

# pathsuris.sty: Paths and URIs for $\text{\TeX}$ \*

Jinbo Zhang, Michael Kohlhase, Dennis Müller  
FAU Erlangen-Nürnberg

November 3, 2020

## Abstract

This package provides macros to deal with paths and base URIs for  $\text{\TeX}$ . In particular, it offers a path canonicalizer, which is used in package `modules`, in order to support modules specified with relative path.

## Contents

|          |                                 |          |
|----------|---------------------------------|----------|
| <b>1</b> | <b>User Interface</b>           | <b>2</b> |
| 1.1      | Base URIs . . . . .             | 2        |
| 1.2      | Using Absolute Paths . . . . .  | 2        |
| 1.3      | Path Canonicalization . . . . . | 2        |
| 1.4      | URIs . . . . .                  | 3        |
| <b>2</b> | <b>The Implementation</b>       | <b>4</b> |
| 2.1      | Base URIs . . . . .             | 4        |
| 2.2      | Using Absolute Paths . . . . .  | 4        |
| 2.3      | Path Canonicalization . . . . . | 4        |
| 2.4      | URIs . . . . .                  | 6        |

---

\*Version v2.1 (last revised 2020/09/30)

# 1 User Interface

## 1.1 Base URIs

`\baseURI` `\baseURI`<sup>1</sup>

## 1.2 Using Absolute Paths

Finally, the separation of documents into multiple modules often profits from a symbolic management of file paths. To simplify this, the `modules` package supplies the `\defpath` macro: `\defpath[⟨baseURI⟩]{⟨cname⟩}{⟨path⟩}` defines a command, so that `\⟨cname⟩{⟨name⟩}` expands to `⟨path⟩/⟨name⟩`. So we could have used

```
\defpath{OPaths}{../other}
\importmodule[load=\OPahts{bar}]{bar}
```

instead of the second line in Example ???. The variant `\OPaths` has the big advantage that we can get around the fact that `TeX/LATeX` does not set the current directory in `\input`, so that we can use systematically deployed `\defpath`-defined path macros to make modules relocatable by defining the path macros locally. The optional parameter `⟨baseURI⟩` is for the `LATeXML` transformation, which (if `⟨baseURI⟩` is specified) resolves `⟨path⟩` to an absolute URI according to [BFM05, section 5.2].

## 1.3 Path Canonicalization

By calling `\@cpath{⟨path⟩}`, the canonicalized path will be stored in `\@CanPath`.  
`\cpath` To print a canonicalized path, simply use `\cpath{⟨path⟩}`. Here is a set of examples with their canonizalized paths for testing.

| path          | canonicalized path | expected      |
|---------------|--------------------|---------------|
| aaa           | aaa                | aaa           |
| ../../aaa     | ../../aaa          | ../../aaa     |
| aaa/bbb       | aaa/bbb            | aaa/bbb       |
| aaa/..        |                    |               |
| ../../aaa/bbb | ../../aaa/bbb      | ../../aaa/bbb |
| ../aaa/..bbb  | ../bbb             | ../bbb        |
| ../aaa/bbb    | ../aaa/bbb         | ../aaa/bbb    |
| aaa/bbb/..ddd | aaa/ddd            | aaa/ddd       |
| aaa/bbb/./ddd | aaa/bbb/ddd        | aaa/bbb/ddd   |
| ./            |                    |               |
| aaa/bbb/../.. |                    |               |

<sup>1</sup>EDNOTE: document it

## 1.4 URIs

By calling `\seturi[\meta{macroname}]{\langle path \rangle}`, the URI will be split into its components `\macronamescheme`, `\macronameauthority`, `\macronamepath`, `\macronamequery` and `\macronamefragment`, and the resolved URI itself is stored in `\macronameuri`, as in the following example. If the optional `macroname` is not provided, the default name is `pathsuris@curruri@`.

In order to differentiate between empty and missing components, a missing component will be equal to `\makeuri@empty`, whose *expansion* is `\relax`.

