Permute sections into definition before use order

/Reference manual/Auxiliary tools

1. SYNOPSIS

```
section [sections] filename
```

2. DESCRIPTION

The `section` tool performs some preliminary processing of the mark-up of a Z specification. Its input is mark-up spread over several files, and its output is a single sequence of sections sorted into definition before use order.

In more detail... It interprets the given `filename` as identifying a file containing mark-up of some Z sections. It checks that all ancestors of those sections exist, looking if necessary for separate files with corresponding names. (Toolkits are typically in separate files.) In each file, it treats any paragraphs that are not preceded by a section header as being in a section whose name is that of the file and whose parent is `standard_toolkit`. If none of those paragraphs are formal ones, it suffixes the name of the section with `Informal`, so that the next section can share the name of the file. It ensures that there are no cycles in the parents

relation. If these checks are passed, it writes to its standard output the mark-up of those sections. The sections are written in an order so that every section is written before other sections of which it is a parent.

Within each section, all of its mark-up directives are moved to immediately follow the section header. For each section, a separate file called `filename.sectionname.mud` is generated containing copies of all the mark-up directives that are in scope in that section. (The separate `.mud` file is used when processing the responses keyed into dialogue boxes, whereas the processing of the mark-up of a section is still based on the mark-up directives within the section and its ancestors.)

Any sections whose names are amongst those given as `sections` on the command-line are omitted from the output. (This is used to omit previously checked sections by cadiz's `-l` option.) In their place, directives are written that refer to the corresponding `.mud` files.

Filename and line number directives are added to the output to assist in locating errors back to the original source.

3. ENVIRONMENT

`ARCH` specifies the processor architecture on which `section` is to be executed. Unless you are accessing `section` remotely in a heterogeneous computing network, the default is likely to be appropriate. Recognised values include irix6, solaris2 and i486.

`CADIZ` names the directory in which CADiZ is installed. In particular, executables are kept in `$CADIZ/mip/bin`. If that directory is in your `$PATH`, then it is not necessary to set the `CADIZ` variable.

`CADIZPATH` is a search path of directories in which `section` will look for files. `section` looks in the directory of toolkits `$CADIZ/mip/kits/$MARKUP` first, then any directories given by `$CADIZPATH`, and lastly in the current directory.

`MARKUP` should be either `latex` or `groff`. This determines which mark-up `section` assumes is used in the files. It defaults to `groff`.

4. EXIT STATUS

The exit status of `section` is

0 if no errors are detected in the entire specification,

1 if some errors are detected in the specification,

2 if `section` detects something amiss with itself.

5. SEE ALSO

Z specifications of earlier versions of this `section` tool may be found in the examples part of this manual and in section 8 of [**?**].

IT 28-Jan-2002