Rcal

A function to print a calendar.

Description

A function that prints the requested monthly calendar. The function relies on pipe and the Unix routine cal.

Usage

Rcal(month, year)

Arguments

month  An optional argument, if supplied a number between 1 and 12.
year   An optional argument, if supplied a year.

Details

By default this prints the calendar for the current month and year. Otherwise if a month and year are given and then the calendar for that month and year is printed.

Value

No value is returned.

Author(s)

R. Gentleman

See Also

pipe

Examples

Rcal()
Rcal(6, 1970)
S4Help

A function to find help for S4 classes and generics

Description

This function takes the name of either a class or a generic function and finds a set of related manual pages. The user then selects which manual page they want.

Usage

S4Help(name, signature)

Arguments

name

The name of the S4 class or generic function.

signature

Currently not used.

Details

For S4 classes, the class and any superclasses are found and the user can select which manual page they want. If the supplied name corresponds to a generic function, then that function, or any of its methods can be selected.

Value

NULL is returned, invisibly. The function is called purely for side effect.

Author(s)

R. Gentleman

See Also

help

Examples

## Not run:
S4Help("classRepresentation")
S4Help("coerce")
## End(Not run)
Example functions for the Chapter on Debugging

Description

These functions are used to demonstrate some of the debugging facilities in R.

Usage

```r
asSimpleVector(x, mode = "logical")
convertMode(from, to)
setVNames(x, nm)
subsetAsCharacter(x, i, j)
```

Arguments

- `x`: input
- `mode`: the mode of `x`
- `from`: a parameter
- `to`: another parameter
- `nm`: names for `x`
- `i`: an index
- `j`: another index

Details

A set of functions that can be used to demonstrate debugging principles and practices.

- `asSimpleVector` converts the argument `x` to a simple R vector of the given mode preserving names, dimension and dimnames.
- `subsetAsCharacter` calculates either a vector or a matrix subset of the argument `x` and returns the subset after converting it to be of mode character. It uses `asSimpleVector` to do the conversion to character and thus also preserves any names, dimension or dimnames in the subset.
- `setVNames` sets the names of the given vector `x` to the argument `nm` and then converts `x` to numeric using `asSimpleVector`.
- `convertMode` converts its first argument to the mode of the second argument.

Value

Various values are returned.

Author(s)

S. DebRoy

See Also

`browser`
computeClassLinearization

classList2Graph  Functions to produce graphs from S4 class definitions

Description
Given either a list of classes, or a single class, these functions produce a graph, with the classes as nodes and edges representing subclass/superclass relationships.

Usage
classList2Graph(class, fullNames=TRUE)
class2Graph(class, fullNames=TRUE)

Arguments
class Either the name of a single class, class2Graph, or the names of classes, classList2Graph
fullNames Indicates whether to use fully qualified (by package) names for the class.

Details
Edges are directed, and go from superclasses to subclasses (more specialized to less specialized).

Value
A graph, typically an instance of the graphNEL class.

Author(s)
R. Gentleman

Examples
graphClasses = getClasses("package:graph")
classList2Graph(graphClasses)

computeClassLinearization

Compute the Class Linearization

Description
There are a number of different methods for computing the class linearization. The methods described here are discussed in more detail in the vignette for this package. LPO and computeClassLinearization are aliases, the former being easier to type, the latter more descriptive.

Usage
computeClassLinearization(inClass, C3 = FALSE)
LPO(inClass, C3 = FALSE)
printWithNumbers

Arguments
inClass The class for which a linearization of its is wanted.
C3 Whether or not to use the C3 method in computing the linearization.

Details
For many different computations, especially method dispatch, an algorithm for specifying a linear order of the class inheritance tree is needed. All object oriented programming languages support the computation of a linearization. Local precedence order (LPO) uses the order of the direct superclasses, given in the class definition, in computing the linearization, with earlier superclasses considered more specific than later ones. If there are no duplicate class labels in the hierarchy then this is then simply a bread-first search of the superclass definitions. But when one or more classes are inherited from different superclasses this definition becomes more complicated, and can in fact not be satisfied.

Value
A vector with the class linearization, the.

Author(s)
R. Gentleman

References
Monotonic Superclass Linearization for Dylan, K. Barrett and others, 1996, OOPSLA

Examples
setClass("object")
setClass("grid-layout", contains="object")
setClass("horizontal-grid", contains="grid-layout")
setClass("vertical-grid", contains="grid-layout")
setClass("hv-grid", contains=c("horizontal-grid", "vertical-grid"))
LPO("hv-grid")

printWithNumbers f

Description
A function to print a function together with relevant line numbers. These can be used to decide where to set trace functions etc.

