

pathsuris.sty: Paths and URIs for \TeX *

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

October 4, 2020

Abstract

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

Contents

1	User Interface	2
1.1	Base URIs	2
1.2	Using Absolute Paths	2
1.3	Path Canonicalization	2
1.4	URI splitting	2
2	The Implementation	4
2.1	Base URIs	4
2.2	Using Absolute Paths	4
2.3	Path Canonicalization	4
2.4	URI splitting	6

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

1 User Interface

1.1 Base URIs

`\baseURI` `\baseURI`¹

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 L^AT_EXML 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`. 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/../..	../	../

1.4 URIs

By calling `\seturi[\meta{macroname}]{⟨path⟩}`, the URI will be split into its components `\macronamescheme`, `\macronameauthority`, `\macronamepath`,

¹EdNOTE: document it

`\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}
```

yields:

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

`\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^AT_EX 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\@ToHere{.}
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     \@cpath@loop%
42   }%
43   \IfEndWith\@CanPath\@Slash{%
44     \ifx\@CanPath\@Slash\else%
45       \StrGobbleRight\@CanPath1[\@CanPath]%
46     \fi%
47   }{}%
48 }
49
50 \def\@cpath@loop{%
51   \IfSubStr\pathsuris@cpath@temp\@Slash{%
52     \StrCut\pathsuris@cpath@temp\@Slash\pathsuris@cpath@temp@a\pathsuris@cpath@temp%
53     \ifx\pathsuris@cpath@temp@a\@ToTop%
54       \ifx\@CanPath\empty%
55         \edef\@CanPath{\@ToTop}%
56       \else%
57         \edef\@CanPath{\@CanPath\@Slash\@ToTop}%
58       \fi%
59     \@cpath@loop%
60   \else%
61     \IfBeginWith\pathsuris@cpath@temp\@ToTop{%
62       \StrBehind\pathsuris@cpath@temp{\@ToTop}[\pathsuris@cpath@temp]%
63       \IfBeginWith\pathsuris@cpath@temp\@Slash{%
64         \edef\pathsuris@cpath@temp{\@CanPath\pathsuris@cpath@temp}%
65       }{%
66         \ifx\@CanPath\empty\else%
67           \edef\pathsuris@cpath@temp{\@CanPath\@Slash\pathsuris@cpath@temp}
68         \fi%
69       }%

```

```

70         \def\@CanPath{}%
71         \@cpath@loop%
72     }{%
73         \ifx\@CanPath\@empty%
74             \edef\@CanPath{\pathsuris@cpath@temp@a}%
75         \else%
76             \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp@a}%
77         \fi%
78         \@cpath@loop
79     }%
80 \fi%
81 }{
82     \ifx\@CanPath\@empty%
83         \edef\@CanPath{\pathsuris@cpath@temp}%
84     \else%
85         \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp}%
86     \fi%
87 }%
88 }

```

Implement `\cpath` to print the canonicalized path.

`\cpath`

```

89 \newcommand\cpath[1]{%
90     \@cpath{#1}%
91     \@CanPath%
92 }

```

2.4 URIs

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

```

93 \def\makeuri@setempty#1{\def#1{\relax}}
94 \def\makeuri@empty{\relax}
95 \def\makeuri@test#1{%
96     \ifx#1\makeuri@empty\else#1\fi%
97 }

```

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

```

98 \def\makeuri@uri{}
99 \def\makeuri#1#2#3#4#5{
100     \edef\makeuri@scheme{#1}
101     \edef\makeuri@authority{#2}
102     \edef\makeuri@path{#3}
103     \ifx\makeuri@path\makeuri@empty\else
104         \@cpath{#3}
105     \edef\makeuri@path{\@CanPath}

