i.e. containing the files for all chapters, index, in case of TEX documents any additional font, class and style files as well as any bibliographic data and information on the order in which to process all the files, if necessary. Any omissions should be reported back to the supplier of the sources immediately, so these can be received as fast as possible. Also, if the source format is unknown this kind of information should (really! :-) be obtained, together with which tools may be helpful, e.g. with which software images were created, which platform the author has used, any special configuration, etc.

What's left to get is a printed version of the document. This can be received usually either from the local library, or if unavailable, often the publisher will supply an issue. The printed version is important before and after the conversion: before, so one can see what's coming up without going through all the source files, and having to figure out a probably non-trivial building process.

During and after the conversion, the paper version of the document is most helpful to compare if the converted document is similar enough to the printed version, and if every detail is realized in a satisfactory way.

6.2 Specifying the Goal

The next step after obtaining the sources is to make some decisions. The first and most important is into which format to convert the document in question, see 4.1 for a discussion.

After the target format is nailed down, other decisions have to be made on which language to use (if there are several), what colors to use, how to divide the chapters into small pieces (segmentation), design of the navigation bar, table of contents, index, as well as the display of cross references within the document. Guidelines and solutions for all these questions have been given in the previous chapter and won't be repeated here.

6.3 Intercepting the Converter

Doing the conversion is no problem if the conversion tools do what they should. But this is rarely the case. Often, the tools produce incorrect output, that can be fixed in one of two ways:

- early interception, i.e. tuning the converter
- late interception, i.e. fixing the output manually

Early interception in the conversion process means to recognize any errors produced by a converter, and to fix the converter to no longer produce this error. This is usually an iterating process, running the converter, fixing, running it again, ... until all bugs have been identified and fixed. There's still a number of fixed to do by hand afterwards, which are either not possible to fix in the converter or source at all, or which would cost too much time and knowledge to fix, both of which may not be available. When fixing the converter, this requires having the source code, which is usually no problem when using freely available products. When using commercial converters, this is a problem, though. Bug reports can be sent to the vendor, but usually the time until a fix comes out is way to long to wait for it. Nevertheless fixing the converter means one only has to do a change in one place, and not many places all over the document (possibly leaving something out by mistake). Another advantage of fixing the converter is that the made fix can be contributed to everyone having the same problem, resulting in the enhancement and support of freely available software.

In case the source for a converter is not available or one doesn't have the knowledge to modify it, the resulting output of the conversion step has to be post processed, either by hand or by a (set of) scripts which fix the mistakes the converter introduced. Often, these fixes can be applied by using a (multi-file) search-and-replace-operation, but sometimes the fix is not so easy. Some scripting language for processing the HTML file is needed then, usually perl,

as it's available on all major conversion platforms. With such a language, it's e.g. possible to find out the first heading (number and/or name) and to write it into the title of a HTML document, to find out which page or chapter number a given symbolic reference in a LaTeX of FrameMaker document corresponds to, etc. See [6] and [5] for some examples.

If it's not possible to employ such a language to fix a problem in all files, the output has to be fixed by hand. This is no problem if a mistake only occurs in a few places, but if a mistake occurs all over the document, fixing by hand is an endless waste of time that should be avoided if possible.

6.4 Feedback

Feedback by people not directly working on a document conversion may be helpful, as these people usually see problems not obvious to the converter himself. Also, the earlier there's feedback, the earlier any fundamental problems can be fixed, and the less damage is done by a wrong decision during the specification of the conversion goals.

Such helpful feedback may come from collegues either working on book conversions, too, or not, from any local supervisor or even from the project leader. The persons who have to be asked for in any case are the author of the book and the publisher, as it is their document that is being converted, and the work is being done for them.

As mentioned above, feedback is welcome at every point during document conversion, but when the conversion is done, the publisher and the author should have a look again to give their final ok before the document is delivered.

6.5 Document delivery

When everything is converted, the title page is done, etc. a final check should be done that all the files fit into one single directory which can be moved around without breaking anything (i.e. if there are no absolute links used by accident). If there's a reading rehearsal, it goes into it's own directory, just as the full version does. Any additional files (title page, table of contents, misc. images, ...) should be kept in one directory as well.

All the data can then be delivered via the same ways as the source of the document was received, either electronically or via traditional media, most probably in the form of compressed archive files when using electronical delivery. If there's a written conversion contract, it may specify this further.

