Module ‘ask’

Julio Manuel Fernández-Díaz

Profesor Titular, Department of Physics, University of Oviedo

February 2010

Table of contents

Module ask

Basic information

`ask` is a system that provides help for modules.

It searches eight types of information:

`basic` `list` `usage` `more` `seealso` `example` `version` `notes`

Execute `ask"/ask^usage"` and `ask"/ask^more"` for more explanation.

Apart from this, documentation in `html` format can be generated (see `ask.doc`).

List of functions

about base doc

Usage of the module

`ask(search)`

@params:

1. `search` is a string in the form `"what^kind"` (a caret separates two parts of the argument).

@returns: nothing.

@effects: it prints in `io.stderr`.

`what` is what are looking for; if the first character in it is `/` an *absolute* path is searched; if not the string defined in `ask.base` is used as a basis;

`kind` is the type of information we want, that can be:

```
"basic"   | "list"    | "usage"   | "more"    | "seealso"
"b"       | "l"       | "u"       | "m"       | "s"

"example" | "version" | "notes"   | "all"
"e"       | "v"       | "n"       | "a"
```

(as we see, we can use only the first letter). If `all` is requested then all information (basic, list, usage, seealso, example, version, notes) *if exists* is given. If no kind is provided (in this case the caret is optional) `"basic"` is assumed.

`ask(nil)` and `ask.about(nil)` are equivalent to `ask"/ask^basic"`, that is, help about this helping system is given.

More specific information

The search of information is controlled by two strings: `what` and `kind`. Both are typed separated by a caret, `"^"`.

The first, `what`, is what we are searching. Normally this is a sequence `name1[.name2[.name3 ...] ]` in which `name1` is a module name, and `name2`, `name3`, ... indicate functions in the module or tables with functions in the module.

If `what` begins with a slash, `/`, then an absolute path is searched. Otherwise the help system adds as a prefix the string stored in a local variable assigned with the function `ask.base`. This improves the interactivity because the user is not enforced to always type the complete path of help.

The second, `kind`, is the type of information we want:

- `basic`: the purpose of the module or function inside a module,
- `list`: the listing of functions in the module,
- `usage`: the use of a function, describing the arguments and returns,
- `more`: more specific information,
- `seealso`: some related information,
- `example`: an example of use,
- `version`: information about version and author,
- `notes`: other information, usually license one,
- `all`: show all the previous information.

For activate the help system the user (interactively) or some module should invoke

```
require"ask"
```

Note: the present help system manages a module variable `_H`. This means that `_H` cannot be used for other purposes in the module.

Examples

Some examples (with equivalences):

We assume the *first* invoking after `require"ask"` (by the user or by some module):

```
ask""                     --><--    ask"/ask^basic"
ask"^a"                   --><--    ask"/ask^all"
ask"/somemodule.fun^u"    --><--    ask"/somemodule.fun^usage"

ask.base"somemodule"      -- changes basis for searching

ask"^l"                   --><--    ask"/somemodule^list"
ask"fun^u"                --><--    ask"/somemodule.fun^usage"
ask"/ask^m"               --><--    ask"/ask^more"
```

Description of functions

ask.about

Basic information

Main function for on line help (synonym of `ask`)

Usage of function

`ask.about(search)`

`ask(search)`

@params:

1. `search` is a string in the form `"what^kind"` (a caret separates two parts of the argument).

@returns: nothing.

@effects: it prints in `io.stderr`.

`what` is what are looking for; if the first character in it is `/` an *absolute* path is searched; if not the string defined through `ask.base` is used as a basis;

`kind` is the type of information we want, that can be:

```
"basic"   | "list"    | "usage"   | "more"   | "seealso"
"b"       | "l"       | "u"       | "m"      | "s"

"example" | "version" | "notes"   | "all"
"e"       | "v"       | "n"       | "a"
```

(as we can see we can use only the first letter). If `all` is requested then all information, (basic, list, usage, seealso, example, version, notes) *if exists* is given. If no kind is provided `"basic"` is assumed.

Spaces typed by the user in the set `what^kind` are deleted before the search of the help. Also, multiple `"."` are collapsed.

`ask(nil)` and `ask.about(nil)` are equivalent to `ask"/ask^basic"`, that is, help about this helping system is given.

ask.base

Basic information

Establishes a *basis prefix* for help searching

Usage of function

`ask.base(basis)`

@params:

1. `basis`, string.

@returns: nothing.

This function changes the `basis` (a string) to add as a prefix in the desired information path when this does not begin with a slash, `/`.

When providind `basis` the slashdot `"/"` is not required.

Initially basis has the value `"ask"` but the loading of a module sets the current `basis` to the module name.

When calling `ask.base""` the current basis is displayed.

Calling `ask.base(nil)` establishes `"ask"` as basis.

ask.doc

Basic information

Create `html` documentation for a module

Usage of function

`ask.doc(modulename, filename)`

@params:

1. `modulename`: string (optional) is the name of the module of which we want the documentation. If not provided then the basis is used.
2. `filename`: string (optional) is the name of the output file, in `html` format. If not given the name of the module is used. If one of the extensions `".html"` or `".htm"` (in lowercase) is not provided then automatically `".html"` is added to the filename.

@returns: nothing

@effects: it creates a file.

For generating the documentation module markdown.lua from Niklas Frykholm must be accessible. (Note: the version in luaforge.net is obsolete.)

More specific information

A CSS file called `default.css`, which is possible to customize, is used. This file is embeded in the `html` output file. If not provided the system uses an internal style.

The resulting `html` file can be converted, v.g., to PS with html2ps, from Jan Kärrman. After that `ps2pdf` can be used to convert it to PDF format.

Version

by Julio M. Fernández-Díaz, Dept. of Physics, University of Oviedo, Spain, Version 0.1, February 2010

julio a t uniovi d o t es

Notes