Hypertext marks in LaTeX

Sebastian Rahtz
Email: s.rahtz@elsevier.co.uk

processed May 6, 1998

Contents

1 The main macros

1 ⟨∗package⟩

1.1 Package options and setup

It *does* need the December 95 release of LaTeX, because it uses `\protected@write`, and it defines commands in options; and the page setup internal code changed at that point. It'll probably break with the later releases!

```
2 \RequirePackage{keyval}
3 \def\hyper@warn#1{\PackageWarningNoLine{hyperref}{#1}}
4 \def\hyper@info#1{\PackageInfo{hyperref}{#1}}
5 \def\pdf@bbox{pdf@llx pdf@lly pdf@urx pdf@ury}
6 \newdimen\@linkdim
7 \newif\ifhy@backref
8 \newif\ifhy@bookmarks
9 \newif\ifhy@bookmarksopen
10 \newif\ifhy@driverloaded
11 \newif\ifhy@psize
12 \newif\ifhy@colorlinks
13 \newif\ifhy@figures
14 \newif\ifhy@nesting
15 \newif\ifhy@hyperindex
16 \newif\ifhy@plainpages
17 \newif\ifhy@activeanchor
18 \newif\ifhy@raiselinks
19 \newif\ifhy@breaklinks
20 \newif\ifhy@pageanchor
21 \newif\ifhy@debug
22 \hy@bookmarkstrue
23 \hy@bookmarksopenfalse
24 \hy@raiselinksfalse
25 \hy@breaklinksfalse
26 \hy@figuresfalse
27 \hy@nestingfalse
28 \hy@backreffalse
29 \hy@plainpagestrue
30 \hy@hyperindextrue
31 \hy@pageanchortrue
32 \hy@driverloadedfalse
33 \hy@psizefalse
34 \@ifpackageloaded{xr}{%
```

If reading external aux files check whether they have a non zero fourth field in `\newlabel` and if so, add the URL as the fifth field.

```
\def\XR@addURL#1{\XR@@dURL#1{}{}{}{}\\}
\def\XR@@dURL#1#2#3#4#5\\{{#1}{#2}%
  \if!#4!\else
    {#3}{#4}{\XR@URL}%
\fi}
}{}
\def\hy@true{true}
\let\literalps@out\@gobble
\let\pdfbookmark\@gobbletwo
\let\Acrobatmenu\@gobble
\def\@writetorep#1#2#3{}%
\def\Hyp@boolkey#1#2{%
  \hyper@info{option #2 set #1}%
  \csname hy@#2\ifx\relax#1\relax true\else#1\fi\endcsname}
\define@key{Hyp}{draft}[true]{\AtBeginDocument{%
    \let\hyper@@anchor\@gobble
    \gdef\hyper@link##1##2##3{##3}%
    \let\hyper@anchorstart\@gobble
    \let\hyper@anchorend\@empty
    \let\hyper@linkstart\@gobbletwo
    \let\hyper@linkend\@empty
    \def\hyper@linkurl##1##2{##1}%
    \def\hyper@linkfile##1##2##3{##1}%
    \def\Acrobatmenu##1##2{##2}%
    \let\PDF@SetupDoc\@empty
    \hyper@warn{ draft mode on}%
   }%
}
\let\KV@Hyp@nolinks\KV@Hyp@draft
\define@key{Hyp}{a4paper}[true]{\def\special@paper{210mm,297mm}%
  \def\hy@pageheight{842}}
\define@key{Hyp}{a5paper}[true]{\def\special@paper{148mm,210mm}%
  \def\hy@pageheight{595}}
\define@key{Hyp}{b5paper}[true]{\def\special@paper{176mm,250mm}%
  \def\hy@pageheight{709}}
\define@key{Hyp}{letterpaper}[true]{\def\special@paper{8.5in,11in}%
  \def\hy@pageheight{792}}
\define@key{Hyp}{legalpaper}[true]{\def\special@paper{8.5in,14in}%
  \def\hy@pageheight{1008}}
\define@key{Hyp}{executivepaper}[true]{\def\special@paper{7.25in,10.5in}%
  \def\hy@pageheight{720}}
\define@key{Hyp}{debug}[true]{%
   \lowercase{\Hyp@boolkey{#1}}{debug}}
\define@key{Hyp}{extension}{\def\XR@ext{.#1}}
\def\XR@ext{.dvi}
```

If we are going to PDF via HyperTeX `\special` commands, the dvips (-Z option) processor does not know the *height* of a link, as it works solely on the position of the

closing `\special`. If we use this option, the `\special` is raised up by the right amount, to fool the dvi processor.

```
\define@key{Hyp}{raiselinks}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{raiselinks}}
```

Most PDF-creating drivers do not allow links to be broken

```
\define@key{Hyp}{breaklinks}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{breaklinks}}
```

Determines whether an automatic anchor is put on each page

```
\define@key{Hyp}{pageanchor}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{pageanchor}}
```

Are the page links done as plain arabic numbers, or do they follow the formatting of the package? The latter loses if you put in typesetting like `\textbf` or the like.

```
\define@key{Hyp}{plainpages}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{plainpages}}
```

Currently, `dvihps` doesn't allow anchors nested within targets, so this option tries to stop that happening. Other processors may be able to cope.

```
\define@key{Hyp}{nesting}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{nesting}}
```

1.2 Options for different drivers

```
\define@key{Hyp}{hyperref}[true]{}
\define@key{Hyp}{tex4ht}[true]{%
   \input{htex4ht.def}%
}
\define@key{Hyp}{pdftex}[true]{%
   \input{hpdftex.def}%
   \def\XR@ext{.pdf}%
   \PassOptionsToPackage{pdftex}{color}%
   \hy@breaklinkstrue
}
\define@key{Hyp}{dvipdf}[true]{%
   \input{hdvipdf.def}%
   \def\XR@ext{.pdf}%
}
\define@key{Hyp}{nativepdf}[true]{%
   \input{pdfmark.def}%
   \input{hdvips.def}%
   \def\XR@ext{.pdf}%
}
\define@key{Hyp}{pdfmark}[true]{%
   \input{pdfmark.def}%
   \input{hdvips.def}%
   \def\XR@ext{.pdf}%
   }
\define@key{Hyp}{dvips}[true]{%
   \input{pdfmark.def}%
```

```
   \input{hdvips.def}%
   \def\XR@ext{.pdf}%
   }
\define@key{Hyp}{hypertex}[true]{%
   \input{hypertex.def}%
   }
\define@key{Hyp}{dviwindo}[true]{%
    \input{hdviwind.def}%
    \setkeys{Hyp}{colorlinks}%
    \PassOptionsToPackage{dviwindo}{color}%
}
\define@key{Hyp}{dvipsone}[true]{%
   \def\XR@ext{.pdf}%
   \input{pdfmark.def}%
   \input{hdvipson.def}%
}
\define@key{Hyp}{textures}[true]{%
   \def\XR@ext{.pdf}%
   \input{pdfmark.def}%
   \input{htexture.def}%
}
\define@key{Hyp}{latex2html}[true]{%
   \AtBeginDocument{\@@hyperrlatextohtmlX}%
 }
```

Some fixes for broken ps2pdf (releases before 5.21) in Ghostscript. Daniel T. Cobra <cobra@gyron.acate.com.br> supplied these.

```
\define@key{Hyp}{ps2pdf}[true]{%
   \input{pdfmark.def}%
   \input{hdvips.def}%
   \def\@pdfborder{0 0 1}
   \define@key{PDF}{Color}{\pdf@addtoks{[##1]}{C}}
   \def\hyper@linkfile##1##2##3{%
    \bgroup
     \pdfmark[##1]{pdfmark=/ANN,Subtype=/Link,
      Border=\@pdfborder,linktype=file,Color=\@filebordercolor,
      Action=<< /S /GoToR /F (##2) /D \ifx\\##3\\[0 /FitB]\else/##3\fi >>}%
     \egroup
   }
   \def\XR@ext{.pdf}%
   \hy@driverloadedtrue
}
```

1.3 Options to add extra features

Make included figures (assuming they use the standard graphics package) be hypertext links. Off by default. Needs more work.

```
\define@key{Hyp}{hyperfigures}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{figures}}
```

Set up back-referencing to be hyper links, by page or section number,

```
\def\back@none{none}
\def\back@section{section}
\def\back@page{page}
\define@key{Hyp}{backref}[section]{%
 \ifx\\#1\\\def\@tempa{section}\else\def\@tempa{section}\fi
 \ifx\@tempa\back@section
  \PassOptionsToPackage{hyperref}{backref}
  \hy@backreftrue
 \else
   \ifx\back@opt\back@page
     \PassOptionsToPackage{hyperpageref}{backref}
     \hy@backreftrue
   \fi
 \fi
}
\define@key{Hyp}{pagebackref}[true]{%
  \PassOptionsToPackage{hyperpageref}{backref}
  \hy@backreftrue
}
```

Make index entries be links back to the relevant pages. By default this is turned on, but may be stopped.

```
\define@key{Hyp}{hyperindex}[true]{%
  \lowercase{\Hyp@boolkey{#1}}{hyperindex}}
```

1.4 Options to change appearance of links

Colouring links at the LaTeX level is useful for debugging, perhaps.

```
\define@key{Hyp}{colorlinks}[true]{\lowercase{\Hyp@boolkey{#1}}{colorlinks}}
\define@key{Hyp}{bookmarks}[true]{%
    \lowercase{\Hyp@boolkey{#1}}{bookmarks}}
\define@key{Hyp}{bookmarksopen}[true]{%
    \lowercase{\Hyp@boolkey{#1}}{bookmarksopen}}
\define@key{Hyp}{linkcolor}{\def\@linkcolor{#1}}
\define@key{Hyp}{anchorcolor}{\def\@anchorcolor{#1}}
\define@key{Hyp}{citecolor}{\def\@citecolor{#1}}
\define@key{Hyp}{urlcolor}{\def\@urlcolor{#1}}
\define@key{Hyp}{menucolor}{\def\@menucolor{#1}}
\define@key{Hyp}{filecolor}{\def\@filecolor{#1}}
\define@key{Hyp}{pagecolor}{\def\@pagecolor{#1}}
```

Default values:

```
\def\@linkcolor{red}
\def\@anchorcolor{black}
\def\@citecolor{green}
\def\@filecolor{cyan}
\def\@urlcolor{magenta}
\def\@menucolor{red}
\def\@pagecolor{red}
\define@key{Hyp}{baseurl}{\def\@baseurl{#1}}
```

```
\def\hyperbaseurl#1{\def\@baseurl{#1}}
\let\@baseurl\@empty
```

1.5 PDF-specific options

```
\define@key{Hyp}{linkbordercolor}{\def\@linkbordercolor{#1}}
\define@key{Hyp}{urlbordercolor}{\def\@urlbordercolor{#1}}
\define@key{Hyp}{menubordercolor}{\def\@menubordercolor{#1}}
\define@key{Hyp}{filebordercolor}{\def\@filebordercolor{#1}}
\define@key{Hyp}{citebordercolor}{\def\@citebordercolor{#1}}
\define@key{Hyp}{pagebordercolor}{\def\@pagebordercolor{#1}}
\define@key{Hyp}{pdfborder}{\def\@pdfborder{#1}}
\define@key{Hyp}{pdfpagemode}{\def\@pdfpagemode{/#1 }}
\define@key{Hyp}{pdftitle}{\def\@pdftitle{#1 }}
\define@key{Hyp}{pdfauthor}{\def\@pdfauthor{#1 }}
\define@key{Hyp}{pdfproducer}{\def\@pdfproducer{#1 }}
\define@key{Hyp}{pdfcreator}{\def\@pdfcreator{#1 }}
\define@key{Hyp}{pdfsubject}{\def\@pdfsubject{#1 }}
\define@key{Hyp}{pdfkeywords}{\def\@pdfkeywords{#1 }}
\define@key{Hyp}{pdfview}{\def\@pdfview{ #1 }}
\define@key{Hyp}{pdfstartpage}{\def\@pdfstartpage{#1}}
\define@key{Hyp}{pdfstartview}{\def\@pdfstartview{ /#1 }}
\define@key{Hyp}{pdfpagescrop}{\edef\@pdfpagescrop{#1}}
```

Default values:

```
\def\@linkbordercolor{1 0 0}
\def\@urlbordercolor{0 1 1}
\def\@menubordercolor{1 0 0}
\def\@filebordercolor{0 .5 .5}
\def\@citebordercolor{0 1 0}
\def\@pagebordercolor{1 1 0}
\def\@pdfpagemode{}
\def\@pdftitle{}
\def\@pdfauthor{}
\def\@pdfproducer{}
\def\@pdfcreator{LaTeX with hyperref package}
\def\@pdfsubject{}
\def\@pdfkeywords{}
\def\@pdfview{ FitBH }
\def\@pdfpagecrop{}
\def\@pdfpagescrop{}
\def\@pdfstartview{ /Fit }
\def\@pdfstartpage{1}
\let\PDF@SetupDoc\@empty
\def\special@paper{210mm,297mm}
\def\hy@pageheight{842}
\def\hypersetup{\setkeys{Hyp}}
```