```

```

106 \fi
107 \edef\makeuri@query{#4}
108 \edef\makeuri@fragment{#5}
109 \ifx\makeuri@scheme\makeuri@empty\else
110 \edef\makeuri@scheme{\makeuri@scheme\@Colon}
111 \fi
112 \ifx\makeuri@authority\makeuri@empty\else
113 \edef\makeuri@authority{\@Slash\@Slash\makeuri@authority}
114 \ifx\makeuri@path\makeuri@empty\else
115 \IfBeginWith\makeuri@path\@Slash{}\{
116 \edef\makeuri@path{\@Slash\makeuri@path}
117 }
118 \fi
119 \fi
120 \ifx\makeuri@query\makeuri@empty\else
121 \edef\makeuri@query{\@QuestionMark\makeuri@query}
122 \fi
123 \ifx\makeuri@fragment\makeuri@empty\else
124 \edef\makeuri@fragment{\@Fragment\makeuri@fragment}
125 \fi
126 \edef\makeuri@uri{%
127 \makeuri@test\makeuri@scheme%
128 \makeuri@test\makeuri@authority%
129 \makeuri@test\makeuri@path%
130 \makeuri@test\makeuri@query%
131 \makeuri@test\makeuri@fragment%
132 }
133 }

\seturi@
134 \newif\if@pathsuris@done@
135 \def\seturi@[#1]#2{%
136 \@pathsuris@done@false%
137 \def\pathsuris@prefix@temp{#1}
138 \edef\pathsuris@curruri{#2}%
139 \let\pathsuris@temp\pathsuris@curruri%
140 \makeuri@setempty\pathsuris@curruri@scheme%
141 \makeuri@setempty\pathsuris@curruri@authority%
142 \makeuri@setempty\pathsuris@curruri@path%
143 \makeuri@setempty\pathsuris@curruri@query%
144 \makeuri@setempty\pathsuris@curruri@fragment%
145 % scheme
146 \IfSubStr{\pathsuris@temp}{\@Colon}{%
147 % TODO check for valid scheme
148 \StrBefore{\pathsuris@temp}{\@Colon}[\pathsuris@curruri@scheme]%
149 \StrBehind{\pathsuris@temp}{\@Colon}[\pathsuris@temp]%
150 }{}%
151 % authority
152 \IfBeginWith{\pathsuris@temp}{\@Slash\@Slash}{%
153 \StrBehind{\pathsuris@temp}{\@Slash\@Slash}[\pathsuris@temp]%

```

```

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

```



```

204 %drop trailing slash of path
205 %\IfEndWith{\pathsuris@curruri@path}{\@Slash}{%
206 % \StrGobbleRight{\pathsuris@curruri@path}{1}{\pathsuris@curruri@path]
207 %}{}%
208 %
209 %\edef\pathsuris@curruri@path{\cpath{\pathsuris@curruri@path}}%
210 \ifx\pathsuris@prefix@temp\empty\else%
211 \expandafter\let\csname \pathsuris@prefix@temp scheme\endcsname\pathsuris@curruri@schem
212 \expandafter\let\csname \pathsuris@prefix@temp authority\endcsname\pathsuris@curruri@au
213 \expandafter\let\csname \pathsuris@prefix@temp path\endcsname\pathsuris@curruri@path%
214 \expandafter\let\csname \pathsuris@prefix@temp query\endcsname\pathsuris@curruri@query%
215 \expandafter\let\csname \pathsuris@prefix@temp fragment\endcsname\pathsuris@curruri@fra
216 \expandafter\let\csname \pathsuris@prefix@temp uri\endcsname\pathsuris@curruri@uri%
217 \fi%
218 }

\seturi
219 \newrobustcmd\seturi[1][]{%
220 \pathsuris@setcatcodes%
221 \expandafter\pathsuris@resetcatcodes\seturi@[#1]%
222 }

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

223
224 \def\asuri#1{%
225 \pathsuris@setcatcodes%
226 \expandafter\pathsuris@resetcatcodes\@asuri[#1]%
227 }
228
229 \def\@asuri[#1]#2{
230 \cpath{#2}
231 \expandafter\def\csname #1\endcsname{
232 \expandafter\edef\csname #1uri\endcsname{\@CanPath}
233 \seturi[#1]{\@CanPath}
234 \expandafter\renewcommand\csname #1\endcsname[1][]{%
235 \pathsuris@setcatcodes%
236 \@asuri@[##1]{#1}%
237 }%
238 }
239
240 \protected\def\@asuri@[#1]#2#3{
241 \pathsuris@resetcatcodes
242 \@asuri[#1]{#2}{#3}
243 }
244
245 \newif\if@asuri@changed@
246 \protected\def\@asuri@[#1]#2#3{
247 \@asuri@changed@false

