anyt

`pathsuris.sty`: Paths and URIs for sTeX[*]

Jinbo Zhang, Michael Kohlhase
Jacobs University, Bremen

November 23, 2017

Abstract

This package provides macros to deal with paths and base URIs for sTeX. 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

2 The Implementation **3**
2.1 Base URIs 3
2.2 Using Absolute Paths 3
2.3 Path Canonicalization 3

[*] Version v1.1 (last revised 2016/02/18)

1 User Interface

1.1 Base URIs

`\baseURI` `\baseURI`[1]

EdN: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 `\defpath` supplies the `\defpath` macro: `\defpath[`⟨*baseURI*⟩`]{`⟨*cname*⟩`}{`⟨*path*⟩`}` defines a command, so that `\`⟨*csname*⟩`{`⟨*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 [**BerFieMas:05**].

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] EdNote: document it

2 The Implementation

```
⟨*package⟩
\RequirePackage{xstring}
\RequirePackage{forloop}
\RequirePackage{calc}
\RequirePackage{etoolbox}
```

2.1 Base URIs

\baseURI On the LaTeX side we do nothing (for the moment).

```
\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 formal 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/smglom/sets}
```

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

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

2.3 Path Canonicalization

We first create some counters. `AddrNum` will count the number of sections in the input path, `iLoop` will be used as the loop iterator, `iName` will be used for generating names such as `Addri`, `Addrii`, `RealAddrNum` will count the number of sections in the canonicalized path, `Cutable` will count the number of sections besides `..`.

```
\newcounter{AddrNum}
\newcounter{iLoop}
\newcounter{iName}
\newcounter{RealAddrNum}
\newcounter{Cutable}
```

We define two macros for later comparison.

```
\def\@ToTop{..}
\def\@Slash{/}
```

Then we split the input path.

```
\def\@MultiAddrs#1/#2\@nil{%
  \def\CurArg{#1}%
  \def\NextArg{#2}%
  \ifx\@empty\CurArg% for the first one
  \else%
    \stepcounter{AddrNum}%
    \expandafter\edef\csname Addr\roman{AddrNum}\endcsname{#1}% storing
  \fi%
  \ifx\@empty\NextArg% for the last one
    \let\next\@gobble%
  \fi%
  \next#2\@nil% recursion
}%
```

Implement \@cpath.

```
\def\@cpath#1{%
  \let\next\@MultiAddrs%
  \setcounter{AddrNum}{0}%
  \setcounter{iLoop}{0}%
  \setcounter{iName}{0}%
  \setcounter{RealAddrNum}{0}%
  \setcounter{Cutable}{0}%
  \def\@CurrPath{}%
  \def\@CanPath{}%
  \def\@TempPath{}%
  \def\@Rubbish{}%
  \expandafter\next#1/\@nil% recursion starts
  \forloop{iLoop}{0}{\value{iLoop} < \value{AddrNum}}{%
    \stepcounter{iName}%
    \edef\@CurrPath{\csname Addr\roman{iName}\endcsname}%
    \ifx\@CurrPath\@ToTop%
  \ifnum\value{Cutable} = 0%
        \edef\@CanPath{\@CanPath\csname Addr\roman{iName}\endcsname/}%
        \stepcounter{RealAddrNum}%
      \else%
        % cut the last part, and add a slash at the end
        \StrCut[\value{RealAddrNum}]{/\@CanPath}{/}\@TempPath\@Rubbish%
        \StrCut[1]{\@TempPath/}{/}\@Rubbish\@CanPath%
        \addtocounter{RealAddrNum}{-1}%
        \addtocounter{Cutable}{-1}%
      \fi%
    \else%
      \edef\@CanPath{\@CanPath\csname Addr\roman{iName}\endcsname/}%
      \stepcounter{RealAddrNum}%
      \stepcounter{Cutable}%
    \fi%
  }%
  \StrCut[\value{RealAddrNum}]{\@CanPath}{/}\@CanPath\@Rubbish% cut last /
}%
```

Implement \cpath to print the canonicalized path.

```
\newcommand\cpath[1]{% print canonical path
\@cpath{#1}%
\@CanPath%
}%
⟨/package⟩
```