Allow the user to use `\ExecuteOptions` in the cfg file even though this package does not use the normal option mechanism. Use `\hyper@normalise` as a scratch

macro, since it is going to be defined in a couple of lines anyway.

```
\let\hyper@normalise\ExecuteOptions
\let\ExecuteOptions\hypersetup
\InputIfFileExists{hyperref.cfg}{}{}
\let\ExecuteOptions\hyper@normalise
```

To add flexibility, we will not use the ordinary processing of package options, but put them through the *keyval* package. This section was written by David Carlisle.

```
\def\ProcessOptionsWithKV#1{%
  \let\@tempa\@empty
```

Add any global options that are known to KV to the start of the list being built in `\@tempa`.

```
  \@for\CurrentOption:=\@classoptionslist\do{%
     \@ifundefined{KV@#1@\CurrentOption}%
       {}%
       {\edef\@tempa{\@tempa,\CurrentOption,}}%
```

Now stick the package options at the end of the list and wrap in a call to `\setkeys`. Can simply use `\edef`, normally KV takes care to avoid expansion, but the package system has already fully expanded the package option list before passing it to the package, so no more harm can occur here.

```
     \edef\@tempa{%
        \noexpand\setkeys{#1}{\@tempa\@ptionlist{\@currname.\@currext}}}%
```

Do it.

```
    \@tempa
```

Just let the end of package cleanup know something happened.

```
  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
\ProcessOptionsWithKV{Hyp}
\ifhy@bookmarks
 \hyper@info{Bookmarks ON}
  \ifx\@pdfpagemode\@empty\def\@pdfpagemode{/UseOutlines }\fi
  \ifhy@bookmarksopen
    \hyper@info{Bookmarks open}%
    \def\@bookmarkopenstatus{}%
  \else
    \def\@bookmarkopenstatus{-}%
  \fi
\else
 \hyper@info{Bookmarks OFF}
 \AtEndOfPackage{\global\let\ReadBookmarks\relax
                 \global\let\WriteBookmarks\relax}%
 \ifx\@pdfpagemode\@empty\def\@pdfpagemode{/None}\fi
\fi
\ifhy@figures
 \hyper@info{Hyper figures ON}
\else
 \hyper@info{Hyper figures OFF}
\fi
\ifhy@nesting
 \hyper@info{Link nesting ON}
```

```
\else
 \hyper@info{Link nesting OFF}
\fi
\ifhy@hyperindex
 \hyper@info{Hyperindex ON}
\else
 \hyper@info{Hyperindex OFF}
\fi
\ifhy@plainpages
 \hyper@info{Plain pages ON}
\else
 \hyper@info{Plain pages OFF}
\fi
\ifhy@backref
 \hyper@info{Backreferencing ON}
\else
 \hyper@info{Backreferencing OFF}
\fi
\ifhy@colorlinks
   \AtEndOfPackage{\RequirePackage{color}}%
   \def\colorlink#1{\let\hyper@color\current@color\color{#1}}%
   \def\hyper@resetcolor{\let\current@color\hyper@color\set@color}%
   \hyper@info{Link coloring ON}
\else
   \let\colorlink\@gobble
   \let\hyper@resetcolor\@empty
   \hyper@info{Link coloring OFF}
\fi
\RequirePackage{nameref}
\AtEndOfPackage{%
 \ifhy@driverloaded
 \else
  \@ifundefined{pdfoutput}
   {\input{hypertex.def}}
   {\input{hpdftex.def}%
   \def\XR@ext{.pdf}%
   \PassOptionsToPackage{pdftex}{color}%
   \hy@breaklinkstrue}%
 \fi
}
\ifhy@backref
 \RequirePackage{backref}
\else
 \let\hy@backout\@gobble
\fi
\hy@activeanchorfalse
```

1.6 User hypertext macros

We need to normalise all user commands taking a URL argument; Within the argument the following special definitions apply: \#, \%, ~ produce #, %, ~ respectively. for consistency \~ produces ~ as well. At the *top level only* ie not within the argument of another command, you can use # and % unescaped, to produce themselves. even if, say, # is entered as # it will be converted to \# so it does not die if written to an aux file etc. \# will write as # locally while making \specials.

```
\begingroup
  \catcode‘\!\active
  \catcode‘\&\active
  \catcode‘\_\active
  \uccode‘\!=‘\%
  \uccode‘\&=‘\#
\uppercase{\endgroup
\def\hyper@normalise#1{%
  \begingroup
  \catcode‘\%\active\def!{\%}%
  \catcode‘\#\active\def&{\#}%
  \catcode‘\_\active\def_{\string_}%
  \let~\hyper@tilde
  \let\~\hyper@tilde
  \hyper@n@rmalise#1}}
\def\hyper@n@rmalise#1#2{%
  \edef\@tempa{\endgroup\noexpand#1{#2}}%
  \@tempa}
\providecommand\hyper@chars{%
  \let\#\hyper@hash
  \let\%\@percentchar}

\def\hyperlink#1#2{%
  \hyper@@link{}{#1}{#2}}

\def\href{\hyper@normalise\href@}
\def\href@#1{\expandafter\href@split#1\#\#\\}
\def\href@split#1\##2\##3\\{%
 \hyper@@link{#1}{#2}%
}
\RequirePackage{url}
\def\Hurl{\begingroup \Url}
\def\url{\hyper@normalise\url@}
\def\url@#1{\hyper@linkurl{\Hurl{#1}}{#1}}

\def\hyperimage{\hyper@normalise\hyper@image}
\providecommand\hyper@image[2]{#2}

\def\hypertarget#1#2{%
  \ifhy@nesting
    \hyper@@anchor{#1}{#2}%
  \else
    \hyper@@anchor{#1}{\relax}#2%
  \fi}
```

`\hyperref` is more complicated, as it includes the concept of a category of link, used to make the name. This is not really used in this package. `\hyperdef` sets up an anchor in the same way. They each have three parameters of category, linkname, and marked text, and `\hyperrref` also has a first parameter of URL. If there is an optional first parameter to `\hyperdef`, it is the name of a LaTeX label which can be used in a short form of `\hyperref` later, to avoid remembering the name and category.

```
\def\hyperref{\@ifnextchar[\label@hyperref\@hyperref}
\def\@hyperref{\hyper@normalise\@@hyperref}
\def\@@hyperref#1#2#3{%
  \edef\ref@one{\ifx\\#2\\\else#2.\fi#3}%
  \expandafter\tryhyper@link\ref@one\\{#1}%
}
\def\tryhyper@link#1\\#2{%
 \hyper@@link{#2}{#1}%
}
```

```
\def\hyperdef{\@ifnextchar[{\label@hyperdef}{\@hyperdef}}
\def\@hyperdef#1#2#3{%, category, name, text
  \ifhy@nesting
    \hyper@@anchor{#1.#2}{#3}%
  \else
    \hyper@@anchor{#1.#2}{\relax}#3%
  \fi}
```

We also have a need to give a LaTeX *label* to a hyper reference, to ease the pain of referring to it later.

```
\def\label@hyperref[#1]{%
  \expandafter\label@@hyperref\csname r@#1\endcsname{#1}}%
\def\label@@hyperref#1#2#3{%
  \ifx#1\relax
   \protect\G@refundefinedtrue
    \@latex@warning{Hyper reference `#2' on page \thepage \space
              undefined}%
   \hyper@@link{}{??}{#3}%
  \else
   \hyper@@link{}{#1}{#3}%
  \fi
}
\def\label@hyperdef[#1]#2#3#4{% label name, category, name,
                                % anchor text
  \@bsphack
  \protected@write\@auxout{}%
         {\string\newlabel{#1}{{}{}{}\##2.#3}}%
  \@esphack
  \ifhy@nesting
   \hyper@@anchor{#2.#3}{#4}%
  \else
   \hyper@@anchor{#2.#3}{\relax}#4%
  \fi
}
```

1.7 Underlying basic hypertext macros

Links have an optional type, a filename (possibly a URL), an internal name, and some marked text. If the second parameter is empty, its an internal link, otherwise we need to open another file or a URL. A link start has a type, and a URL.

```
\def\hyper@@link{\let\reserved@a\relax
  \@ifnextchar[{\hyper@link@}{\hyper@link@[link]}}
\def\hyper@link@[#1]#2#3#4{%
  \edef\@tempa{#2}%
  \ifx\@tempa\@empty
      \hyper@link{#1}{#3}{#4}%
  \else
      \expandafter\hyper@readexternallink#2\\{#1}{#3}{#4}%
  \fi
}
```

The problem here is that the first (URL) parameter may be a local `file:` reference (in which case some browsers treat it differently) or a genuine URL, in which case we'll have to activate a real Web browser. Note that a simple name is also a URL, as that is interpreted as a relative file name. We have to worry about # signs in a local file as well.

```
\def\hyper@readexternallink#1\\#2#3#4{%
```

Parameters are:

1. The URL or file name
2. The type
3. The internal name
4. The link string

We need to get the 1st parameter properly expanded, so we delimit the arguments rather than passing it inside a group.

```
 \expandafter\@hyper@readexternallink{#2}{#3}{#4}#1::\\{#1}%
}
```

Now (potentially), we are passed: 1) The link type 2) The internal name, 3) the link string, 4) the URL type (http, mailto, file etc), 5) the URL details 6) anything after a real : in the URL 7) the whole URL again

```
\def\@hyper@readexternallink#1#2#3#4:#5:#6\\#7{%
```

If there are no colons at all (`#6` is blank), its a local file; if the URL type (`#4`) is blank, its probably a Mac filename, so treat it like a `file:` URL. The only flaw is if its a relative Mac path, with several colon-separated elements — then we lose. Such names must be prefixed with an explicit `dvi:`

```
\ifx\\#6\\%
  \expandafter\@hyper@linkfile file:#7..\\{#3}{#2}%
\else
  \ifx\\#4\\%
   \expandafter\@hyper@linkfile file:#7..\\{#3}{#2}%
  \else
```

If the URL type is ‘file’, pass it for local opening

```
  \def\@pdftempa{#4}\def\@pdftempb{file}%
  \ifx\@pdftempa\@pdftempb
     \expandafter\@hyper@linkfile#7..\\{#3}{#2}%
  \else
```

otherwise its a URL

```
      \hyper@linkurl{#3}{#7\ifx\\#2\\\else\##2\fi}%
  \fi
 \fi
\fi
}
\def \@hyper@linkfile file:#1.#2.#3\\#4#5{%
    %file url, extension,xxx,link string, name
 \ifx\\#2\\\edef\this@ext{\XR@ext}\else\def\this@ext{.#2}\fi
 \ifx\this@ext\XR@ext
    \hyper@linkfile{#4}{#1\this@ext}{#5}%
 \else
    \hyper@linkurl{#4}{file:#1\this@ext\ifx\\#5\\\else\##5\fi}%
 \fi%
}
```

Anchors have a name, and marked text. We have to be careful with the marked text, as if we break off part of something to put a `\special` around it, all hell breaks loose. Therefore, we check the category code of the first token, and only proceed if its safe. Tanmoy sorted this out.

A curious case arises if the original parameter was in braces. That means that `#2` comes here a multiple letters, and the `noexpand` just looks at the first one, putting the rest in the output. Yuck.

```
\long\def\hyper@@anchor#1#2{\@hyper@@anchor#1\relax#2\relax}
\long\def\@hyper@@anchor#1\relax#2#3\relax{%
\ifx\\#1\\#2\hyper@warn{empty link? #1: #2#3}%
\else
 \def\anchor@spot{#2#3}%
 \let\put@me@back\@empty
 \ifx\relax#2\relax
 \else
    \ifhy@nesting
    \else
       \ifcat a\noexpand#2\relax
       \else
         \ifcat 0\noexpand#2 \relax
         \else
%\typeout{Anchor start is not alphanumeric on input line\the\inputlineno}%
           \let\anchor@spot\@empty
           \def\put@me@back{#2#3}%
         \fi
      \fi
    \fi
 \fi
```

```
 \ifhy@activeanchor
      \anchor@spot
 \else
      \hyper@anchor{#1}%
 \fi
 \expandafter\put@me@back
\fi
\let\anchor@spot\@empty
}
```

1.8 Compatibility with the *LaTeX2html* package

Map our macro names on to Nikos', so that documents prepared for that system will work without change.