If the publisher or project leader requires a conversion report, as was mandatory during the MeDoc project, this report should be included on the media for final delivery as well.

If a media such as CD-ROM is used for delivery, it should contain conversion report, title-page (and any auxiliary files), reading rehearsal and full version each in it's own directory, together with a README file describing which file/directory belongs to what, and (with directories full of HTML files) where the starting file is (if there is no index.html).

7 Conclusion

To sum up all the conversions, it can be said that the source texts obtained in the various formats can rarely be converted without doing some substantial manual work on them first; detailled guidelines for authors will help avoiding this. Also, guidelines for document converters as given in this document help a lot, as many experiences and ideas can be given without costing sweat and tears (again). Tools are available, but they don't always do what they promise. Knowing their strengths and flaws is essential for efficient work.

Given the experiences in document conversion performed during the MeDoc project, it can be hoped that future documents comply to then-given standards for authoring, as well as existing tools get improved to leave less work that has to be done manually.

References

[1] Brüggemann-Klein, A., "Präsentation auf dem MeDoc-Verleger-Treffen anläßlich der Frankfurter Buchmesse", TU München.

URL: http://medoc.informatik.tu-muenchen.de/deutsch/organisation/-projektbibliothek/folien/brueggem.ps

[2] Adobe Systems Inc., "PostScript language tutorial and cookbook" (the blue book), Addison-Wesley, Reading, MA, 1985.

PostScript language tutorial and cookbook is the most common tutorial book on PostScript. It provides an easy, informal introduction to the PostScript language and graphics primitives. The cookbook is a collection of programs that are offered as examples of PostScript usage. These samples have been chosen both as illustrations of the functional range of PostScript and as useful ingredients for inclusion in application packages.

[3] Adobe Systems Inc., "PostScript language reference manual" (the red book), 2nd ed, Addison-Wesley, Reading, MA, 1990.

PostScript language reference manual is the book that defines the PostScript language. The second edition not only defines Level 1 PostScript, but also encompasses the color, composite font, file system, and DPS extensions and the PostScript language Level 2.

[4] Adobe Systems Inc., "PostScript Language Program Design" (the green book), Addison-Wesley, Reading, MA, 1988.

PostScript Language Program Design is intended to teach the fundamentals of designing PostScript language programs and to show how the language works, so the your programs will be fast, well-behaved, easy to understand, and portable.

[5] Graf, S., Eibisch, C.: "Konvertierungsbericht zum Weich-Buch", FH Regensburg, FB Informatik, Prüfeningerstr. 58, 93049 Regensburg, Germany.

URL: http://medoc.springer.de/konvertierung/konvertierungsberichte/-weich_eibisch_kurz.html, http://medoc.springer.de/konvertierung/-konvertierungsberichte/weich_eibisch_lang.html

[6] Feyrer, H.: "Konvertierungsbericht zum Raasch-Buch", FH Regensburg, FB Informatik, Prüfeningerstr. 58, 93049 Regensburg, Germany.

URL: http://medoc.springer.de/konvertierung/konvertierungsberichte/-raasch_feyrer.html

[7] Mayerhofer, J.: "Konvertierungsbericht zum Kopka, 1. Band", FH Regensburg, FB Informatik, Prüfeningerstr. 58, 93049 Regensburg, Germany.

URL: To be placed somewhere below http://medoc.springer.de/-konvertierung/konvertierungsberichte/ when ready.

[8] Various: Collection of all conversion reports given for the book conversions during the MeDoc project.

URL: http://medoc.springer.de/konvertierungen/konvertierungsberichte/

[9] Knuth, Donald E., "The TEX Book", Addison-Wesley, Reading, MA, 1986.

[10] Klorz: "Konvertierung des Buches 'Theoretische Informatik' von Werner Brecht ins HTML-Format - Erfahrungsbericht", Lehrstuhl Informatik 5, RWTH Aachen.

URL: http://medoc.springer.de/konvertierung/konvertierungsberichte/-brecht_sklorz.html

[11] Microsoft Product Support Services: "Rich Text Format (RTF) Specification", Microsoft Corp., 1994.

URL: http://night.primate.wisc.edu:70/0/software/RTF/RTF-Spec-1.3.ps

[12] Tim Bienz, Richard Cohn, James R. Meehan: "Portable Document Format Reference Manual, Version 1.2", Adobe Systems Incorporated, 1996.

URL: http://www.adobe.com/supportservice/devrelations/PDFS/TN/-PDFSPEC.PDF