```

```

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

```

```

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

```

auxiliary code:

```

334 \def\@Space{ }
335 \def\trimstring#1{
336     \edef\pathsuris@trim@temp{#1}
337     \IfBeginWith\pathsuris@trim@temp\@Space{
338         \StrGobbleLeft\pathsuris@trim@temp1[#1]
339         \trimstring{#1}
340     }{
341         \IfEndWith\pathsuris@trim@temp\@Space{
342             \StrGobbleRight\pathsuris@trim@temp1[#1]
343             \trimstring{#1}
344         }{
345             \edef#1{\pathsuris@trim@temp}

```

```

346     }
347   }
348 }
349
350 % windows paths
351
352 \catcode'\.=0
353 .catcode'\.=12
354 .let.\@BackSlash\
355 .catcode'\.=0
356 \catcode'\.=12
357
358 \newif\if@windowstopath@inpath@
359 \def\windows@to@path#1{
360   \@windowstopath@inpath@false
361   \def\windows@temp{ }
362   \edef\windows@path{#1}
363   \ifx\windows@path\@empty\else
364     \expandafter\windows@path@loop\windows@path\windows@path@end
365   \fi
366   \let#1\windows@temp
367 }
368 \def\windows@path@loop#1#2\windows@path@end{
369   \def\windows@temp@b{#2}
370   \ifx\windows@temp@b\@empty
371     \def\windows@continue{ }
372   \else
373     \def\windows@continue{\windows@path@loop#2\windows@path@end}
374   \fi
375   \if@windowstopath@inpath@
376     \ifx#1\@BackSlash
377       \edef\windows@temp{\windows@temp\@Slash}
378     \else
379       \edef\windows@temp{\windows@temp#1}
380     \fi
381   \else
382     \ifx#1:
383       \edef\windows@temp{\@Slash\windows@temp}
384       \@windowstopath@inpath@true
385     \else
386       \edef\windows@temp{\windows@temp#1}
387     \fi
388   \fi
389   \windows@continue
390 }
391
392 \def\path@to@windows#1{
393   \@windowstopath@inpath@false
394   \def\windows@temp{ }
395   \edef\windows@path{#1}

```

```

396 \edef\windows@path{\expandafter\@gobble\windows@path}
397 \ifx\windows@path\@empty\else
398 \expandafter\path@windows@loop\windows@path\windows@path@end
399 \fi
400 \let#1\windows@temp
401 }
402 \def\path@windows@loop#1#2\windows@path@end{
403 \def\windows@temp@b{#2}
404 \ifx\windows@temp@b\@empty
405 \def\windows@continue{}
406 \else
407 \def\windows@continue{\path@windows@loop#2\windows@path@end}
408 \fi
409 \if@windowstopath@inpath@
410 \ifx#1/
411 \edef\windows@temp{\windows@temp\@BackSlash}
412 \else
413 \edef\windows@temp{\windows@temp#1}
414 \fi
415 \else
416 \ifx#1/
417 \edef\windows@temp{\windows@temp:\@BackSlash}
418 \@windowstopath@inpath@true
419 \else
420 \edef\windows@temp{\windows@temp#1}
421 \fi
422 \fi
423 \windows@continue
424 }
425
426 \</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>.