Note, however, that the whole complicated structure for segmenting documents is not supported; it is assumed that the user will load `html.sty` first, and then `hyperref.sty`, so that the definitions in `html.sty` take effect, and are then overridden in a few circumstances by this package.

```
\let\htmladdimg\hyperimage
```

```
\def\htmladdnormallink#1#2{\href{#2}{#1}}
\def\htmladdnormallinkfoot#1#2{\href{#2}{#1}\footnote{#2}}
\def\htmlref#1#2{% anchor text, label
  \label@hyperref[#2]{#1}%
}
```

This is really too much. The LaTeX2html package defines its own `\hyperref` command, with a different syntax. Was this always here? Its weird, anyway. We interpret it in the 'printed' way, since we are about fidelity to the page.

```
\def\@@latextohtmlX{%
 \let\hhyperref\hyperref
 \def\hyperref##1##2##3##4{% anchor text for HTML
                      % text to print before label in print
                      % label
                      % post-label text in print
  ##2\ref{##4}##3}%
}
```

1.9 Automated LaTeX hypertext cross-references

Emend `\@setref` to put out a hypertext link as well as its normal text (which is used as an anchor).

```
\let\real@setref\@setref
\def\@setref#1#2#3{% csname, extract group, refname
  \ifx#1\relax
   \protect\G@refundefinedtrue
   \nfss@text{\reset@font\bfseries ??}%
   \@latex@warning{Reference '#3' on page \thepage \space
             undefined}%
```

```
  \else
   \hyper@@link
       {\expandafter\@fifthoffive#1}%
       {\expandafter\@fourthoffive#1\@empty\@empty}%
       {\expandafter#2#1\@empty\@empty\null}%
  \fi}
```

Set `\pageref` to be a link. `\realpageref` is available for people who know about these things, which uses a copy of `\@setref`.

```
\def\pageref#1{\expandafter\@pagesetref\csname r@#1\endcsname
                                  \@secondoftwo{#1}}
\def\@pagesetref#1#2#3{% csname, extract macro, ref
  \ifx#1\relax
   \protect\G@refundefinedtrue
   \nfss@text{\reset@font\bfseries ??}%
   \@latex@warning{Reference `#3' on page \thepage \space
            undefined}%
  \else
   \protect\hyper@@link{\expandafter\@fifthoffive#1}%
      {page.\expandafter\@secondoffive#1}%
      {\expandafter\@secondoffive#1}%
  \fi}
\def\realpageref#1{\expandafter\real@setref
  \csname r@#1\endcsname\@secondoffive{#1}}
```

Anything which can be referenced advances some counter; we overload this to put in a hypertext starting point (with no visible anchor), and make a note of that for later use in `\label`. This will fail badly if `\theH<name>` does not expand to a sensible reference. This means that classes or package which introduce new elements need to define an equivalent `\theH<name>` for every `\the<name>`. We do make a trap to make `\theH<name>` be the same as `\arabic{<name>}`, if `\theH<name>` is not defined, but this is not necessarily a good idea.

All the shenanigans is to make sure section numbers etc are always arabic, separated by dots. Who knows how people will set up `\@currentlabel`? If they put spaces in, or brackets (quite legal) then the hypertext processors will get upset.

But this is flaky, and open to abuse. Styles like `subeqn` will mess it up, for starters. Appendices are an issue, too. We just hope to cover most situations. We can at least cope with the standard sectioning structure, allowing for `\part` and `\chapter`.

Start with a fallback for equations

```
\newcommand\theHequation {\theHsection.\arabic{equation}}
\@ifundefined{thepart}{}{\newcommand\theHpart{\arabic{part}}}
\@ifundefined{thechapter}{%
     \newcommand\theHsection        {\arabic{section}}
     \newcommand\theHfigure         {\arabic{figure}}
     \newcommand\theHtable         {\arabic{table}}
  }{%
  \newcommand\theHchapter       {\arabic{chapter}}
  \newcommand\theHfigure        {\theHchapter.\arabic{figure}}
  \newcommand\theHtable        {\theHchapter.\arabic{table}}
```

```
  \newcommand\theHsection       {\theHchapter.\arabic{section}}
  }
\newcommand\theHsubsection    {\theHsection.\arabic{subsection}}
\newcommand\theHsubsubsection {\theHsubsection.\arabic{subsubsection}}
\newcommand\theHparagraph     {\theHsubsubsection.\arabic{paragraph}}
\newcommand\theHsubparagraph  {\theHparagraph.\arabic{subparagraph}}
\newcommand\theHtheorem       {\theHsection.\arabic{theorem}}
\newcommand\theHthm       {\theHsection.\arabic{thm}}
```

Thanks to Greta Meyer (gbd@pop.cwru.edu) for making me realize that enumeration starts at 0 for every list! But `\item` occurs inside `\trivlist`, so check if its a real `\item` before incrementing counters.

```
\let\H@item\item
\newcounter{Item}
\def\theHItem{\arabic{Item}}
\def\item{%
 \@hyper@itemfalse
 \if@nmbrlist\@hyper@itemtrue\fi
 \H@item
}
```

```
\newcommand\theHenumi  {\theHItem}
\newcommand\theHenumii  {\theHItem}
\newcommand\theHenumiii  {\theHItem}
\newcommand\theHenumiv  {\theHItem}
\newcommand\theHHfootnote  {\arabic{Hfootnote}}
\newcommand\theHmpfootnote  {\arabic{mpfootnote}}
\let\theHHmpfootnote\theHHfootnote
\newcommand\theHslide  {\arabic{slide}}
\let\orig@appendix\appendix
\def\appendix{\orig@appendix
  \@ifundefined{thechapter}%
   {\renewcommand\theHsection{\Alph{section}}}%
   {\renewcommand\theHchapter{\Alph{chapter}}}%
}
```

Tanmoy asked for this default handling of undefined `\theH<name>` situations. It really isn't clear what would be ideal, whether to turn off hyperizing of unknown elements, to pick up the textual definition of the counter, or to default it to something like `\arabic{name}`. We take the latter course, slightly worriedly.

```
\let\H@refstepcounter\refstepcounter
```

```
\edef\name@of@eq{equation}%
```

We do not want the handler for `\refstepcounter` to cut in during the processing of `\item` (we handle that separately), so we provide a bypass conditional.

```
\newif\if@hyper@item
\newif\if@skiphyperref
\@hyper@itemfalse
\@skiphyperreffalse
\def\refstepcounter#1{%
```

```
\H@refstepcounter{#1}%
\if@skiphyperref
\else
 \if@hyper@item
   \stepcounter{Item}%
   \hyper@refstepcounter{Item}%
 \else
   \hyper@refstepcounter{#1}%
 \fi
\fi
}
```

AMSLaTeX processes all equations twice; we want to make sure that the hyper stuff is not executed twice, so we use the AMS `\ifmeasuring@`, initialized if AMS math is not used.

```
\@ifpackageloaded{amsmath}{}{\newif\ifmeasuring@\measuring@false}
\def\hyper@refstepcounter#1{%
  \edef\This@name{#1}%
  \ifx\This@name\name@of@eq
     \make@stripped@name{\theequation}%
     \let\theHequation\newname
  \fi
  \@ifundefined{theH#1}{%
  \expandafter\def\csname theH#1\endcsname{\arabic{#1}}%
     }{}%
  \hyper@makecurrent{#1}%
  \ifmeasuring@\else
   \hyper@anchorstart{\@currentHref}\hyper@anchorend
  \fi
}
\def\hyper@makecurrent#1{%
 \edef\@currentHlabel{\csname theH#1\endcsname}%
 \global\edef\@currentHref{#1.\expandafter
    \strip@prefix\meaning\@currentHlabel}%
}
\@ifpackageloaded{fancyvrb}{%
 \def\FV@StepLineNo{%
   \FV@SetLineNo
   \def\FV@StepLineNo{\H@refstepcounter{FancyVerbLine}}%
   \FV@StepLineNo}%
}{}
```

1.9.1 Equations

We want to make the whole equation a target anchor. Overload equation, temporarily reverting to original `\refstepcounter`. If, however, it is in AMS math, we do not do anything, as the tag mechanism is used there (see section 1.9.8).

```
\let\new@refstepcounter\refstepcounter
\let\H@equation\equation
```

```
\let\H@endequation\endequation
\@ifpackageloaded{amsmath}{}{%
 \def\equation{%
 \let\refstepcounter\H@refstepcounter
 \H@equation
 \make@stripped@name{\theequation}%
 \let\theHequation\newname
 \hyper@makecurrent{equation}%
 \hyper@anchorstart{\@currentHref}%
 \let\refstepcounter\new@refstepcounter
 }\def\endequation{\hyper@anchorend\H@endequation}%
}
```

My goodness, why can't LaTeX be consistent? Why is `\eqnarray` set up differently from other objects?

```
\newif\if@eqnstar
\@eqnstarfalse
\let\H@eqnarray\eqnarray
\let\H@endeqnarray\endeqnarray
\def\eqnarray{%
 \let\reserved@a\relax
 \H@eqnarray
 \if@eqnstar\else
 \make@stripped@name{\theequation}%
 \let\theHequation\newname
 \hyper@makecurrent{equation}%
 \hyper@anchorstart{\@currentHref}%
 \fi
}
\def\endeqnarray{%
 \if@eqnstar\else\hyper@anchorend\fi
 \H@endeqnarray
}
```

This is quite heavy-handed, but it works for now. If its an `eqnarray*` we need to disable the hyperref actions. There may well be a cleaner way to trap this. Bill Moss found this.

```
\@namedef{eqnarray*}{\def\@eqncr{\nonumber\@seqncr}\@eqnstartrue\eqnarray}
\@namedef{endeqnarray*}{\nonumber\endeqnarray\@eqnstarfalse}
```

Then again, we have the *subeqnarray* package. Tanmoy provided some code for this:

```
\@ifundefined{subeqnarray}{}%
{\let\H@subeqnarray\subeqnarray
 \let\H@endsubeqnarray\endsubeqnarray
 \def\subeqnarray{%
  \let\reserved@a\relax
  \H@subeqnarray
  \make@stripped@name{\theequation}%
  \let\theHequation\newname
  \hyper@makecurrent{equation}%
  \hyper@anchorstart{\@currentHref}%
```

```
  }%
  \def\endsubeqnarray{%
   \hyper@anchorend
   \H@endsubeqnarray
  }%
\newcommand\theHsubequation {\theHequation\alph{subequation}}%
}
```

The aim of this macro is to produce a sanitized version of its argument, to make it a safe label.

```
\def\make@stripped@name#1{{%
 \escapechar\m@ne
 \global\let\newname\@empty
 \protected@edef\@tempa{#1}%
 \edef\@tempb{%
  \noexpand\@tfor\noexpand\@tempa:=\expandafter\strip@prefix\meaning\@tempa}%
 \@tempb\do{%
  \if{\@tempa\else
    \if}\@tempa\else
      \xdef\newname{\newname\@tempa}%
    \fi
  \fi}}}
```

1.9.2 Footnotes

The footnote mark is a hypertext link, and the text is a target. We separately number the footnotes sequentially through the text, separately from whatever labels the text assigns. Too hard to keep track of markers otherwise.

```
\newcounter{Hfootnote}
\let\H@@footnotetext\@footnotetext
\let\H@@footnotemark\@footnotemark
\let\H@@mpfootnotetext\@mpfootnotetext
\long\def\@mpfootnotetext#1{%
  \H@@mpfootnotetext{%
 \ifhy@nesting
  \hyper@@anchor{\@currentHref}{#1}%
 \else
  \hyper@@anchor{\@currentHref}{\relax}#1%
 \fi
  }%
}
\long\def\@footnotetext#1{%
  \H@@footnotetext{%
 \ifhy@nesting
  \hyper@@anchor{\@currentHref}{#1}%
 \else
  \hyper@@anchor{\@currentHref}{\relax}#1%
 \fi
 }%
}
```

Redefine `\@footnotemark`, borrowing its code (at the cost of getting out of sync with latex.ltx), to take advantage of its white space and hyphenation fudges. If we just overload it, we can get variant documents (the word before the footnote is treated differently). Thanks to David Carlisle and Brian Ripley for confusing and helping me on this.

```
\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
  \H@refstepcounter{Hfootnote}%
  \hyper@makecurrent{Hfootnote}%
  \hyper@linkstart{link}{\@currentHref}%
  \@makefnmark
  \hyper@linkend
  \ifhmode\spacefactor\@x@sf\fi
  \relax
}
\def\realfootnote{\@ifnextchar[\@xfootnote{\stepcounter{\@mpfn}%
     \protected@xdef\@thefnmark{\thempfn}%
     \H@@footnotemark\H@@footnotetext}}
```

But the special footnotes in `\maketitle` are much too hard to deal with properly. Let them revert to plain behaviour.

```
\let\orig@maketitle\maketitle
\def\maketitle{%
 \let\H@@origfootnotemark\@footnotemark
 \let\H@@origfootnotetext\@footnotetext
 \let\@footnotemark\H@@footnotemark
 \let\@footnotetext\H@@footnotetext
 \orig@maketitle
 \ifx\@footnotemark\H@@footnotemark
  \let\@footnotemark\H@@origfootnotemark
 \fi
 \ifx\@footnotetext\H@@footnotetext
  \let\@footnotetext\H@@origfootnotetext
 \fi
}
```

1.9.3 Float captions

Make the float caption the hypertext anchor; curiously enough, we can't just copy the definition of `\@caption`. Its all to do with expansion. It screws up. Sigh.

```
\def\caption{\H@refstepcounter\@captype \@dblarg{\@caption\@captype}}
\long\def\@caption#1[#2]#3{%
  \hyper@makecurrent{\@captype}%
  \par\addcontentsline{\csname
  ext@#1\endcsname}{#1}{\protect\numberline{\csname
  the#1\endcsname}{\ignorespaces #2}}\begingroup
    \@parboxrestore
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces
```

If we cannot have nesting, the anchor is empty.

```
\ifhy@nesting
 \hyper@@anchor{\@currentHref}{#3}%
\else
 \hyper@@anchor{\@currentHref}{\relax}#3%
\fi
}\par
  \endgroup}
```

1.9.4 Bibliographic references

This is not very robust, since many styles redefine these things. The package used to redefine `\@citex` and the like; then we tried adding the hyperref call explicitly into the .aux file. Now we redefine `\bibcite`; this still breaks some citation packages so we have to work around them. But this remains extremely dangerous. Any or all of *achemso*, *chapterbib*, and *drftcite* may break.

