Relative inverse path calculation

Will Robertson

2008/07/31 v0.2

inversepath is a simple package to calculate inverse relative paths. For example, when writing to an auxiliary file in a subdirectory (or a series of nested subdirectories), it can be useful to know how to get back to the original file.

If the absolute path of the original file is specified, this package can also calculate the relative path of a file in parent or sibling directories.

`\inversepath{`⟨*path*⟩`}` — prints the inverse of ⟨*path*⟩.
`\inversepath*{`⟨*path*⟩`}` — calculates the inverse of ⟨*path*⟩.

`\absolutepath{`⟨*abs. path*⟩`}` — specifies the absolute path for calculating parent/sibling relative paths.

Regular usage:

../../../
four.tex
one/two/three/

```
\inversepath*{one/two/three/four.tex}\par
\ip@inversepath\par
\ip@lastelement\par
\ip@directpath
```

Expands to ⟨*empty*⟩ if the relative path is the same directory:

[]
one.tex
[]

```
[\inversepath*{one.tex}]\par
\ip@inversepath\par
\ip@lastelement\par
[\ip@directpath]
```

For ‘back-relative’ paths, the absolute path needs to be specified:

../../there/everywhere/
three.tex
../../one/two/

```
\absolutepath{/xyz/here/there/everywhere/}

\inversepath*{../../one/two/three.tex}\par
\ip@inversepath\par
\ip@lastelement\par
\ip@directpath
```

File I

inversepath implementation

This is the package.

```
\ProvidesPackage{inversepath}
  [2008/07/31 v0.2 Inverse relative paths]
```

`\inversepath` `#1` : Path to invert

```
\newcommand\inversepath{%
  \@ifstar{\inversepath@star}{\inversepath@nostar}}
\newcommand\inversepath@star[1]{%
```

`\ip@jobpath` is preserved to restore after truncation for back-relative paths.

```
  \let\ip@origjobpath\ip@jobpath
  \let\ip@directpath\@empty
  \let\ip@inversepath\@empty
  \ip@strippath#1/\@nil/%
  \let\ip@jobpath\ip@origjobpath}
\newcommand\inversepath@nostar[1]{%
  \inversepath@star{#1}%
  \let\ip@jobpath\ip@origjobpath}
```

`\absolutepath` `#1` : Absolute path used for calculating parent/sibling relative paths.

```
% macro to define the absolute path of where we are:
\newcommand\absolutepath[1]{\def\ip@jobpath{#1}}
```

For `\ifx` comparisons for relative back-paths:

```
\def\ip@literaldotdot{..}
```

`\ip@strippath` This is the macro that does all the work. It takes input like `a/b/c/...x/y/z/\@nil/` and expands to `\ip@inversepath`, the inverse path of `\ip@directpath` (`a/b/.../y/`).

```
\def\ip@strippath#1/#2/{%
  \ifx\@nil#2\relax
```

If input is `z/\@nil/` then we've reached the end:

```
    \def\ip@lastelement{#1}%
  \else
```

If we're in the middle of the slash-separated list; build up `\ip@directpath`:

```
    \edef\ip@directpath{\ip@directpath#1/}
    \def\@tempa{#1}%
    \ifx\@tempa\ip@literaldotdot
    \unless\ifdefined\ip@jobpath
      \PackageError{inversepath}
        {No absolute path specified}
        {You must declare the file path of the main
         file with \protect\absolutepath{} to be able to
         resolve back-relative paths}%
    \fi
```

If the path is a back-relative path, things are more complex. to get the inverse of `../`, we need the absolute file path. this requires using `\ip@strippath` on `\ip@jobpath` itself, so save out our current definitions of `\ip@directpath`/`\ip@inversepath` and (re-)initialise them:

```
    \let\ip@olddirectpath\ip@directpath
    \let\ip@oldinversepath\ip@inversepath
    \let\ip@directpath\@empty
    \let\ip@inversepath\@empty
```

`\ip@strippath`on `\ip@jobpath` gives us the topmost directory in `\ip@lastelement`:

```
    \expandafter\ip@strippath\ip@jobpath\@nil/
    \let\@tempa\ip@lastelement
```

`\ip@jobpath`is now truncated so `\iplastelement` in the next iteration is one folder up the hierarchy.

```
    \let\ip@jobpath\ip@directpath
```

Now we restore everything to how it was: (this would be better with grouping, but I don't want to use `\global`)

```
    \let\ip@directpath\ip@olddirectpath
    \let\ip@inversepath\ip@oldinversepath
```

Build up the inverse path:

```
    \ifx\@tempa\@empty
      \PackageError{inversepath}
        {Absolute path too shallow to resolve
          such a deep relative path}
        {You're trying to go back more directories than you have!}
    \fi
    \edef\ip@inversepath{\@tempa/\ip@inversepath}%
  \else
```

If the path is a simple relative path, then build up the inverse path by prepending `../`:

```
    \edef\ip@inversepath{../\ip@inversepath}%
  \fi
```

Iterate:

```
  \def\@tempa{\ip@strippath#2/}%
  \expandafter\@tempa
\fi}
```