Package 'hexView'

February 20, 2015

Version 0.3-3

Title Viewing Binary Files

Author Paul Murrell <paul@stat.auckland.ac.nz>

Maintainer Paul Murrell <paul@stat.auckland.ac.nz>

Depends R (>= 1.8.0)

Description Functions to view files in raw binary form like in a hex editor. Additional functions to specify and read arbitrary binary formats.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-17 10:59:27

R topics documented:

`as.character.rawBlock` *Convert Block of Binary Data into Strings*

Description

This function takes a `"rawBlock"` object and generates human-readable strings for displaying the block. Each string contains a binary offset, the binary data in a raw machine format, and an interpretation of the data in a human-readable format.

Usage

```
## S3 method for class 'rawBlock'
as.character(x, width = NULL, machine = NULL,
             sep1 = "  :  ", sep2 = "  |  ",
             showOffset = TRUE, showHuman = TRUE, ...)
```

Arguments

`x`	A `"rawBlock"` object.
`width`	The number of bytes to display per string.
`machine`	The machine format to display; either `"hex"` or `"binary"`.
`sep1`	A separator to insert between the block offset and the machine format.
`sep2`	A separator to insert between the machine format and the human-readable format.
`showOffset`	If `FALSE`, the column of offsets is not included in the strings.
`showHuman`	If `FALSE`, the column of human-readable format is not included in the strings.
`...`	Other arguments to `as.character`.

Details

If either `width` or `machine` is `NULL`, the relevant value is taken from the `"rawBlock"` object. The human-readable format is taken from the `"rawBlock"` object.

Value

A character vector.

Author(s)

Paul Murrell

See Also

`readRaw` `print.rawBlock`

Examples

```
fileBlock <- readRaw(hexViewFile("rawTest.txt"), width=8)
as.character(fileBlock)
```

`as.character.rawFormat` *Convert Binary File Format into Strings*

Description

This function takes a `"rawFormat"` object and generates human-readable strings for displaying the format. Each string contains a binary offset, the binary data in a raw machine format, and an interpretation of the data in a human-readable format. The format consists of one or more sub-blocks and a heading line is added for each block.

Usage

```
## S3 method for class 'rawFormat'
as.character(x, sep1 = "  :  ", sep2 = "  |  ",
                blockHead = TRUE, blockChar = "=", ...)
```

Arguments

`x`	A `"rawFormat"` object.
`sep1`	A separator to insert between the format offset and the machine format.
`sep2`	A separator to insert between the machine format and the human-readable format.
`blockHead`	A logical indicating whether to print a header between blocks of the format.
`blockChar`	The character used as a prefix to the block names for printing headers between blocks.
`...`	Other arguments to `as.character`.

Details

Information on the number of bytes on each line, the machine representation of each byte and the human-readable format are all taken from the taken from the `"rawBlock"` elements of the `"rawFormat"` object. Consequently each block can have a quite different appearance. Considerable effort is made to line up the separators across all blocks within the format.

Value

A character vector.

Author(s)

Paul Murrell

See Also

readFormat print.rawFormat

Examples

```
fileFormat <- readFormat(hexViewFile("rawTest.int"),
                         memFormat(int1=integer4, int2=integer4))
as.character(fileFormat)
```

---

atomicBlock *Create an atomicBlock Object*

Description

This function creates an "atomicBlock" object, which is a description of a block of binary data. This can be used as part of a description of a binary format.

Usage

```
atomicBlock(type = "char", width = NULL, machine = "hex",
            size = switch(type, char = 1, int = 4, real = 8),
            endian = "little", signed = TRUE)
```

Arguments

type	How the block of binary data will be interpreted. either "char" (an ASCII character), "int" (an integer), or "real" (a floating point number).
width	The number of bytes to print per row when displaying the block.
machine	How to print each byte when displaying; either "hex" or "binary".
size	The number of bytes used to generate each value when interpreting the raw binary as character or numeric data.
endian	The endianness of the binary data; used when interpreting bytes as numeric values.
signed	Whether the bytes should be interpreted as a signed numeric value.

Details

An "atomicBlock" object describes a binary block representing a single value.

Several standard binary types are predefined (with common C type equivalents in brackets): ASCIIchar (char), integer1 (signed char), integer2 (short), integer4 (int, long), integer8 (long long), real4 (float), real8 (double).

Value

An "atomicBlock" object.