However, lets make an attempt to get *natbib* right, because thats a powerful, important package. Patrick Daly (`daly@linmpi.mpg.de`) has provided hooks for us, so all we need to do is activate them.

```
\def\hyper@natlinkstart#1{%
  \hy@backout{#1}%
  \hyper@linkstart{cite}{cite.#1}%
}
\def\hyper@natlinkend{%
  \hyper@linkend
}
\def\hyper@natanchorstart#1{%
     \hyper@anchorstart{cite.#1}%
}
\def\hyper@natanchorend{\hyper@anchorend}

\@ifpackageloaded{natbib}{}{%
\def\bibcite#1#2{%
 \@newl@bel{b}{#1}{\hyper@@link[cite]{}{cite.#1}{#2}}}%
```

`\@BIBLABEL` is working around a 'feature' of RevTEX.

```
\providecommand{\@BIBLABEL}{\@biblabel}%
\def\@lbibitem[#1]#2{%
 \@skiphyperreftrue
 \H@item[\hyper@anchorstart{cite.#2}%
   \@BIBLABEL{#1}\hyper@anchorend\hfill]%
 \@skiphyperreffalse
 \if@filesw{\let\protect\noexpand
   \immediate\write\@auxout{%
     \string\bibcite{#2}{#1}}}%
  \fi
  \ignorespaces
}%
```

Since `\bibitem` is doing its own labelling, call the raw version of `\item`, to avoid extra spurious labels

```
\def\@bibitem#1{%
 \@skiphyperreftrue\H@item\@skiphyperreffalse
 \hyper@anchorstart{cite.#1}\relax\hyper@anchorend
 \if@filesw {\let\protect\noexpand
 \immediate\write\@auxout{%
   \string\bibcite{#1}{\the\value{\@listctr}}}}%
 \fi
 \ignorespaces
}%
}
```

Revtex (bless its little heart) takes over `\bibcite` and looks at the result to measure something. Make this a hypertext link and it goes ape. Therefore, make an anodyne result first, call its business, then go back to the real thing.

```
\@ifclassloaded{revtex}{%
 \hyper@info{*** compatibility with revtex **** }%
 \def\revtex@checking#1#2{%
   \expandafter\let\expandafter\T@temp\csname b@#1\endcsname
   \expandafter\def\csname b@#1\endcsname{#2}%
   \@SetMaxRnhefLabel{#1}%
   \expandafter\let\csname b@#1\endcsname\T@temp
  }%
```

Tanmoy provided this replacement for CITEX. Lord knows what it does.

```
\@ifundefined{@CITE}{\def\@CITE{\@cite}}{}
\def\@CITEX[#1]#2{%
  \let\@citea\@empty
  \leavevmode\unskip$^{\scriptstyle
  \@CITE{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb}%
     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
     \@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
       \G@refundefinedtrue
       \@latex@warning
         {Citation `\@citeb' on page \thepage \space undefined}}%
       {{\csname b@\@citeb\endcsname}}}}{#1}}$}
```

No, life is too short. I am not going to understand the Revtex `\@collapse` macro, I shall just restore the original behaviour of `\@citex`; sigh. This is SO vile.

```
\def\@citex[#1]#2{%
  \let\@citea\@empty
  \@cite{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb}%
     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
     \@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
       \G@refundefinedtrue
```

```
      \@latex@warning
        {Citation `\@citeb' on page \thepage \space undefined}}%
      {\hbox{\csname b@\@citeb\endcsname}}}}{#1}}
}{}
```

Override Peter Williams' Harvard package; we have to a) make each of the citation types into a link; b) make each citation write a backref entry, and c) kick off a backreference section for each bibliography entry.

```
\@ifpackageloaded{harvard}{%
 \hyper@info{*** compatibility with harvard **** }%
 \hy@raiselinksfalse
 \def\harvardcite#1#2#3#4{%
  \global\@namedef{HAR@fn@#1}{\hyper@@link[cite]{}{cite.#1}{#2}}%
  \global\@namedef{HAR@an@#1}{\hyper@@link[cite]{}{cite.#1}{#3}}%
  \global\@namedef{HAR@yr@#1}{\hyper@@link[cite]{}{cite.#1}{#4}}%
  \global\@namedef{HAR@df@#1}{\csname HAR@fn@#1\endcsname}%
 }%
 \def\HAR@citetoaux#1{%
   \if@filesw\immediate\write\@auxout{\string\citation{#1}}\fi%
   \ifhy@backref
    \ifx\@empty\@currentlabel\else
     \@bsphack
     \protected@write\@auxout{}%
     {\string\@writefile{brf}%
       {\string\backcite{#1}{{\@currentlabel}{\thepage}{\@currentHref}}}}%
     \@esphack
    \fi
  \fi
 }
 \def\harvarditem{\@ifnextchar[{\@harvarditem}{\@harvarditem[\null]}}
 \def\@harvarditem[#1]#2#3#4#5\par{%
 \item[]%
 \hyper@anchorstart{cite.#4}\relax\hyper@anchorend%
 \if@filesw{ \def\protect##1{\string ##1\space}%
 \ifthenelse{\equal{#1}{\null}}
  {\def\next{{#4}{#2}{#2}{#3}}}
  {\def\next{{#4}{#2}{#1}{#3}}}
 \immediate\write\@auxout{\string\harvardcite\codeof\next}%
 }\fi%
 \protect\hspace*{-\labelwidth}\protect\hspace*{-\labelsep}\ignorespaces%
 #5
 \ifhy@backref
  \newblock
  \backref{\csname br@#4\endcsname}%
 \fi
 \par
}%
}{}
```

1.9.5 Page numbers

Give every page an automatic number anchor. This involves, sigh, overloading LaTeX's output bits and pieces, which must be dangerous. This used to be `\@shipoutsetup`, now `\@begindvi`. We cannot even overload this, as it sets itself to null. SIGH.

```
\def\@begindvi{%
  \unvbox \@begindvibox
 \ifhy@pageanchor
  \@hyperfixhead
  \global\let \@begindvi \@hyperfixhead
 \else
  \global\let \@begindvi \@empty
 \fi
}
\def\hyperpageanchor{%
   \ifhy@plainpages
     \hyper@anchorstart{page.\arabic{page}}\hyper@anchorend
   \else
     \hyper@anchorstart{page.\thepage}\hyper@anchorend
   \fi
 }
```

This is needed for some unremembered reason...

```
\let\HYPERPAGEANCHOR\hyperpageanchor
\def\@hyperfixhead{%
 \let\H@old@thehead\@thehead
   \ifhy@plainpages
     \gdef\@foo{\hyper@@anchor{page.\arabic{page}}}%
   \else
     \gdef\@foo{\hyper@@anchor{page.\thepage}}%
   \fi
   \expandafter\ifx\expandafter\@empty\H@old@thehead
     \def\H@old@thehead{\hfil}\fi
  \def\@thehead{\@foo\relax\H@old@thehead}%
}
```

1.9.6 Table of contents

TV Raman noticed that people who add arbitrary material into the TOC generate a bad or null link. We used to avoid that by checking if the current destination was empty; instead, generate an arbitrary anchor at this point, and link to it. A separate macro is used for the automatic uses of `\addcontentsline` in section headings. THIS DOES NOT WORK YET!!!!!

```
\def\toclevel@part{-1}
\def\toclevel@chapter{0}
\def\toclevel@section{1}
\def\toclevel@subsection{2}
\def\toclevel@subsection{3}
\def\toclevel@paragraph{4}
```

```
\def\toclevel@subparagraph{5}
\def\@nameoftoc{toc}
\newcount\OddToc
\def\manual@addcontentsline#1#2#3{%
  \advance\OddToc by 1
  \hyper@@anchor{toc\the\OddToc}{\relax}%
  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}{toc\the\OddToc}}%
  \@writetorep{#3}{toc\the\OddToc}{\csname toclevel@#2\endcsname}%
}
\def\addcontentsline#1#2#3{%
  \ifx\@currentHref\@empty
  \hyper@warn{contentsline with no destination
     at line \the\inputlineno}\fi
  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}{\@currentHref}}%
}
\def\contentsline#1#2#3#4{%
 \ifx\\#4\\%
  \csname l@#1\endcsname{#2}{#3}%
\else
 \csname l@#1\endcsname{%
    \hyper@linkstart{link}{#4}{#2}\hyper@linkend}%
 {#3}%
\fi
}
```

1.9.7 New counters

The whole theorem business makes up new counters on the fly; we are going to intercept this. Sigh. Do it at the level where new counters are defined.

```
\let\H@definecounter\@definecounter
\def\@definecounter#1{%
   \H@definecounter{#1}%
   \expandafter\def\csname theH#1\endcsname       {\arabic{#1}}%
}
```

But what if they have used the optional argument to e.g. `\newtheorem` to determine when the numbering is reset? OK, we'll trap that too.

```
\let\H@newctr\@newctr
\def\@newctr#1[#2]{%
  \H@newctr#1[#2]%
  \expandafter\def\csname theH#1\endcsname
    {\csname the#2\endcsname.\arabic{#1}}%
}
```

1.9.8 AMS LaTeX compatibility

Oh, no, they don't use anything as simple as `\refstepcounter` in the AMS! We need to intercept some low-level operations of theirs. Damned if we are going to try and work out what the hell they get up to. Just stick a label of 'AMS' on the front, and use the label

they worked out. If that produces something invalid, I give up. They'll change all the code again anyway, I expect.

```
\let\Hmake@df@tag@@\make@df@tag@@
\def\make@df@tag@@#1{%
 \Hmake@df@tag@@{#1}%
 \global\edef\@currentHref{AMS.\theequation}%
}
\let\H@seteqlabel\@seteqlabel
\def\@seteqlabel#1{%
 \H@seteqlabel{#1}%
 \global\edef\@currentHref{AMS.\theequation}%
}
```

This code I simply cannot remember what I was trying to achieve. The final result seems to do nothing anyway.

```
\let\H@tagform@\tagform@
\def\tagform@#1{%
  \maketag@@@{\hyper@@anchor{\@currentHref}%
  {(\ignorespaces#1\unskip)}}%
}
\def\eqref#1{\textup{\H@tagform@{\ref{#1}}}}
```

1.9.9 Included figures

Simply intercept the low level graphics package macro.

```
\ifhy@figures
 \let\hy@Gin@setfile\Gin@setfile
 \def\Gin@setfile#1#2#3{%
 \hyperimage{#3}{\hy@Gin@setfile{#1}{#2}{#3}}%
 }
\fi
```

1.10 hyperindex entries

Hyper-indexing works crudely, by forcing code onto the end of the index entry with the | feature; this puts a hyperlink around the printed page numbers. It will not proceed if the author has already used the | specifier for something like emboldening entries. That would make Makeindex fail (cannot have two | specifiers). The solution is for the author to use generic coding, and put in the requisite `\hyperpage` in his/her own macros along with the boldness.