`\seturi[myuri]{http://this.isatest/foo/bar/?query#fragment}`

|         |                              |  |
|---------|------------------------------|--|
|         | macro                        | value                                      |
| yields: | <code>\myuriuri</code>       | http://this.isatest/foo/bar?query#fragment |
|         | <code>\myurischeme</code>    | http                                       |
|         | <code>\myuriauthority</code> | this.isatest                               |
|         | <code>\myuripath</code>      | foo/bar                                    |
|         | <code>\myuriquery</code>     | query                                      |
|         | <code>\myurifragment</code>  | fragment                                   |

`\makeuri{\meta{scheme}}{\meta{authority}}{\meta{path}}{\meta{query}}{\meta{fragment}}` constructs a URI from its individual components. The (expanded and resolved) individual components will be stored in `\makeuri@scheme`, `\makeuri@authority`, etc.; the resolved URI will be stored in `\makeuri@uri`.

`\asuri{\meta{macroname}}{\meta{uri}}`, similarly to `setpath`, defines a new macro `\macroname[\meta{newmacroname}]{\meta{command}}` that allows manipulating `uri` in various ways. `\asuri` calls `\seturi[\meta{macroname}]{\meta{uri}}`, so the individual components and full `uri` (as string) are subsequently stored in `\macronamescheme`, `\macronameauthority`, etc.

If an optional new macro name is given in `\macroname`, then the result of the modification is stored in that new macro, as if defined via `\asuri`; otherwise, the macro is modified “in place”.

- `\macroname{drop query}` drops the query component.
- `\macroname{drop fragment}` drops the fragment component.
- `\macroname{/other/path}` drops query and fragment, appends `other/path` to the path and resolves the URI.
- `\macroname{?newquery}` drops the fragment and either declares `newquery` as a new query component, or appends `?newquery` to the existing query component, if it is not `\makeuri@empty`. Note, that this behaviour diverges from the official URI specification, but it conforms to MMT URI’s, which use `?` as separator between `DPaths`, modules names and declaration names.
- `\macroname{#newfragment}` analogously to `{?newquery}`.

## 2 The Implementation

```

1 <*package>
2 \RequirePackage{stex-base}
3 \RequirePackage{xstring}
4 \RequirePackage{etoolbox}

```

### 2.1 Base URIs

`\baseURI` On the L<sup>A</sup>T<sub>E</sub>X side we do nothing (for the moment).

```

5 \newcommand\baseURI[2] [] {}

```

### 2.2 Using Absolute Paths

`\defpath` `\defpath[optional argument]{macro name}{base path}` defines a new macro which can take another path to form one integrated path. For example, `\MathHub` in every `localpaths.tex` is defined as:

```

\defpath{MathHub}{/path/to/localmh/MathHub}

```

then we can use `\MathHub` to form other paths, for example,

```

\MathHub{source/smgglom/sets}

```

will generate `/path/to/localmh/MathHub/source/smgglom/sets`.

```

6 \newrobustcmd\defpath[3] [] {%
7   \expandafter\newcommand\csname #2\endcsname[1] {#3/##1}%
8 }%

```

### 2.3 Path Canonicalization

We define two macros for changing the category codes of common characters in URIs, in particular `#`.

```

9 \def\pathsuris@setcatcodes{%
10   \edef\pathsuris@oldcatcode@hash{\the\catcode'\#}%
11   \catcode'\#=12\relax%
12   \edef\pathsuris@oldcatcode@slash{\the\catcode'\/%}
13   \catcode'\/=12\relax%
14   \edef\pathsuris@oldcatcode@colon{\the\catcode'\:%}
15   \catcode'\:=12\relax%
16   \edef\pathsuris@oldcatcode@qm{\the\catcode'\?}%
17   \catcode'\?=12\relax%
18 }
19 \def\pathsuris@resetcatcodes{%
20   \catcode'\#\pathsuris@oldcatcode@hash\relax%
21   \catcode'\/>\pathsuris@oldcatcode@slash\relax%
22   \catcode'\:\pathsuris@oldcatcode@colon\relax%
23   \catcode'\?\pathsuris@oldcatcode@qm\relax%
24 }

```

We define some macros for later comparison.

```

25 \def\@ToTop{..}
26 \def\@Slash{/}
27 \def\@Colon{:}
28 \def\@QuestionMark{?}
29 \def\@Dot{.}
30
31 \pathsuris@setcatcodes
32 \def\@Fragment{#}
33 \pathsuris@resetcatcodes

```

Implement \@cpath.

