Package ‘argparser’

April 4, 2024

Type Package

Title Command-Line Argument Parser

Version 0.7.2

Author David J. H. Shih

Maintainer David J. H. Shih <djh.shih@gmail.com>

Description Cross-platform command-line argument parser written purely in R
with no external dependencies. It is useful with the Rscript
front-end and facilitates turning an R script into an executable script.

URL https://bitbucket.org/djhshih/argparser

BugReports https://bitbucket.org/djhshih/argparser/issues

Depends methods

Suggests testthat (>= 3.0.0)

License GPL (>= 3)

RoxygenNote 7.2.3

Encoding UTF-8

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-04-04 05:03:14 UTC

R topics documented:

add.argument 2
add_argument 3
arg.parser 5
arg_parser 5
include 6
parse.args 6
parse_args 7
print.arg.parser 8
show_arg_labels 8
spaces 9

Index **10**

`add.argument` *Add an argument to a parser.*

Description

This function is deprecated. Use `add_argument` instead.

Usage

```
add.argument(
  parser,
  arg,
  help,
  default = NULL,
  type = NULL,
  flag = NULL,
  short = NULL
)
```

Arguments

`parser`	an `arg.parser` object
`arg`	argument name (use no prefix for positional arguments, `--` or `-` prefix for optional arguments or flags)
`help`	help description for the argument
`default`	default value for the argument [default: NA]
`type`	variable type of the argument (which can be inferred from `default`), assumed to be `character` otherwise
`flag`	whether argument is a flag (and does not consume a value) [default: FALSE]
`short`	short-form for flags and positional arguments; short-forms can be assigned automatically based on the first character of the argument name, unless a conflict arises with an existing short-form; to avoid conflicts, add the argument as early as possible

Value

an `arg.parser` object with the argument added `add_argument` *Add an argument to a parser.*

Description

This function adds an argument to an `arg.parser` object and returns the modified object.

Usage

```
add_argument(
  parser,
  arg,
  help,
  default = NULL,
  type = NULL,
  nargs = NULL,
  flag = NULL,
  short = NULL
)
```

Arguments

`parser`	an `arg.parser` object
`arg`	argument name (use no prefix for positional arguments, `--` or `-` prefix for optional arguments or flags)
`help`	help description for the argument
`default`	default value for the argument [default: NA]
`type`	variable type of the argument (which can be inferred from `default`); assumed to be `character` otherwise. See details for more information.
`nargs`	number of argument values (which can be inferred from `default`); set to `Inf` for an indefinite number; an optional argument with an indefinite number of values may need to be followed by another optional argument or flag (e.g. `--`) to separate the indefinite optional argument from possible position arguments
`flag`	whether argument is a flag (and does not consume a value) [default: FALSE]; during argument parsing, a flag argument is FALSE by default if it is not set
`short`	short-form for flags and positional arguments; short-forms can be assigned automatically based on the first character of the argument name, unless a conflict arises with an existing short-form; to avoid conflicts, add the argument as early as possible

Details

This function supports multiple arguments in a vector. To ensure that the argument variable type is set correctly, either specify `type` directly or supply `default` argument values as a list. Custom types are supported by defining a new class and a S4 method for `coerce`, see the examples section.

Value

an `arg.parser` object with the argument added

Note

Dashes `-` that occur in the stem of the argument names (e.g. –argument-name) will be converted to underscores `_` (e.g. argument_name) in the name of the corresponding variable.

Examples

```
p <- arg_parser("A text file modifying program")

# Add a positional argument
p <- add_argument(p, "input", help="input file")

# Add an optional argument
p <- add_argument(p, "--output", help="output file", default="output.txt")

# Add a flag
p <- add_argument(p, "--append", help="append to file", flag=TRUE)

# Add multiple arguments together
p <- add_argument(p,
    c("ref", "--date", "--sort"),
    help = c("reference file", "date stamp to use", "sort lines"),
    flag = c(FALSE, FALSE, TRUE))

# Print the help message
print(p)

# Example of custom type, using the example from pythons argparse
setClass("perfectSquare")
setMethod("coerce", c(from = "ANY", to = "perfectSquare"),
    function(from, to) {
      from <- as.numeric(from)
      if (!all.equal(from, as.integer(from))) {
        stop("Type error: ", from, " is not an integer!")
      }
      sqt <- sqrt(from)
      if (sqt != as.integer(sqt)) {
         stop("Type error: ", from, " is not a perfect square!")
      }
      from
    }
)

p2 <- arg_parser("Perfect square checker")
p2 <- add_argument(p2, arg = c("--perfect-square"),
                   help = "A perfect square integer",
                   type = "perfectSquare")

parse_args(p2, c("--perfect-square", 144))
```