This section is poor stuff; it's open to all sorts of abuse. Sensible large projects will design their own indexing macros any bypass this.

```
\ifhy@hyperindex
 \def\@wrindex#1{\@@wrindex#1||\\}
 \def\@@wrindex#1|#2|#3\\{%
 \ifx\\#2\\%
   \protected@write\@indexfile{}%
      {\string\indexentry{#1|hyperpage}{\thepage}}%
```

```
 \else
   \protected@write\@indexfile{}%
      {\string\indexentry{#1|#2}{\thepage}}%
 \fi
 \endgroup
 \@esphack
}
\fi
```

This again is quite flaky, but allow for the common situation of a page range separated by en-rule. We split this into two different hyperlinked pages.

```
\def\hyperpage#1{\@hyperpage#1----\\}
\def\@hyperpage#1--#2--#3\\{%
 \ifx\\#2\\%
    \@commahyperpage{#1}%
 \else
  \hyperlink{page.#1}{#1}--\hyperlink{page.#2}{#2}%
 \fi
}
\def\@commahyperpage#1{\@@commahyperpage#1, ,\\}
\def\@@commahyperpage#1, #2,#3\\{%
 \ifx\\#2\\%
   \hyperlink{page.#1}{#1}%
 \else
   \hyperlink{page.#1}{#1}, \hyperlink{page.#2}{#2}%
 \fi
}
```

1.11 Compatibility with seminar slide package

```
\let\oldslide@heading\slide@heading
\def\slide@heading[#1]#2{%
 \@writetorep{#1}{slide.\theslide}{0}%
 \oldslide@heading[#1]{#2}%
}
\@ifundefined{listofslides}{}{%
 \def\l@slide#1#2#3{%
    \slide@undottedcline{\slidenumberline{#3}{\hyperlink{slide.#2}{#2}}}{}}%
}
```

1.12 Localized nullifying of package

Sometimes we just don't want the wretched package interfering with us. Define an environment we can put in manually, or include in a style file, which stops the hypertext functions doing anything. This is used, for instance, in the Elsevier classes, to stop `hyperref` playing havoc in the front matter.

```
\def\NoHyper{%
    \def\hyper@link@[##1]##2##3##4{##4}%
    \def\hyper@@anchor##1{}%
    \gdef\hyper@link##1##2##3{##3}%
```

```
    \def\hyper@anchorstart##1{}%
    \let\hyper@anchorend\@empty
    \def\hyper@linkstart##1##2{}%
    \let\hyper@linkend\@empty
    \def\hyper@linkurl##1##2{##1}%
    \def\hyper@linkfile##1##2##3{##1}%
    \let\hy@backout\@gobble
}
\def\stop@hyper{%
    \def\hyper@link@[##1]##2##3##4{##4}%
    \let\hy@backout\@gobble
    \def\hyper@@anchor##1{}%
    \def\hyper@link##1##2##3{##3}%
    \def\hyper@anchorstart##1{}%
    \let\hyper@anchorend\@empty
    \def\hyper@linkstart##1##2{}%
    \let\hyper@linkend\@empty
    \def\hyper@linkurl##1##2{##1}%
    \def\hyper@linkfile##1##2##3{##1}%
}
\let\endNoHyper\@empty
```

1.13 Low-level utility macros

We need unrestricted access to the #, ~ and " characters, so make them nice macros.

```
\edef\hyper@hash{\string#}
\edef\hyper@tilde{\string~}
\edef\hyper@quote{\string"}
\let\@currentHref\@empty
```

1.14 Setup

```
\ifhy@figures
 \hyper@info{Hyper figures ON}
\else
 \hyper@info{Hyper figures OFF}
\fi
\ifhy@nesting
 \hyper@info{Link nesting ON}
\else
 \hyper@info{Link nesting OFF}
\fi
\ifhy@hyperindex
 \hyper@info{Hyper index ON}
\else
 \hyper@info{Hyper index OFF}
\fi
\ifhy@plainpages
 \hyper@info{Plain pages ON}
```

```
\else
 \hyper@info{Plain pages OFF}
\fi
\ifhy@backref
 \hyper@info{backreferencing ON}
\else
 \hyper@info{backreferencing OFF}
\fi
\ifhy@colorlinks
    \hyper@info{Link coloring ON}
\else
    \hyper@info{Link coloring OFF}
\fi
```

We give the start of document a special label; this is used in backreferencing-by-section, to allow for cites before any sectioning commands.

```
\AtBeginDocument{\PDF@SetupDoc
   \hyper@anchorstart{Doc-Start}\hyper@anchorend}
```

1.15 Compatibility of aux and toc files

[This section is by David Carlisle]. Some extra tests so that the hyperref package may be removed or added to a document without having to remove .aux and .toc files. All the code is delayed to `\begin{document}`

```
\AtBeginDocument{%
```

First the code to deal with removing the hyperref package from a document.

Write some stuff into the aux file so if the next run is done without hyperref, then `\contentsline` and `\newlabel` are defined to cope with the extra arguments.

```
  \immediate\write\@auxout{%
    \string\ifx\string\hyper@anchor\string\@undefined^^J%
       \global\let\string\oldcontentsline\string\contentsline^^J%
       \gdef\string\contentsline\string#1\string#2\string#3\string#4{%
           \string\oldcontentsline{\string#1}{\string#2}{\string#3}}^^J%
       \global\let\string\oldnewlabel\string\newlabel^^J%
       \gdef\string\newlabel\string#1\string#2{%
           \string\newlabelxx{\string#1}\string#2}^^J%
       \gdef\string\newlabelxx%
           \string#1\string#2\string#3\string#4\string#5\string#6{%
             \string\oldnewlabel{\string#1}{{\string#2}{\string#3}}}^^J%
```

But the new aux file will be read again at the end, with the normal definitions expected, so better put things back as they were.

```
       \string\AtEndDocument{%
          \let\string\contentsline\string\oldcontentsline^^J%
          \let\string\newlabel\string\oldnewlabel}^^J%
```

If the document is being run with hyperref put this definition into the aux file, so we can spot it on the next run.

```
    \string\else^^J%
```

```
      \global\let\string\hyper@last\relax^^J%
    \string\fi^^J%
    }%
```

Now the code to deal with adding the hyperref package to a document with aux and toc written the standard way.

If hyperref was used last time, do nothing. If it was not used, or an old version of hyperref was used, don't use that TOC at all but generate a warning. Not ideal, but better than failing with pre-5.0 hyperref TOCs.

```
\ifx\hyper@last\@undefined
  \def\@starttoc#1{%
    \begingroup
      \makeatletter
      \IfFileExists{\jobname.#1}%
         {\hyper@warn{old #1 file detected, not used; run LaTeX again}}{}%
      \if@filesw
        \expandafter\newwrite\csname tf@#1\endcsname
        \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
      \fi
      \@nobreakfalse
    \endgroup}%
  \def\newlabel#1#2{\@newl@bel r{#1}{#2{}{}{}{}}}
\fi}
```

⟨/package⟩
⟨∗check⟩
```
\ifhy@driverloaded\endinput\fi
\hy@driverloadedtrue
```
⟨/check⟩
⟨∗pdftex⟩

2 Configuration files

2.1 pdftex

This driver is for Han The Thanh's TEX variant which produces PDF directly. This has new primitives to do PDF things, which usually translate almost directly to PDF code, so there is a lot of flexibility which we do not at present harness.

First define the anchors:

```
\def\new@pdflink#1{%
```
⟨pdftexold⟩ `\pdfdest name {#1!} \@pdfview}`
⟨!pdftexold⟩ `\pdfdest name {#1} \@pdfview }`
```
\let\pdf@endanchor\@empty
%
```

Now the links; the interesting part here is the set of attributes which define how the link looks. We probably want to add a border and color it, but there are other choices. This directly translates to PDF code, so consult the manual for how to change this. We will add an interface at some point.

```
\AtBeginDocument{%
  \ifhy@colorlinks
   \def\pdfBorderAttrs{/Border [0 0 0]}%
  \fi
}
\def\@pdfborder{0 0 1}%
\def\pdfBorderAttrs{/Border [\@pdfborder]}
\def\find@pdflink#1#2{%
  \leavevmode\pdfannotlink
  attr{\pdfBorderAttrs /C [\CurrentBorderColor]}
⟨pdftexold⟩  goto name {#2!}%
⟨!pdftexold⟩  goto name {#2}%
  \colorlink{\csname @#1color\endcsname}%
}
\def\close@pdflink{\pdfendlink\hyper@resetcolor}
\def\hyper@anchor#1{\new@pdflink{#1}\anchor@spot\pdf@endanchor}
\def\hyper@anchorstart#1{\new@pdflink{#1}\hy@activeanchortrue}
\def\hyper@anchorend{\pdf@endanchor\hy@activeanchorfalse}
\def\hyper@linkstart#1#2{%
  \edef\CurrentBorderColor{\csname @#1bordercolor\endcsname}%
  \find@pdflink{#1}{#2}}
\def\hyper@linkend{\close@pdflink}
\def\hyper@link#1#2#3{%
    \edef\CurrentBorderColor{\csname @#1bordercolor\endcsname}%
    \find@pdflink{#1}{#2}#3\close@pdflink
}
\def\CurrentBorderColor{\@linkbordercolor}
\def\hyper@linkurl#1#2{%
   \bgroup
   \hyper@chars
   \leavevmode\pdfannotlink
   attr{\pdfBorderAttrs /C [\@urlbordercolor]}
⟨∗!pdftexold⟩
   user{/Subtype /Link
    /A <<
        /Type /Action
        /S /URI
        /URI (#2)
    >>}{\colorlink{\@urlcolor}#1}%
⟨/!pdftexold⟩
⟨∗pdftexold⟩
   user{/S /URI /URI (#2)}{\colorlink{\@urlcolor}#1}%
⟨/pdftexold⟩
   \pdfendlink
   \egroup
}
\def\hyper@linkfile#1#2#3{% anchor text, filename, linkname
   \bgroup
   \leavevmode\pdfannotlink
   attr{\pdfBorderAttrs /C [\@filebordercolor]}
```

```
   goto file{#2} name{#3}{\colorlink{\@filecolor}#1}%
   \pdfendlink
   \egroup
}
\def\@pdfproducer{pdfTeX}
⟨*!pdftexold⟩
\def\PDF@SetupDoc{%
  \pdfcatalog {            % Catalog dictionary of PDF output.
    /PageMode \@pdfpagemode
    /URI (\@baseurl)
    } openaction goto page \@pdfstartpage {\@pdfstartview}
 \pdfinfo {
  /Author (\@pdfauthor)
  /Title (\@pdftitle)
  /Subject (\@pdfsubject)
  /Creator ( \@pdfcreator)
  /Producer ( \@pdfproducer)
  /Keywords (\@pdfkeywords)
  \ifx\@pdfpagescrop\@empty\else
   /CropBox [\@pdfpagescrop]
  \fi
  }%
}
⟨/!pdftexold⟩
⟨*pdftexold⟩
\def\PDF@SetupDoc{%
\edef\x@pdfcatalog{%
  pagemode {\@pdfpagemode}
  \ifx\@baseurl\@empty\else uri {\@baseurl} \fi}%
\edef\x@pdfinfo{%
  author {\@pdfauthor}
  title {\@pdftitle}
  subject {\@pdfsubject}
  keywords {\@pdfkeywords}
 \ifx\@pdfpagescrop\@empty\else
  \pdfpagesattr={/CropBox [\@pdfpagescrop]}
 \fi
}%
 \expandafter\pdfcatalog \x@pdfcatalog
 \expandafter\pdfinfo \x@pdfinfo
}
⟨/pdftexold⟩
```

Let us explicitly turn on PDF generation; they can reverse this decision in the document, but since we are emitting PDF links anyway, we *must* be in PDF mode.

```
\pdfoutput=1
\pdfcompresslevel=9
\pdfpagewidth\paperwidth
\pdfpageheight\paperheight
⟨*!pdftexold⟩
```

```
\def\Acrobatmenu#1#2{%
   \leavevmode\pdfannotlink
   attr{\pdfBorderAttrs /C [\@menubordercolor]}
   user{
    /Subtype /Link
    /A <<
       /S /Named /N /#1
    >>
    }{\colorlink{\@menucolor}#2}%
   \pdfendlink
}
```
⟨/!pdftexold⟩
⟨∗pdftexold⟩
```
\def\Acrobatmenu#1#2{%
   \leavevmode\pdfannotlink
   attr{\pdfBorderAttrs /C [\@menubordercolor]}
   user{/S /Named /N /#1}{\colorlink{\@menucolor}#2}%
   \pdfendlink
}
```
⟨/pdftexold⟩
⟨/pdftex⟩
⟨∗hypertex⟩

2.2 hypertex

The HyperTEX specification (this is borrowed from an article by Arthur Smith) says that conformant viewers/translators must recognize the following set of `\special` commands:

href: `html:<a href = "href_string">`

name: `html:<a name = "name_string">`

end: `html:</a>`

image: `html:<img src = "href_string">`

base_name: `html:<base href = "href_string">`

The *href*, *name* and *end* commands are used to do the basic hypertext operations of establishing links between sections of documents. The *image* command is intended (as with current html viewers) to place an image of arbitrary graphical format on the page in the current location. The *base_name* command is be used to communicate to the *dvi* viewer the full (URL) location of the current document so that files specified by relative URL's may be retrieved correctly.