\@cpath

```

34 \def\@cpath#1{%
35   \edef\pathsuris@cpath@temp{#1}%
36   \def\@CanPath{}%
37   \IfBeginWith\pathsuris@cpath@temp\@Slash{%
38     \@cpath@loop%
39     \edef\@CanPath{\@Slash\@CanPath}%
40   }{%
41     \IfBeginWith\pathsuris@cpath@temp\@Dot\@Slash{%
42       \StrGobbleLeft\pathsuris@cpath@temp2[\pathsuris@cpath@temp]%
43       \@cpath@loop%
44     }{%
45       \ifx\pathsuris@cpath@temp\@Dot\else%
46       \@cpath@loop\fi%
47     }%
48   }%
49   \IfEndWith\@CanPath\@Slash{%
50     \ifx\@CanPath\@Slash\else%
51       \StrGobbleRight\@CanPath1[\@CanPath]%
52     \fi%
53   }{}%
54 }
55
56 \def\@cpath@loop{%
57   \IfSubStr\pathsuris@cpath@temp\@Slash{%
58     \StrCut\pathsuris@cpath@temp\@Slash\pathsuris@cpath@temp@a\pathsuris@cpath@temp%
59     \ifx\pathsuris@cpath@temp@a\@ToTop%
60       \ifx\@CanPath\empty%
61         \edef\@CanPath{\@ToTop}%
62       \else%
63         \edef\@CanPath{\@CanPath\@Slash\@ToTop}%
64       \fi%
65       \@cpath@loop%
66     \else%
67     \ifx\pathsuris@cpath@temp@a\@Dot%
68       \@cpath@loop%
69     \else%

```

```

70      \IfBeginWith\pathsuris@cpath@temp\@ToTop{%
71          \StrBehind{\pathsuris@cpath@temp}{\@ToTop}[\pathsuris@cpath@temp]%
72          \IfBeginWith\pathsuris@cpath@temp\@Slash{%
73              \edef\pathsuris@cpath@temp{\@CanPath\pathsuris@cpath@temp}%
74          }{%
75              \ifx\@CanPath\@empty\else%
76                  \edef\pathsuris@cpath@temp{\@CanPath\@Slash\pathsuris@cpath@temp}
77              \fi%
78          }%
79          \def\@CanPath{}%
80          \@cpath@loop%
81      }{%
82          \ifx\@CanPath\@empty%
83              \edef\@CanPath{\pathsuris@cpath@temp@a}%
84          \else%
85              \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp@a}%
86          \fi%
87          \@cpath@loop
88      }%
89      \fi\fi%
90  }{
91      \ifx\@CanPath\@empty%
92          \edef\@CanPath{\pathsuris@cpath@temp}%
93      \else%
94          \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp}%
95      \fi%
96  }%
97 }

```

Implement `\cpath` to print the canonicalized path.

`\cpath`

```

98 \newcommand\cpath[1]{%
99     \@cpath{#1}%
100     \@CanPath%
101 }

```

## 2.4 URIs

Various macros for dealing with URIs. To deal with empty URI components (scheme, authority, etc.), we use `\relax` to signify a non-existent component as opposed to an empty one.

```

102 \def\makeuri@setempty#1{\def#1{\relax}}
103 \def\makeuri@empty{\relax}
104 \def\makeuri@test#1{%
105     \ifx#1\makeuri@empty\else#1\fi%
106 }

```

`\makeuri` `\makeuri` constructs a URI from scheme, authority, path, query and fragment separately.

```

107 \def\makeuri@uri{}
108 \def\makeuri#1#2#3#4#5{
109     \edef\makeuri@scheme{#1}
110     \edef\makeuri@authority{#2}
111     \edef\makeuri@path{#3}
112     \ifx\makeuri@path\makeuri@empty\else
113         \@cpath{#3}
114         \edef\makeuri@path{\@CanPath}
115     \fi
116     \edef\makeuri@query{#4}
117     \edef\makeuri@fragment{#5}
118     \ifx\makeuri@scheme\makeuri@empty\else
119         \edef\makeuri@scheme{\makeuri@scheme\@Colon}
120     \fi
121     \ifx\makeuri@authority\makeuri@empty\else
122         \edef\makeuri@authority{\@Slash\@Slash\makeuri@authority}
123         \ifx\makeuri@path\makeuri@empty\else
124             \IfBeginWith\makeuri@path\@Slash{{
125                 \edef\makeuri@path{\@Slash\makeuri@path}
126             }}
127         \fi
128     \fi
129     \ifx\makeuri@query\makeuri@empty\else
130         \edef\makeuri@query{\@QuestionMark\makeuri@query}
131     \fi
132     \ifx\makeuri@fragment\makeuri@empty\else
133         \edef\makeuri@fragment{\@Fragment\makeuri@fragment}
134     \fi
135     \edef\makeuri@uri{%
136         \makeuri@test\makeuri@scheme%
137         \makeuri@test\makeuri@authority%
138         \makeuri@test\makeuri@path%
139         \makeuri@test\makeuri@query%
140         \makeuri@test\makeuri@fragment%
141     }
142 }