Usage
printWithNumbers(f)

Arguments
f Any R function.
Details

The function is printed, all lines that correspond to potential break points in the code are numbered.

Value

The function is invoked primarily for its side effect; the printing of the function with line numbers. However, the character vector is returned and could be used as input for other tools.

Author(s)

R. Gentleman

See Also

trace, capture.output

Examples

printWithNumbers(convertMode)

randDNA

A function to generate random DNA sequences.

Description

This function generates random DNA sequences, nucleotides are sampled with frequency 0.25 each.

Usage

randDNA(n)

Arguments

n

The length of the sequence desired.

Details

This function generates random sequences of A, C, T and G. Real DNA is quite different, so one should not use these sequences for much other than pedagogical purposes.

Value

A length one character vector, with n characters randomly chosen from A, C, T and G.

Author(s)

R. Gentleman

Examples

randDNA(100)
simplePVect

A function to print a vector at the C level.

Description

This function takes as input any vector, matrix or array of numeric values and passes that array out to C, where it is printed, in order from the first value stored to the last, regardless of the dimensioning information. And information about the location is printed as well.

Usage

simplePVect(iV)

Arguments

iV The input vector.

Details

As above.

Value

NULL is returned. The function is called only for its side effects.

Author(s)

R. Gentleman

Examples

simplePVect(1:3)

simpleRand

A simple interface to C level random number generation.

Description

A simple interface function to R’s C level rng code. Primarily a pedagogical tool to accompany the monograph R for Bioinformatics.

Usage

simpleRand(x, y = "notused")

Arguments

x The number of random numbers to generate.

y Not used in the base implementation, but various exercises involve making use of this.
### simpleSort

#### Details
An interface, via `.Call` to R’s underlying RNG code.

#### Value
The random numbers generated, plus information about the RNG used.

#### Author(s)
R. Gentleman

#### See Also
- `simpleSort`, `.Call`

#### Examples
```r
simpleRand(4)
```

---

`simpleSort`  
*A simple interface to sorting routines in R*

#### Description
The function demonstrates how to access R’s internal sorting routines via the `.Call` interface.

#### Usage
```r
simpleSort(x)
```

#### Arguments
- `x` The vector to be sorted.

#### Details
A simple interface to sorting routines in R. It is intended to be modified following exercises in the accompanying monograph.

#### Value
The sorted vector.

#### Author(s)
R. Gentleman

#### See Also
- `simpleRand`, `.Call`

#### Examples
```r
simpleSort(c(4,2,6))
```
subClassNames

Functions to return the names of either subclasses or superclasses.

Description

Given the name of a S4 class, or a S4 classRepresentation object, these functions return either the names of the direct subclasses or of the direct superclasses.

Usage

subClassNames(x)
superClassNames(x)

Arguments

x

Either the name of a class, or an instance of classRepresentation.

Details

If a name is given then getClass is used to get the class representation object.

Value

A character vector, listing either the direct subclasses or the direct superclasses, depending on which function was called.

Author(s)

R. Gentleman

See Also

gClass

Examples

subClassNames("matrix")
superClassNames("matrix")

superClasses

Return a list of super classes.

Description

This function computes and returns a list of all super classes given a classRepresentation.

Usage

superClasses(x)
superClasses

Arguments

×

Details

This function needs to be rationalized with superClassNames.

Value

A list of the super classes.

Author(s)

R. Gentleman

See Also

superClassNames

Examples

superClassNames(getClass("graphNEL"))
Index

*Topic **manip**
  asSimpleVector, 3
  printWithNumbers, 5
  Rcal, 1
  simpleRand, 7
  simpleSort, 8
  subClassNames, 9
  .Call, 8
  asSimpleVector, 3
  browser, 3
  capture.output, 6
  class2Graph(classList2Graph), 4
  classList2Graph, 4
  computeClassLinearization, 4
  convertMode(asSimpleVector), 3
  getClass, 9
  help, 2
  LPO(computeClassLinearization), 4
  pipe, 1
  printWithNumbers, 5
  randDNA, 6
  Rcal, 1
  S4Help, 2
  setVNames(asSimpleVector), 3
  simplePVect, 7
  simpleRand, 7, 8
  simpleSort, 8, 8
  subClassNames, 9
  subsetAsCharacter(asSimpleVector), 3
  superClasses, 9
  superClassNames, 10
  superClassNames(subClassNames), 9
  trace, 6