The *href* and *name* commands must be paired with an *end* command later in the TEX file — the TEX commands between the two ends of a pair form an *anchor* in the document. In the case of an *href* command, the *anchor* is to be highlighted in the *dvi* viewer, and when clicked on will cause the scene to shift to the destination specified by *href_string*. The *anchor* associated with a name command represents a possible location to which other hypertext links may refer, either as local references (of the form

`href="#name_string"` with the *name_string* identical to the one in the name command) or as part of a URL (of the form *URL#name_string*). Here *href_string* is a valid URL or local identifier, while name_string could be any string at all: the only caveat is that '"' characters should be escaped with a backslash (\), and if it looks like a URL name it may cause problems.

```
\def\PDF@SetupDoc{%
 \ifx\@baseurl\@empty\else
  \special{html:<base href="\@baseurl">}%
 \fi
}
\def\hyper@anchor#1{%
   {\let\protect=\string\special{html:<A name=\hyper@quote #1\hyper@quote>}}%
   \hy@activeanchortrue
   \bgroup\colorlink{\@anchorcolor}\anchor@spot\egroup
   \special{html:</A>}%
   \hy@activeanchorfalse
}
\def\hyper@anchorstart#1{%
 {\hyper@chars\special{html:<A name=\hyper@quote#1\hyper@quote>}}%
  \hy@activeanchortrue
}
\def\hyper@anchorend{%
  \special{html:</A>}%
  \hy@activeanchorfalse
}
\def\@urltype{url}
\def\hyper@linkstart#1#2{%
  \colorlink{\csname @#1color\endcsname}%
  \def\@tempa{#1}\ifx\@tempa\@urltype
    \special{html:<A href=\hyper@quote#2\hyper@quote>}%
  \else
   {\hyper@chars\special{html:<A href=\hyper@quote\##2\hyper@quote>}}%
  \fi
}
\def\hyper@linkend{%
  \special{html:</A>}%
  \hyper@resetcolor
}
\def\hyper@linkfile#1#2#3{%
 \hyper@linkurl{#1}{file:#2\ifx\\#3\\\else\##3\fi}%
}
\def\hyper@linkurl#1#2{%
```

If we want to raise up the final link `\special`, we need to get its height; ask me why LaTeX constructs make this totally foul up, and make us revert to basic TeX. I do not know.

```
 \ifhy@raiselinks
  \setbox\@tempboxa=\hbox{#1}%
  \@linkdim\dp\@tempboxa
  \leavevmode\lower\@linkdim\hbox{%
```

```
      {\hyper@chars\special{html:<A href=\hyper@quote#2\hyper@quote>}}%
   }%
  {\colorlink{\@urlcolor}#1}%
  \@linkdim\ht\@tempboxa
```

Because of the interaction with the dvihps processor, we have to subtract a little from the height. This is not clean, or checked. Check with Mark Doyle about what gives here. It may not be needed with the new dvips (Jan 1997).

```
  \advance\@linkdim by -6.5\p@
  \raise\@linkdim\hbox{\special{html:</A>}}%
 \else
  {\hyper@chars
   \special{html:<A href=\hyper@quote#2\hyper@quote>}%
   \colorlink{\@urlcolor}#1}%
   \special{html:</A>}%
 \fi
}
\def\hyper@link#1#2#3{%
  \hyper@linkurl{#3}{\##2}%
}
\def\hyper@image#1#2{%
  {\hyper@chars
   \special{html:<img src=\hyper@quote#1\hyper@quote>}}}
⟨/hypertex⟩
⟨*dviwindo⟩
```

2.3 dviwindo

[This was developed by David Carlisle]. Within a file dviwindo hyperlinking is used, for external URL's a call to `\wwwbrowser` is made. (You can define this command before or after loading the hyperref package if the default `c:/netscape/netscape` is not suitable) Dviwindo could in fact handle external links to dvi files on the same machine without calling a web browser, but that would mean parsing the URL to recognise such, and this is currently not done.

This was more or less blindly copied from the hypertex cfg. For dviwindo, LaTeX must specify the size of the active area for links. For some hooks this information is available but for some, the start and end of the link are specified separately in which case a fixed size area of 10000000sp wide by `\baselineskip` high is used.

```

\providecommand\wwwbrowser{c:\string\netscape\string\netscape}
\def\hyper@anchor#1{%
   {\let\protect=\string\special{mark: \hyper@quote #1\hyper@quote}}%
   \hy@activeanchortrue
   \bgroup\colorlink{\@anchorcolor}\anchor@spot\egroup
   \hy@activeanchorfalse
}
\def\hyper@anchorstart#1{%
  \special{mark: \hyper@quote#1\hyper@quote}%
```

```
  \hy@activeanchortrue
}
\def\hyper@anchorend{%
  \hy@activeanchorfalse
}
\def\hyper@linkstart#1#2{%
  \colorlink{\csname @#1color\endcsname}%
  \special{button:  10000000 \number\baselineskip \space
      \hyper@quote#2\hyper@quote}%
}
\def\hyper@linkend{%
  \hyper@resetcolor
}
\def\hyper@link#1#2#3{%
 \ifhy@raiselinks
  \setbox\@tempboxa=\hbox{#3}%
  \@linkdim\dp\@tempboxa
  \leavevmode\lower\@linkdim\hbox{%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
      \hyper@quote#2\hyper@quote}%
  {\colorlink{\csname @#1color\endcsname}#3}}
  \@linkdim\ht\@tempboxa
  \advance\@linkdim by -6.5\p@
  \raise\@linkdim\hbox{}%
 \else
  \setbox\@tempboxa=\hbox{#3}%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
      \hyper@quote#2\hyper@quote}%
  {\colorlink{\csname @#1color\endcsname}#3}%
 \fi
}
\def\hyper@linkurl#1#2{%
\bgroup\hyper@chars
 \ifhy@raiselinks
  \setbox\@tempboxa=\hbox{#1}%
  \@linkdim\dp\@tempboxa
  \leavevmode\lower\@linkdim\hbox{%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
        launch: \wwwbrowser\space
       \hyper@quote#2\hyper@quote}%
  {\colorlink{\@urlcolor}#1}}%
  \@linkdim\ht\@tempboxa
  \advance\@linkdim by -6.5\p@
  \raise\@linkdim\hbox{}%
 \else
  \setbox\@tempboxa=\hbox{#1}%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
        launch: \wwwbrowser\space
      \hyper@quote#2\hyper@quote}%
  {\colorlink{\@urlcolor}#1}%
```

```
 \fi
\egroup
}
\def\hyper@linkfile#1#2#3{%
\bgroup\hyper@chars
 \ifhy@raiselinks
  \setbox\@tempboxa=\hbox{#1}%
  \@linkdim\dp\@tempboxa
  \leavevmode\lower\@linkdim\hbox{%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
      \hyper@quote#3\hyper@quote\space file: \hyper@quote#2\hyper@quote}%
  {\colorlink{\@filecolor}#1}}%
  \@linkdim\ht\@tempboxa
  \advance\@linkdim by -6.5\p@
  \raise\@linkdim\hbox{}%
 \else
  \setbox\@tempboxa=\hbox{#1}%
  \special{button:  \number\wd\@tempboxa\space \number\ht\@tempboxa\space
      \hyper@quote#3\hyper@quote\space file: \hyper@quote#2\hyper@quote}%
  {\colorlink{\@filecolor}#1}%
 \fi
\egroup
}
\def\@pdfproducer{dviwindo + Distiller}
\def\PDF@SetupDoc{
 \special{PDF: Keywords \@pdfkeywords}%
 \special{PDF: Title \@pdftitle}%
 \special{PDF: Creator \@pdfcreator}%
 \special{PDF: Author \@pdfauthor}%
 \special{PDF: Producer \@pdfproducer}%
 \special{PDF: Subject \@pdfsubject}%
 \ifx\@baseurl\@empty\else
  \special{PDF: Base \@baseurl}%
 \fi
 \ifx\@pdfpagescrop\@empty\else
  \special{PDF: BBox \@pdfpagescrop}%
 \fi
}
⟨/dviwindo⟩
⟨∗dvipdf | pdfmark | pdftex⟩
⟨∗pdfmark | dvipdf⟩
```

2.4 Direct pdfmark support (dvipdf and pdfmark)

```
\def\hyper@anchor#1{%
   {\pdfmark[\anchor@spot]{pdfmark=/DEST,linktype=anchor,View=/\@pdfview,Dest=#1}}%
}
⟨∗dvipdf⟩
\def\hyper@anchorstart#1{\hy@activeanchortrue}
```

```
\def\hyper@anchorend{\hy@activeanchorfalse}
\def\hyper@linkstart#1#2{%
   \colorlink{\csname @#1color\endcsname}%
   \global\edef\hyper@currentanchor{#2}%
}
\def\hyper@linkend{%
\hyper@resetcolor
}
```
⟨/dvipdf⟩
⟨∗pdfmark⟩
```
\@ifundefined{hyper@anchorstart}{}{\endinput}
\def\hyper@anchorstart#1{%
  \PSHyperAnchorStart
  \global\edef\hyper@currentanchor{#1}%
  \hy@activeanchortrue
}
\def\hyper@anchorend{%
   \PSHyperAnchorEnd
   \pdfmark{pdfmark=/DEST,linktype=anchor,View=/\@pdfview,
       Dest=\hyper@currentanchor,
       Rect=\pdf@bbox}%
  \hy@activeanchorfalse
}
\def\hyper@linkstart#1#2{%
   \ifhy@breaklinks\else
    \leavevmode\hbox\bgroup\color@begingroup
   \fi
   \colorlink{\csname @#1color\endcsname}%
   \PSHyperLinkStart
   \global\edef\hyper@currentanchor{#2}%
   \gdef\hyper@currentlinktype{#1}%
}
\def\hyper@linkend{%
    \PSHyperLinkEnd
    \edef\@foo{\csname @\hyper@currentlinktype bordercolor\endcsname}%
       \pdfmark{pdfmark=/ANN,linktype=link,Subtype=/Link,
           Dest=\hyper@currentanchor,
           Border=\@pdfborder,Color=\@foo,Rect=\pdf@bbox}%
   \hyper@resetcolor
   \ifhy@breaklinks\else\color@endgroup\egroup\fi
}
```
⟨/pdfmark⟩
```
\def\hyper@image#1#2{%
  \hyper@linkurl{#2}{#1}}
\def\hyper@link#1#2#3{%
    \edef\@foo{\csname @#1bordercolor\endcsname}%
    \bgroup
```
⟨dvipdf⟩ `\pdfmark[#3]{pdfmark=/LNK,{},linktype=#1,Border=\@pdfborder,Color=\@foo,Dest=`
⟨pdfmark⟩ `\pdfmark[#3]{Color=\@foo,linktype=#1,Border=\@pdfborder,pdfmark=/ANN,Subtyp`

```
    \egroup
}
\newtoks\pdf@docset
\def\@pdfproducer{dvips + Distiller}
\def\PDF@SetupDoc{%
 \pdfmark{pdfmark=/DOCINFO,
  Title=\@pdftitle,
  Subject=\@pdfsubject,
  Creator= \@pdfcreator,
  Author=\@pdfauthor,
  Producer= \@pdfproducer,
  Keywords=\@pdfkeywords
  }%
 \ifx\@baseurl\@empty\def\@dobaseurl{}\else
     \def\@dobaseurl{}%
 \fi
 \pdfmark{pdfmark=/DOCVIEW,
   Page=\@pdfstartpage,
   View=\@pdfstartview,
   URI={<< /Base (\@baseurl) >>},
   PageMode=\@pdfpagemode
   }
 \ifx\@pdfpagescrop\@empty\else
   \pdfmark{pdfmark=/PAGES,CropBox=\@pdfpagescrop}%
 \fi
}
```

We define a single macro, pdfmark, which uses the 'keyval' system to define the various allowable keys; these are *exactly* as listed in the pdfmark reference for Acrobat 3.0. The only addition is `pdfmark` which specifies the type of pdfmark to create (like ANN, LINK etc). The surrounding round and square brackets in the pdfmark commands are supplied, but you have to put in / characters as needed for the values.

```
\def\pdfmark{\@ifnextchar[{\pdfmark@}{\pdfmark@[]}}
\def\pdfmark@[#1]#2{%
    \edef\goforit{\noexpand\pdf@toks={ \the\pdf@defaulttoks}}%
    \goforit
    \let\pdf@type\relax
    \setkeys{PDF}{#2}%
    \ifx\pdf@type\relax
       \hyper@warn{no pdfmark type specified in #2!}%
       \ifx\\#1\\\relax\else\pdf@rect{#1}\fi
    \else
       \bgroup
       \ifx\\#1\\\relax
       \else
         \colorlink{\@ifundefined{@\pdf@linktype color}%
                     {\@linkcolor}%
                     {\csname @\pdf@linktype color\endcsname}}%
         \pdf@rect{#1}%
       \fi
```

```
⟨pdfmark⟩   \literalps@out{[\the\pdf@toks\space \pdf@type\space pdfmark}%
⟨dvipdf⟩   \literalps@out{/ANN >>}%
        \egroup
    \fi
}
```

The complicated bit is working out the right enclosing rectangle of some piece of TEX text, needed by the /Rect key. This solution originates with Toby Thain (tobyt@netspace.net.au).

```
\newsavebox{\pdf@box}
\def\pdf@rect#1{%
⟨dvipdf⟩    \literalps@out{/ANN \pdf@type \the\pdf@toks\space <<}#1
    \leavevmode
    \sbox\pdf@box{#1}%
    \dimen@\ht\pdf@box
    \leavevmode\lower\dp\pdf@box\hbox{\PSHyperRectStart}%
```

If the text has to be horizontal mode stuff then just unbox the saved box like this, which saves executing it twice, which can mess up counters etc (thanks DPC...).

```
    \ifhy@breaklinks\unhbox\else\box\fi\pdf@box
```

but if it can have multiple paragraphs you'd need one of these, but in that case the measured box size would be wrong anyway. `\ifhy@breaklinks#1\else\box\pdf@box\fi` `\ifhy@breaklinks{#1}\else\box\pdf@box\fi`

```
    \raise\dimen@\hbox{\PSHyperRectEnd}%
    \pdf@addtoks{[\pdf@bbox]}{Rect}%
}
```

All the supplied material is stored in a token list; since I do not feel sure I quite understand these, things may not work as expected with expansion. We'll have to experiment.

```
\newtoks\pdf@toks
\newtoks\pdf@defaulttoks
\pdf@defaulttoks={ }%
\def\pdf@addtoks#1#2{%
    \edef\goforit{\pdf@toks{\the\pdf@toks\space /#2 #1}}%
    \goforit
}
\def\PDFdefaults#1{%
 \pdf@defaulttoks={#1}%
}
```