```

\seturi@

```

143 \newif\if@pathsuris@done@
144 \def\seturi@[#1]#2{%
145     \@pathsuris@done@false%
146     \def\pathsuris@prefix@temp{#1}
147     \edef\pathsuris@curruri{#2}%
148     \edef\pathsuris@curruri{\expandafter\detokenize\expandafter{\pathsuris@curruri}}
149     \let\pathsuris@temp\pathsuris@curruri%
150     \makeuri@setempty\pathsuris@curruri@scheme%
151     \makeuri@setempty\pathsuris@curruri@authority%
152     \makeuri@setempty\pathsuris@curruri@path%
153     \makeuri@setempty\pathsuris@curruri@query%
154     \makeuri@setempty\pathsuris@curruri@fragment%

```

```

155 % scheme
156 \IfSubStr{\pathsuris@temp}{\@Colon}{%
157 % TODO check for valid scheme
158 \StrBefore{\pathsuris@temp}{\@Colon}[\pathsuris@curruri@scheme]%
159 \StrBehind{\pathsuris@temp}{\@Colon}[\pathsuris@temp]%
160 }{}%
161 % authority
162 \IfBeginWith{\pathsuris@temp}{\@Slash\@Slash}{%
163 \StrBehind{\pathsuris@temp}{\@Slash\@Slash}[\pathsuris@temp]%
164 \IfSubStr{\pathsuris@temp}{\@Slash}{%
165 \StrBefore{\pathsuris@temp}{\@Slash}[\pathsuris@curruri@authority]%
166 \StrBehind{\pathsuris@temp}{\@Slash}[\pathsuris@temp]%
167 % TODO userinfo,host,port
168 }{%
169 \IfSubStr\pathsuris@temp\@QuestionMark{
170 \StrBefore{\pathsuris@temp}{\@QuestionMark}[\pathsuris@curruri@authority]%
171 \StrBehind{\pathsuris@temp}{\@QuestionMark}[\pathsuris@temp]%
172 \edef\pathsuris@temp{\@QuestionMark\pathsuris@temp}%
173 }{
174 \IfSubStr\pathsuris@temp\@Fragment{
175 \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@authority]%
176 \StrBehind{\pathsuris@temp}{\@Fragment}[\pathsuris@temp]%
177 \edef\pathsuris@temp{\@Fragment\pathsuris@temp}%
178 }{
179 \edef\pathsuris@curruri@authority{\pathsuris@temp}%
180 \@pathsuris@done@true%
181 }
182 }
183 }%
184 }{}%
185 % path, query, fragment
186 \if@pathsuris@done@else%
187 \IfSubStr{\pathsuris@temp}{\@QuestionMark}{%
188 % path
189 \StrBefore{\pathsuris@temp}{\@QuestionMark}[\pathsuris@curruri@path]%
190 \@cpath\pathsuris@curruri@path%
191 \edef\pathsuris@curruri@path{\@CanPath}%
192 \StrBehind{\pathsuris@temp}{\@QuestionMark}[\pathsuris@temp]%
193 % query, fragment
194 \IfSubStr{\pathsuris@temp}{\@Fragment}{%
195 \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@query]%
196 \StrBehind{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@fragment]%
197 }{%
198 \edef\pathsuris@curruri@query{\pathsuris@temp}%
199 }%
200 }{%
201 % path, fragment
202 \IfSubStr{\pathsuris@temp}{\@Fragment}{%
203 \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@path]%
204 \@cpath\pathsuris@curruri@path%