`arg.parser` *Create an argument parser.*

Description

This function is deprecated. Use `arg_parser` instead.

Usage

```
arg.parser(description, name = NULL)
```

Arguments

`description`	description of the program
`name`	name of the program

Value

a new `arg.parser` object

`arg_parser` *Create an argument parser.*

Description

This function creates an `arg.parser` object. It infers the program name from the file name of the invoked script.

Usage

```
arg_parser(description, name = NULL, hide.opts = FALSE)
```

Arguments

`description`	description of the program
`name`	name of the program
`hide.opts`	hide the `--opts` argument

Details

The argument parser will be created by default with two arguments: `--help` and `--opts`. The latter argument can be used for loading a list of argument values that are saved in a RDS file.

Value

a new `arg.parser` object

Examples

```
p <- arg_parser("A test program")
```

`include` *Include R script file*

Description

Include R script with behaviour similar to C++ `#include "header.h"`, by searching in the directory where the current script file resides.

Usage

```
include(file)
```

Arguments

`file`	name

`parse.args` *Parse arguments with a parser.*

Description

This function is deprecated. Use `parse_args` instead.

Usage

```
parse.args(parser, argv = commandArgs(trailingOnly = TRUE))
```

Arguments

`parser`	an `arg.parser` object
`argv`	a character vector to parse (arguments and values should already be split by whitespace)

Value

a list with argument values

parse_args — *Parse arguments with a parser.*

Description

This function uses an `arg.parser` object to parse command line arguments or a character vector.

Usage

```
parse_args(parser, argv = NULL)
```

Arguments

`parser`	an `arg.parser` object
`argv`	a character vector to parse (arguments and values should already be split by whitespace); if `NULL`, values will be obtained from `argv` if `argv` exists in the global scope, or from `commandArgs(trailingOnly=TRUE)`.

Value

a list with argument values

Examples

```
p <- arg_parser('pi')
p <- add_argument(p, "--digits",
  help="number of significant digits to print", default=7)

## Not run:
# If arguments are passed from the command line,
# then we would use the following:
argv <- parse_args(p)

## End(Not run)

# For testing purposes, we can pass a character vector:
argv <- parse_args(p, c("-d", "30"))

# Now, the script runs based on the passed arguments
digits <- if (argv$digits > 22) 22 else argv$digits
print(pi, digits=digits)

## Not run:
# We can also save an argument list for later use
saveRDS(argv, "arguments.rds")

# To use the saved arguments, use the --opts argument at the command line
#$ ./script.R --opts arguments.rds
```

```
## End(Not run)
```

`print.arg.parser` *Print the help message for an arg.parser.*

Description

This function prints the help message.

Usage

```
## S3 method for class 'arg.parser'
print(x, ...)
```

Arguments

`x`	an `arg.parser` object
`...`	unused arguments

Details

At the command line, we would use the `--help` or `-help` flag to print the help message: `$ script --help`

`show_arg_labels` *Extract label and help strings from parser.*

Description

Extract label and help strings from parser.

Usage

```
show_arg_labels(parser)
```

Arguments

`parser`	`arg.parser` object

Value

a list containing a `reg.args`, `flags`, and `opt.args` list, which each containing a `label` string and a `help` string

`spaces` *Space string.*

Description

Space string.

Usage

```
spaces(n)
```

Arguments

`n`	number of spaces

Value

a character string containing `n` spaces

Index

add.argument, 2
add_argument, 3
arg.parser, 5
arg_parser, 5

include, 6

parse.args, 6
parse_args, 7
print.arg.parser, 8

show_arg_labels, 8
spaces, 9