This is the list of allowed keys. See the Acrobat manual for an explanation.

```
% what is the type of pdfmark?
\define@key{PDF}{pdfmark}{\def\pdf@type{#1}}
% what is the link type?
\define@key{PDF}{linktype}{\def\pdf@linktype{#1}}
\def\pdf@linktype{link}
% parameter is a name
\define@key{PDF}{Action}{\pdf@addtoks{#1}{Action}}
% parameter is a array
\define@key{PDF}{Border}{\pdf@addtoks{[#1]}{Border}}
% parameter is a array
\define@key{PDF}{Color}{\pdf@addtoks{[#1]}{Color}}
```

```
% parameter is a string
\define@key{PDF}{Contents}{\pdf@addtoks{(#1)}{Contents}}
% parameter is a integer
\define@key{PDF}{Count}{\pdf@addtoks{#1}{Count}}
% parameter is a array
\define@key{PDF}{CropBox}{\pdf@addtoks{[#1]}{CropBox}}
% parameter is a string
\define@key{PDF}{DOSFile}{\pdf@addtoks{(#1)}{DOSFile}}
% parameter is a string or file
\define@key{PDF}{DataSource}{\pdf@addtoks{(#1)}{DataSource}}
% parameter is a destination
\define@key{PDF}{Dest}{\ifx\\#1\\\else\pdf@addtoks{/#1}{Dest}\fi}
% parameter is a string
\define@key{PDF}{Dir}{\pdf@addtoks{(#1)}{Dir}}
% parameter is a string
\define@key{PDF}{File}{\pdf@addtoks{(#1)}{File}}
% parameter is a int
\define@key{PDF}{Flags}{\pdf@addtoks{#1}{Flags}}
% parameter is a array
\define@key{PDF}{ID}{\pdf@addtoks{[#1]}{ID}}
% parameter is a string
\define@key{PDF}{MacFile}{\pdf@addtoks{(#1)}{MacFile}}
% parameter is a string
\define@key{PDF}{ModDate}{\pdf@addtoks{(#1)}{ModDate}}
% parameter is a string
\define@key{PDF}{Op}{\pdf@addtoks{(#1)}{Op}}
% parameter is a Boolean
\define@key{PDF}{Open}{\pdf@addtoks{#1}{Open}}
% parameter is a integer or name
\define@key{PDF}{Page}{\pdf@addtoks{#1}{Page}}
% parameter is a name
\define@key{PDF}{PageMode}{\pdf@addtoks{#1}{PageMode}}
% parameter is a string
\define@key{PDF}{Params}{\pdf@addtoks{(#1)}{Params}}
% parameter is a array
\define@key{PDF}{Rect}{\pdf@addtoks{[#1]}{Rect}}
% parameter is a integer
\define@key{PDF}{SrcPg}{\pdf@addtoks{#1}{SrcPg}}
% parameter is a name
⟨pdfmark⟩ \define@key{PDF}{Subtype}{\pdf@addtoks{#1}{Subtype}}
⟨dvipdf⟩ \define@key{PDF}{Subtype}{\pdf@addtoks{#1}{}}
% parameter is a string
\define@key{PDF}{Title}{\pdf@addtoks{(#1)}{Title}}
% parameter is a string
\define@key{PDF}{Unix}{\pdf@addtoks{(#1)}{Unix}}
% parameter is a string
\define@key{PDF}{UnixFile}{\pdf@addtoks{(#1)}{UnixFile}}
% parameter is an array
\define@key{PDF}{View}{\pdf@addtoks{[#1]}{View}}
% parameter is a string
```

```
\define@key{PDF}{WinFile}{\pdf@addtoks{(#1)}{WinFile}}
```
These are the keys used in the DOCINFO section.
```
\define@key{PDF}{Author}{\pdf@addtoks{(#1)}{Author}}
\define@key{PDF}{CreationDate}{\pdf@addtoks{(#1)}{CreationDate}}
\define@key{PDF}{Creator}{\pdf@addtoks{(#1)}{Creator}}
\define@key{PDF}{Producer}{\pdf@addtoks{(#1)}{Producer}}
\define@key{PDF}{Subject}{\pdf@addtoks{(#1)}{Subject}}
\define@key{PDF}{Keywords}{\pdf@addtoks{(#1)}{Keywords}}
\define@key{PDF}{ModDate}{\pdf@addtoks{(#1)}{ModDate}}
\define@key{PDF}{Base}{\pdf@addtoks{(#1)}{Base}}
\define@key{PDF}{URI}{\pdf@addtoks{#1}{URI}}
\def\Acrobatmenu#1#2{%
 \pdfmark[#2]{linktype=menu,pdfmark=/ANN,
    Action=<< /Subtype /Named /N /#1 >>,Subtype=/Link}}
```
And now for some useful examples:
```
\def\PDFNextPage{\@ifnextchar[{\PDFNextPage@}{\PDFNextPage@[]}}
\def\PDFNextPage@[#1]#2{%
 \pdfmark[#2]{#1,Border=\@pdfborder,Color=.2 .1 .5,
  pdfmark=/ANN,Subtype=/Link,Page=/Next}}
\def\PDFPreviousPage{\@ifnextchar[{\PDFPreviousPage@}{\PDFPreviousPage@[]}}
\def\PDFPreviousPage@[#1]#2{%
 \pdfmark[#2]{#1,Border=\@pdfborder,Color=.4 .4 .1,
  pdfmark=/ANN,Subtype=/Link,Page=/Prev}}
\def\PDFOpen#1{%
  \pdfmark{#1,pdfmark=/DOCVIEW}%
}
```
This is not as simple as it looks; if we make the argument of this macro eg `\pageref{foo}` and expect it to expand to '3', we need a special version of `\pageref` which does *not* produce '3 '.... David Carlisle looked at this bit and provided the solution, as ever!
```
\def\PDFPage{\@ifnextchar[{\PDFPage@}{\PDFPage@[]}}
\def\PDFPage@[#1]#2#3{%
 \let\pageref\simple@pageref
 \pdfmark[#3]{#1,Page=#2,Border=\@pdfborder,
   Color=\@pagebordercolor,pdfmark=/ANN,Subtype=/Link}}
\def\simple@pageref#1{%
  \expandafter\ifx\csname r@#1\endcsname\relax
   0%
  \else
    \expandafter\expandafter\expandafter
          \@secondoffive\csname r@#1\endcsname
  \fi}
```
This will only work if you use Distiller 2.1 or higher.
```
\def\hyper@linkurl#1#2{%
 \bgroup\hyper@chars
⟨pdfmark⟩ \pdfmark[#1]{pdfmark=/ANN,linktype=url,Border=\@pdfborder,Color=\@urlborder
⟨dvipdf⟩ \pdfmark[#1]{pdfmark=/LNK,linktype=url,Border=\@pdfborder,Color=\@urlborderco
⟨pdfmark⟩ Action={<< /Subtype /URI /URI (#2) >>},Subtype=/Link}%
⟨dvipdf⟩ Action=URI /URI (#2)}%
```

```
 \egroup
}
\def\hyper@linkfile#1#2#3{%
 \bgroup
⟨pdfmark⟩ \pdfmark[#1]{pdfmark=/ANN,Subtype=/Link,
⟨pdfmark⟩ Border=\@pdfborder,linktype=file,Color=\@filebordercolor,Action=/GoToR,File
⟨dvipdf⟩ \pdfmark[#1]{pdfmark=/LNK,linktype=file,Border=\@pdfborder, Color=\@filebord
⟨dvipdf⟩ Action=/GoToR,File=#2,Dest=#3}%
 \egroup
}
⟨/pdfmark | dvipdf⟩
⟨/dvipdf | pdfmark | pdftex⟩
⟨*outlines⟩
```

3 Bookmarks in the PDF file

This was originally developed by Yannis Haralambous (it was the separate `repere.sty`); it needed the `repere` or `makebook.pl` post-processor to work properly. Now redundant, as it is done all in LaTeX macros.

To write out the current section title, and its rationalized number, we have to intercept the `\@sect` command, which is rather dangerous. But how else to see the information we need?

```
\let\H@old@sect\@sect
\def\@sect#1#2#3#4#5#6[#7]#8{%
 \H@old@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]{#8}%
⟨dvipdf⟩ \literalps@out{/BOOK <<}%%
```

If the sectioning internal commands are abused and beaten, what are we to do? Answer, nothing if the first parameter is empty.

```
 \ifx\\#1\\\else
  \ifnum#2>\c@secnumdepth\else
     \edef\@thishlabel{\csname theH#1\endcsname}%
⟨dvipdf⟩\literalps@out{/BOOK /title (#7) \space /level #1.\@thishlabel\space >> }%%
⟨!dvipdf⟩\@writetorep{#7}{#1.\@thishlabel}{#2}\fi
 \fi
}
```

Unfortunately, that only works if the section headings use the standard `\@startsection` macros. Chapters and parts typically don't. This makes it almost impossible to get 100% right, so we just intercept the code in the standard styles.

```
\let\H@old@part\@part
\def\@part[#1]#2{%
⟨dvipdf⟩\literalps@out{/BOOK << }%%
⟨dvipdf⟩ \literalps@out{/BOOK /title (#1) /level part.\theHpart\space >> } %%
⟨!dvipdf⟩ \@writetorep{#1}{part.\theHpart}{-1}%
 \H@old@part[{#1}]{#2}%
}
\let\H@old@chapter\@chapter
\def\@chapter[#1]#2{%
⟨dvipdf⟩\literalps@out{/BOOK << }%
```

```
 \H@old@chapter[{#1}]{#2}%
⟨dvipdf⟩  \literalps@out{/BOOK /title (#1) \space /level chapter.\theHchapter \space >>
⟨!dvipdf⟩   \@writetorep{#1}{chapter.\theHchapter}{0}%
  }

\expandafter\def\csname Parent-2\endcsname{}
\expandafter\def\csname Parent-1\endcsname{}
\expandafter\def\csname Parent0\endcsname{}
\expandafter\def\csname Parent1\endcsname{}

\newwrite\@outlinefile

\def\@writetorep#1#2#3{%
\ifx\WriteBookmarks\relax\else
    \@tempcnta#3
    \expandafter\edef\csname Parent#3\endcsname{#2}%
    \advance\@tempcnta by -1
    \protected@write\@outlinefile%
       {\let~\space
        \def\LaTeX{LaTeX}%
        \def\eTeX{e-TeX}%
        \def\TeX{TeX}%
        \let\label\@gobble
        \let\index\@gobble
        \let\glossary\@gobble}%
       {%
  \protect\BOOKMARK{#2}{#1}{\csname Parent\the\@tempcnta\endcsname}}%
\fi
}
```

Tobias Oetiker rightly points out that we need a way to force a bookmark entry. So we introduce `\pdfbookmark`, with two parameters, the title, and a symbolic name. By default this is at level 1, but we can reset that with the optional first argument.

```
\def\pdfbookmark{\@ifnextchar[{\pdf@bookmark}{\pdf@bookmark[0]}}
\def\pdf@bookmark[#1]#2#3{%
\ifx\WriteBookmarks\relax\else
 \@writetorep{#2}{#3.#1}{#1}%
 \hyper@anchorstart{#3.#1}\hyper@anchorend
\fi
}
```

The macros for calculating structure of outlines are derived from those by Petr Olsak used in the texinfopdf macros.

```
\AtBeginDocument{\ReadBookmarks}
\def\ReadBookmarks{%
  \def\BOOKMARK ##1##2##3{\calc@bm@number{##3}}%
  \InputIfFileExists{\jobname.out}{
}{}%
  \def\BOOKMARK ##1##2##3{%
    \def\@tempx{##2}%
⟨*pdftex⟩
     \pdfoutline goto
```

```
⟨pdftexold⟩  name{##1!}%
⟨!pdftexold⟩  name{##1}%
     count\@bookmarkopenstatus\check@bm@number{##1}{%
   \expandafter\strip@prefix\meaning\@tempx}%
⟨/pdftex⟩
⟨*pdfmark⟩
 \pdfmark{pdfmark=/OUT,Count=\@bookmarkopenstatus\check@bm@number{##1},
     Dest=###1,Title=\expandafter\strip@prefix\meaning\@tempx}%
⟨/pdfmark⟩
  }%
  {\def\WriteBookmarks{0}%
   \escapechar\m@ne\InputIfFileExists{\jobname.out}{}{}%
   \ifx\WriteBookmarks\relax\else
    \immediate\openout\@outlinefile=\jobname.out
   \fi
}
\def\check@bm@number#1{\expandafter \ifx\csname#1\endcsname \relax 0%
    \else \csname#1\endcsname \fi}
\def\calc@bm@number#1{\@tempcnta=\check@bm@number{#1}\relax
    \advance\@tempcnta by1
    \expandafter\xdef\csname#1\endcsname{\the\@tempcnta}}
⟨/outlines⟩
⟨*repere⟩
\newwrite\@reperefile
\immediate\openout\@reperefile=\jobname.rep
\def\@writetorep#1#2#3{%
    \protected@write\@reperefile
       {\def\TeX{TeX}%
        \def\LaTeX{LaTeX}%
        \let\label\@gobble
        \let\index\@gobble
        \let\glossary\@gobble}%
       {(#2) <#1>}%
}
⟨/repere⟩
⟨*pdfmark⟩
```

We have to allow for `\baselineskip` having an optional stretch and shrink (you meet this in slide packages, for instance), so we need to strip off the junk. David Carlisle, of course, wrote this bit of code.

```
\begingroup
  \catcode‘P=12
  \catcode‘T=12
  \lowercase{\endgroup
\gdef\rem@ptetc#1.#2PT#3!{#1\ifnum#2>\z@.#2\fi}}
\def\strip@pt@and@otherjunk#1{\expandafter\rem@ptetc\the#1!}
⟨/pdfmark⟩
```

3.1 Device dependent setup

Unfortunately, some parts of the `pdfmark` PostScript code depend on vagaries of the dvi driver. We isolate here all the problems.