```



```

205             \edef\pathsuris@curruri@path{\@CanPath}%
206             \StrBehind{\pathsuris@temp}{\@Fragment}{\pathsuris@curruri@fragment}%
207         }{%
208             \edef\pathsuris@curruri@path{\pathsuris@temp}%
209         }%
210     }%
211 \fi%
212 \makeuri\pathsuris@curruri@scheme\pathsuris@curruri@author\pathsuris@curruri@path\pathsuris@curruri@query\pathsuris@curruri@fragment\pathsuris@curruri@uri
213 \let\pathsuris@curruri@uri\makeuri@uri
214 %drop trailing slash of path
215 %\IfEndWith{\pathsuris@curruri@path}{\@Slash}{%
216 %    \StrGobbleRight{\pathsuris@curruri@path}{1}{\pathsuris@curruri@path}
217 %}%}%
218 %
219 %\edef\pathsuris@curruri@path{\cpath{\pathsuris@curruri@path}}%
220 \ifx\pathsuris@prefix@temp\empty\else%
221     \expandafter\let\csname \pathsuris@prefix@temp scheme\endcsname\pathsuris@curruri@scheme%
222     \expandafter\let\csname \pathsuris@prefix@temp authority\endcsname\pathsuris@curruri@author%
223     \expandafter\let\csname \pathsuris@prefix@temp path\endcsname\pathsuris@curruri@path%
224     \expandafter\let\csname \pathsuris@prefix@temp query\endcsname\pathsuris@curruri@query%
225     \expandafter\let\csname \pathsuris@prefix@temp fragment\endcsname\pathsuris@curruri@fragment%
226     \expandafter\let\csname \pathsuris@prefix@temp uri\endcsname\pathsuris@curruri@uri%
227 \fi%
228 }

\seturi
229 \newrobustcmd\seturi[1][{}]{%
230     \pathsuris@setcatcodes%
231     \expandafter\pathsuris@resetcatcodes\seturi@[#1]%
232 }

\asuri \asuri{macroname}{uri} generates \macroname[optional new macro name]{action},
that allows for modifying uri in various ways.

233
234 \def\asuri#1{%
235     \pathsuris@setcatcodes%
236     \expandafter\pathsuris@resetcatcodes\asuri@[#1]%
237 }
238
239 \def\@asuri[#1]#2{
240     \@cpath{#2}
241     \expandafter\def\csname #1\endcsname{
242         \expandafter\edef\csname #1uri\endcsname{\@CanPath}
243         \seturi[#1]{\@CanPath}
244         \expandafter\renewcommand\csname #1\endcsname[1][{}]{%
245             \pathsuris@setcatcodes%
246             \@asuri@[##1]{#1}%
247         }%
248 }

```

```

249
250 \protected\def\@asuri@[#1]#2#3{
251     \pathsuris@resetcatcodes
252     \@asuri@[#1]{#2}{#3}
253 }
254
255 \newif\if@asuri@changed@
256 \protected\def\@asuri@[#1]#2#3{
257     \@asuri@changed@false
258     \edef\@asuri@command{#3}
259     \trimstring\@asuri@command
260     \IfBeginWith\@asuri@command{drop}{
261         \StrBehind{\@asuri@command}{drop}[\@asuri@command]
262         \trimstring\@asuri@command
263         \IfStrEq\@asuri@command{query}{
264             \makeuri{\csname #2scheme\endcsname}%
265                 {\csname #2authority\endcsname}%
266                 {\csname #2path\endcsname}%
267                 \makeuri@empty%
268                 {\csname #2fragment\endcsname}%
269             \@asuri@changed@true
270         }{
271             \IfStrEq\@asuri@command{fragment}{
272                 \makeuri{\csname #2scheme\endcsname}%
273                     {\csname #2authority\endcsname}%
274                     {\csname #2path\endcsname}%
275                     {\csname #2query\endcsname}%
276                 \makeuri@empty%
277                 \@asuri@changed@true
278             }{
279                 \IfStrEq\@asuri@command{extension}{
280                     \edef\@asuri@oldpath{\csname #2path\endcsname}
281                     \StrCount\@asuri@oldpath.[\@asuri@lastdot]
282                     \ifnum\@asuri@lastdot>0
283                         \StrBehind[\@asuri@lastdot]\@asuri@oldpath.[\@asuri@extension]
284                         \IfSubStr\@asuri@extension\@Slash{{
285                             \StrBefore[\@asuri@lastdot]\@asuri@oldpath.[\@asuri@oldpath]
286                         }
287                     \fi
288                     \makeuri{\csname #2scheme\endcsname}%
289                         {\csname #2authority\endcsname}%
290                         \@asuri@oldpath%
291                         \makeuri@empty%
292                         \makeuri@empty%
293                     \@asuri@changed@true
294                 }{}}{
295             }{
296                 \IfBeginWith\@asuri@command{\@Slash}{
297                     \@cpath{\csname #2path\endcsname\@asuri@command}
298                     \makeuri{\csname #2scheme\endcsname}%

```

```

299         {\csname #2authority\endcsname}%
300         {\@CanPath}%
301         \makeuri@empty%
302         \makeuri@empty%
303         \@asuri@changed@true
304     }{
305     \IfBeginWith\@asuri@command{\@QuestionMark}{
306         \expandafter\ifx\csname #2query\endcsname\makeuri@empty
307             \StrBehind\@asuri@command\@QuestionMark[\@asuri@command]
308             \edef\@asuri@nquery{\@asuri@command}
309         \else
310             \edef\@asuri@nquery{\csname #2query\endcsname\@asuri@command}
311         \fi
312         \makeuri{\csname #2scheme\endcsname}%
313             {\csname #2authority\endcsname}%
314             {\csname #2path\endcsname}%
315             {\@asuri@nquery}%
316         \makeuri@empty%
317         \@asuri@changed@true
318     }{
319     \IfBeginWith\@asuri@command{\@Fragment}{
320         \expandafter\ifx\csname #2fragment\endcsname\makeuri@empty
321             \StrBehind\@asuri@command\@Fragment[\@asuri@command]
322             \edef\@asuri@nfrag{\@asuri@command}
323         \else
324             \edef\@asuri@nfrag{\csname #2fragment\endcsname\@asuri@command}
325         \fi
326         \makeuri{\csname #2scheme\endcsname}%
327             {\csname #2authority\endcsname}%
328             {\csname #2path\endcsname}%
329             {\csname #2query\endcsname}%
330             {\@asuri@nfrag}%
331         \@asuri@changed@true
332     }{}
333     }}
334     \edef\@asuri@ncs{#1}
335     \if@asuri@changed@
336         \ifx\@asuri@ncs\@empty
337             \asuri{#2}\makeuri@uri
338         \else
339             \asuri\@asuri@ncs\makeuri@uri
340         \fi
341     \fi
342 }
343

```

auxiliary code:

```

344 \def\@Space{ }
345 \def\trimstring#1{
346     \edef\pathsuris@trim@temp{#1}

```

```

347 \IfBeginWith\pathsuris@trim@temp\@Space{
348 \StrGobbleLeft\pathsuris@trim@temp1[#1]
349 \trimstring{#1}
350 }{
351 \IfEndWith\pathsuris@trim@temp\@Space{
352 \StrGobbleRight\pathsuris@trim@temp1[#1]
353 \trimstring{#1}
354 }{
355 \edef#1{\pathsuris@trim@temp}
356 }
357 }
358 }
359
360 % windows paths
361
362 \catcode'\.=0
363 .catcode'\.=12
364 .let.\@BackSlash\
365 .catcode'\.=0
366 \catcode'\.=12
367
368 \newif\if@windowstopath@inpath@
369 \def\windows@to@path#1{
370 \@windowstopath@inpath@false
371 \def\windows@temp{}
372 \edef\windows@path{#1}
373 \ifx\windows@path\@empty\else
374 \expandafter\windows@path@loop\windows@path\windows@path@end
375 \fi
376 \let#1\windows@temp
377 }
378 \def\windows@path@loop#1#2\windows@path@end{
379 \def\windows@temp@b{#2}
380 \ifx\windows@temp@b\@empty
381 \def\windows@continue{}
382 \else
383 \def\windows@continue{\windows@path@loop#2\windows@path@end}
384 \fi
385 \if@windowstopath@inpath@
386 \ifx#1\@BackSlash
387 \edef\windows@temp{\windows@temp\@Slash}
388 \else
389 \edef\windows@temp{\windows@temp#1}
390 \fi
391 \else
392 \ifx#1:
393 \edef\windows@temp{\@Slash\windows@temp:}
394 \@windowstopath@inpath@true
395 \else
396 \edef\windows@temp{\windows@temp#1}