3.1.1 Rokicki's dvips

dvips thinks in 10ths of a big point, its coordinate space is resolution dependent, and its y axis starts at the top of the page. Other drivers can and will be different!

The work is done in `SDict`, because we add in some header definitions in a moment.

```
⟨∗dvips⟩
\def\@pdfview{XYZ pdf@hoff pdf@voff null}
\def\literalps@out#1{\special{ps:SDict begin #1 end}}%
```

The calculation of upper left y is done without raising the point in TEX, by simply adding on the current `\baselineskip` to the current y. This is usually too much, so we remove a notional 2 points.

We have to see what the current baselineskip is, and convert it to the dvips coordinate system.

```
\def\pdf@setheight{\literalps@out{%
  \strip@pt@and@otherjunk\baselineskip
  \space 2 sub dup
  /HyperBasePt exch def
  PDFToDvips /HyperBaseDvips exch def
  }%
}
\def\PSHyperAnchorStart{\literalps@out{HyperStart }}
\def\PSHyperAnchorEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd HyperAutoVoff }%
}
\def\PSHyperLinkStart{\literalps@out{HyperStart }}
\def\PSHyperLinkEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd}%
}
\def\PSHyperRectStart{\literalps@out{HyperStart }}
\def\PSHyperRectEnd{\literalps@out{HyperEnd HyperVoff }}
```

Do not ask what this rubbish is. Trying to make `pdfmark` get the right destination for views.

```
\AtBeginDvi{\special{!
```

Unless I am going mad, this *appears* to be the relationship between the default coordinate system (PDF), and dvips;

```
/DvipsToPDF { .01383701 div Resolution div } def
/PDFToDvips { .01383701 mul Resolution mul } def
```

the latter's coordinates are resolution dependent, but what that .01383701 is, who knows? well, almost everyone except me, I expect... And yes, Maarten Gelderman <mgelderman@econ.vu.nl> points out that its 1/72.27 (the number of points to an inch, big points to inch is 1/72). This also suggests that the code would be more understandable (and exact) if 0.013 div would be replaced by 72.27 mul, so here we go. If this isn't right, I'll revert it.

```
/DvipsToPDF { 72.27 mul Resolution div } def
/PDFToDvips { 72.27 div Resolution mul } def
```

The rectangle around the links starts off *exactly* the size of the box; we will to make it slightly bigger, 1 point on all sides.

```
/HyperBorder  { 1 PDFToDvips } def
% the distance from the top of the page to the current point, in
% PDF coordinates
/HyperVoff {
   currentpoint exch pop vsize 72 sub
   exch DvipsToPDF sub /pdf@voff exch def
 } def
```

the distance from the top of the page to a point `\baselineskip` above the current point in PDF coordinates

```
/HyperAutoVoff {
   currentpoint exch pop
    vsize 72 sub exch DvipsToPDF
    HyperBasePt sub % baseline skip
    sub /pdf@voff exch def
 } def
```

the x and y coordinates of the current point, in PDF coordinates

```
/HyperStart {
   currentpoint
    HyperBorder add /pdf@lly exch def
    dup DvipsToPDF  /pdf@hoff exch def
    HyperBorder sub /pdf@llx exch def
} def
```

the x and y coordinates of the current point, minus the baselineskip

```
/HyperAutoEnd  {
   currentpoint
    HyperBaseDvips sub /pdf@ury exch def
    /pdf@urx exch def
} def
 /HyperEnd  {
   currentpoint
    HyperBorder sub /pdf@ury exch def
    HyperBorder add /pdf@urx exch def
 } def
 systemdict
 /pdfmark known not
 {userdict /pdfmark systemdict /cleartomark get put} if
```

```
}}
\AtBeginDocument{%
  \ifhy@colorlinks
   \def\@pdfborder{0 0 0}%
  \fi
  \special{papersize=\special@paper}%
}
\def\@pdfborder{0 0 12}
⟨/dvips⟩
```

3.1.2 Textures

At the suggestion of Jacques Distler (distler@golem.ph.utexas.edu), try to derive a suitable driver for Textures. This is a copy of dvips, with some guesses about Textures behaviour.

```
⟨*textures⟩
\def\literalps@out#1{\special{rawpostscript #1}}%
\def\pdf@setheight{\literalps@out{%
  \strip@pt@and@otherjunk\baselineskip
  \space 2 sub
  PDFToDvips /HyperBase exch def
  }%
}
\def\@pdfview{XYZ pdf@hoff pdf@voff null}
%
```

These are called at the start and end of unboxed links; their job is to leave available PS variables called `pdf@llx pdf@lly pdf@urx pdf@ury`, which are the coordinates of the bounding rectangle of the link, and `pdf@hoff pdf@voff` which are the PDF page offsets. The Rect pair are called at the LL and UR corners of a box known to TEX.

```
\def\PSHyperAnchorStart{\literalps@out{HyperStart }}
\def\PSHyperAnchorEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd HyperAutoVoff }%
}
\def\PSHyperLinkStart{\literalps@out{HyperStart }}
\def\PSHyperLinkEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd}%
}
\def\PSHyperRectStart{\literalps@out{HyperStart }}
\def\PSHyperRectEnd{\literalps@out{HyperEnd HyperVoff }}
\special{prepostscript
```

Textures lives in normal points, I think. So conversion from one coordinate system to another involves doing nothing.

```
/vsize { \hy@pageheight } def
/DvipsToPDF { } def
/PDFToDvips { } def
```

```
/HyperBorder  { 1 PDFToDvips } def
/HyperVoff {
  currentpoint exch pop vsize 72 sub
  exch DvipsToPDF sub /pdf@voff exch def
 } def
/HyperAutoVoff {
  currentpoint exch pop
   vsize 72 sub exch DvipsToPDF
   HyperBase sub % baseline skip
   sub /pdf@voff exch def
 } def
/HyperStart {
  currentpoint
   HyperBorder add /pdf@lly exch def
   dup DvipsToPDF  /pdf@hoff exch def
   HyperBorder sub /pdf@llx exch def
} def
/HyperAutoEnd  {
  currentpoint
   HyperBase sub /pdf@ury exch def
   /pdf@urx exch def
} def
 /HyperEnd  {
  currentpoint
   HyperBorder sub /pdf@ury exch def
   HyperBorder add /pdf@urx exch def
 } def
 systemdict
 /pdfmark known not
 {userdict /pdfmark systemdict /cleartomark get put} if
}
\AtBeginDocument{%
  \ifhy@colorlinks
   \def\@pdfborder{0 0 0}%
  \fi
}
\def\@pdfborder{0 0 1}
⟨/textures⟩
```

3.1.3 dvipsone

```
⟨*dvipsone⟩
\def\literalps@out#1{\special{ps:#1}}%
\def\pdf@setheight{\literalps@out{%
  \strip@pt@and@otherjunk\baselineskip
  \space 2 sub
  PDFToDvips /HyperBase exch def
  }%
}
\def\@pdfview{%
```

```
XYZ gsave revscl currentpoint grestore 72 add
   exch pop null exch null}
%
```

These are called at the start and end of unboxed links; their job is to leave available PS variables called `pdf@llx pdf@lly pdf@urx pdf@ury`, which are the coordinates of the bounding rectangle of the link, and `pdf@hoff pdf@voff` which are the PDF page offsets. These latter are currently not used in the dvipsone setup. The Rect pair are called at the LL and UR corners of a box known to TEX.

```
\def\PSHyperAnchorStart{\literalps@out{HyperStart }}
\def\PSHyperAnchorEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd HyperAutoVoff }%
}
\def\PSHyperLinkStart{\literalps@out{HyperStart }}
\def\PSHyperLinkEnd{%
  \pdf@setheight
  \literalps@out{HyperAutoEnd}%
}
\def\PSHyperRectStart{\literalps@out{HyperStart }}
\def\PSHyperRectEnd{\literalps@out{HyperEnd HyperVoff }}
\special{headertext=
```

dvipsone lives in scaled points; does this mean 65536 or 65781?

```
/DvipsToPDF { 65781 div  } def
/PDFToDvips { 65781 mul } def
/HyperBorder  { 1 PDFToDvips } def
/HyperVoff {
   currentpoint exch pop DvipsToPDF /pdf@voff exch def
 } def
/HyperAutoVoff {
   currentpoint exch pop
    HyperBase sub % baseline skip
    DvipsToPDF /pdf@voff exch def
 } def
/HyperStart {
   currentpoint
    HyperBorder add /pdf@lly exch def
    dup   DvipsToPDF /pdf@hoff exch def
    HyperBorder sub /pdf@llx exch def
} def
/HyperAutoEnd  {
   currentpoint
    HyperBase sub /pdf@ury exch def
    /pdf@urx exch def
} def
/HyperEnd  {
   currentpoint
    HyperBorder sub /pdf@ury exch def
    HyperBorder add /pdf@urx exch def
```

```
 } def
 systemdict
 /pdfmark known not
 {userdict /pdfmark systemdict /cleartomark get put} if
}
\AtBeginDocument{%
  \ifhy@colorlinks
   \def\@pdfborder{0 0 0}%
  \fi
}
\def\@pdfborder{0 0 65781}
⟨/dvipsone⟩
⟨*dvipdf⟩
\def\literalps@out#1{\special{pdf: #1}}%
\AtBeginDocument{%
  \ifhy@colorlinks
   \def\@pdfborder{0 0 0}%
  \fi
}
\def\@pdfborder{0 0 1}
⟨/dvipdf⟩
⟨*tex4ht⟩
\RequirePackage[htex4ht]{tex4ht}
\def\PDF@SetupDoc{%
 \ifx\@baseurl\@empty\else
  \special{t4ht=<base href="\@baseurl">}%
 \fi
}
\def\hyper@anchor#1{%
   {\let\protect=\string\special{t4ht=<A name=\hyper@quote #1\hyper@quote>}}%
   \hy@activeanchortrue
   \bgroup\colorlink{\@anchorcolor}\anchor@spot\egroup
   \special{t4ht=</A>}%
   \hy@activeanchorfalse
}
\def\hyper@anchorstart#1{%
 {\hyper@chars\special{t4ht=<A name=\hyper@quote#1\hyper@quote>}}%
  \hy@activeanchortrue
}
\def\hyper@anchorend{%
  \special{t4ht=</A>}%
  \hy@activeanchorfalse
}
\def\@urltype{url}
\def\hyper@linkstart#1#2{%
  \colorlink{\csname @#1color\endcsname}%
  \def\@tempa{#1}\ifx\@tempa\@urltype
    \special{t4ht=<A href=\hyper@quote#2\hyper@quote>}%
  \else
   {\hyper@chars\special{t4ht=<A href=\hyper@quote\##2\hyper@quote>}}%
```

```
  \fi
}
\def\hyper@linkend{%
  \special{t4ht=</A>}%
  \hyper@resetcolor
}
\def\hyper@linkfile#1#2#3{%
 \hyper@linkurl{#1}{file:#2\ifx\\#3\\\else\##3\fi}%
}
\def\hyper@linkurl#1#2{%
 \ifhy@raiselinks
  \setbox\@tempboxa=\hbox{#1}%
  \@linkdim\dp\@tempboxa
  \leavevmode\lower\@linkdim\hbox{%
      {\hyper@chars\special{t4ht=<A href=\hyper@quote#2\hyper@quote>}}%
   }%
  {\colorlink{\@urlcolor}#1}%
  \@linkdim\ht\@tempboxa
  \advance\@linkdim by -6.5\p@
  \raise\@linkdim\hbox{\special{t4ht=</A>}}%
 \else
  {\hyper@chars
   \special{t4ht=<A href=\hyper@quote#2\hyper@quote>}%
   \colorlink{\@urlcolor}#1}%
   \special{t4ht=</A>}%
 \fi
}
\def\hyper@link#1#2#3{%
  \hyper@linkurl{#3}{\##2}%
}
\def\hyper@image#1#2{%
  {\hyper@chars
   \special{t4ht=<img src=\hyper@quote#1\hyper@quote>}}}
⟨/tex4ht⟩
⟨*tex4htcfg⟩
\Preamble{html}
   \begin{document}
\EndPreamble
\def\TeX{TeX}
\def\OMEGA{Omega}
\def\LaTeX{LaTeX}
\def\LaTeXe{LaTeX2e}
\def\eTeX{e-\TeX}
\def\MF{Metafont}
⟨/tex4htcfg⟩
```