```

```

397         \fi
398     \fi
399     \windows@continue
400 }
401
402 \def\path@to@windows#1{
403     \@windowstopath@inpath@false
404     \def\windows@temp{}
405     \edef\windows@path{#1}
406     \edef\windows@path{\expandafter\@gobble\windows@path}
407     \ifx\windows@path\empty\else
408         \expandafter\path@windows@loop\windows@path\windows@path@end
409     \fi
410     \let#1\windows@temp
411 }
412 \def\path@windows@loop#1#2\windows@path@end{
413     \def\windows@temp@b{#2}
414     \ifx\windows@temp@b\empty
415         \def\windows@continue{}
416     \else
417         \def\windows@continue{\path@windows@loop#2\windows@path@end}
418     \fi
419     \if@windowstopath@inpath@
420         \ifx#1/
421             \edef\windows@temp{\windows@temp\@BackSlash}
422         \else
423             \edef\windows@temp{\windows@temp#1}
424         \fi
425     \else
426         \ifx#1/
427             \edef\windows@temp{\windows@temp:\@BackSlash}
428             \@windowstopath@inpath@true
429         \else
430             \edef\windows@temp{\windows@temp#1}
431         \fi
432     \fi
433     \windows@continue
434 }
435
436 % kpsewhich
437
438 \newif\if@iswindows@\@iswindows@false
439 \IfFileExists{nul:}{\IfFileExists{/dev/null}}{\@iswindows@true}}{}
440
441 \def\kpsewhich#1#2{\begingroup
442     \def\@Space{ }
443     \edef\kpsewhich@cmd{"|kpsewhich #2"}
444     \everyeof{\noexpand}
445     \catcode'\@=12
446     \def\par{}

```

```

447 \edef#1{\@@input\kpsewhich@cmd\@Space}
448 \trimstring#1
449 \global\let#1#1
450 \endgroup}
451
452 % main directory
453
454 \edef\oldpercentcatcode{\the\catcode'\%}
455 \catcode'\%=12
456 \let\percent%
457 \catcode'\%=\oldpercentcatcode
458
459 \edef\pwd@cmd{\if@iswindows@ -expand-var \percent CD\percent\else -var-value PWD\fi}
460 \kpsewhich\stex@maindir\pwd@cmd
461 \if@iswindows@\windows@to@path\stex@maindir\fi
462
463 \def\path@filename#1#2{
464   \edef\filename@oldpath{#1}
465   \StrCount\filename@oldpath\@Slash[\filename@lastslash]
466   \ifnum\filename@lastslash>0
467     \StrBehind[\filename@lastslash]\filename@oldpath\@Slash[\filename@oldpath]
468     \csedef{#2}{\filename@oldpath}
469   \else
470     \csedef{#2}{\filename@oldpath}
471   \fi
472 }
473
474 \def\path@dropeextension#1#2{
475   \path@filename{#1}{dropeextension@temp}
476   \StrCount\dropeextension@temp\@Dot[\dropeextension@lastdot]
477   \ifnum\dropeextension@lastdot>0
478     \StrBehind[\dropeextension@lastdot]\dropeextension@temp\@Dot[\dropeextension@ext]
479     \StrLen\dropeextension@ext[\dropeextension@lastdot]
480     \StrGobbleRight{#1}{\the\numexpr\dropeextension@lastdot+1\@Space}[\dropeextension@temp]
481     \trimstring\dropeextension@temp
482     \csedef{#2}{\dropeextension@temp}
483   \else
484     \csedef{#2}{#1}
485   \fi
486 }
487
488 </package>

```

## Change History

v1.0

General: First Version with  
Documentation . . . . . 1

v1.1

General: adding `\baseURI` from  
`omdoc.sty` and `\defpath` from  
`modules.sty` . . . . . 1

## References

- [BFM05] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Internet Engineering Task Force (IETF), 2005. URL: <http://www.ietf.org/rfc/rfc3986.txt>.