Author(s)

Paul Murrell

See Also

memFormat readFormat memBlock vectorBlock lengthBlock mixedBlock markedBlock

Examples

```
# A C long
atomicBlock("int", size=4)
integer4
```

---

blockValue *Extract the Value of a Binary Block*

Description

The blockValue function returns the interpreted value of a block of binary data (a "rawBlock" object).

The blockString function returns a null-terminated string from a block of binary data that is interpreted as a character data.

Usage

```
blockValue(block)

blockString(block)
```

Arguments

block	A "rawBlock" object.

Details

The type of the value returned is determined when the binary block is created (e.g., by readRaw) *not* when the value is extracted by the blockValue function.

The blockString function is useful for extracting a value from a binary block which is a string padded with null characters.

The blockString function throws an error if the block is not interpreted as a character value.

Value

Either a character, or numeric vector depending on how the binary block should be interpreted.

Author(s)

Paul Murrell

See Also

readRaw readFormat

Examples

```
charBlock <- readRaw(hexViewFile("rawTest.txt"), width=8)
blockValue(charBlock)
blockString(charBlock)

intBlock <- readRaw(hexViewFile("rawTest.int"), human="int")
blockValue(intBlock)
```

---

hexViewFile *Specify an Example File*

Description

This is just a convenience function for specifying one of the example files contained in the hexView package. It is used in examples in hexView help pages.

Usage

```
hexViewFile(filename)
```

Arguments

filename	The name of the example file.

Value

The full path to the appropriate example file.

Author(s)

Paul Murrell

Examples

```
hexViewFile("rawTest.txt")
```

`markedBlock` *Create a markedBlock Object*

Description

This function creates a `"markedBlock"` object, which is a description of a block of binary data. This can be used as part of a description of a binary format.

Usage

```
markedBlock(marker=integer4,
            switch=function(marker) { ASCIIchar })
```

Arguments

`marker`	A `"memBlock"` object.
`switch`	A function that returns a `"memBlock"` object.

Details

A `"markedBlock"` object describes a block of binary data that consists of a "marker" block containing information on further blocks. The `marker` block is read first, then this block is passed to the `switch` function. The `switch` function can look at the contents of the `marker` block and decide what sort of block should be read next. The result of the `switch` function is read from the end of the `marker` block. The `marker` block and the result of the `switch` function can be any type of `"memBlock"` object.

Value

A `"markedBlock"` object.

Author(s)

Paul Murrell

See Also

`memFormat` `readFormat` `memBlock` `atomicBlock` `vectorBlock` `lengthBlock` `mixedBlock`

Examples

```
# A single-byte integer which dictates how many
# subsequent four-byte reals to read
markedBlock(integer1,
            function(marker) {
              lengthBlock(real4, blockValue(marker))
            })
```

`memBlock` *Create a memBlock Object*

Description

This function creates a `"memBlock"` object, which is a description of a block of binary data. This can be used as part of a description of a binary format.

Usage

```
memBlock(nbytes = 1, width = NULL, machine = "hex")
```

Arguments

`nbytes`	The number of bytes in the block.
`width`	The number of bytes to print per row when displaying the block.
`machine`	How to print each byte when displaying; either `"hex"` or `"binary"`.

Details

A binary block that is read in from a file using this description is interpreted as `nbytes` single-byte characters.

The description includes parameters controlling how a block of data should be displayed if this description is used to read in a block of binary data.

Value

A `"memBlock"` object.

Author(s)

Paul Murrell

See Also

`memFormat` `readFormat` `atomicBlock` `vectorBlock` `lengthBlock` `mixedBlock` `markedBlock`

Examples

```
memBlock(8)
```

memFormat *Create a memFormat Object*

Description

This function creates a "memFormat" object which is a description of a binary file format.

Usage

```
memFormat(...)
```

Arguments

...	One or more "memBlock" objects.

Details

A "memFormat" object is made up of one or more "memBlock" objects.

Value

A "memFormat" object.

Author(s)

Paul Murrell

See Also

readFormat memBlock atomicBlock vectorBlock lengthBlock mixedBlock markedBlock

Examples

```
memFormat(int1=integer4, int2=integer4)
```

mixedBlock *Create a mixedBlock Object*

Description

This function creates a "mixedBlock" object, which is a description of a block of binary data. This can be used as part of a description of a binary format.

Usage

```
mixedBlock(...)
```

Arguments

...	One or more "memBlock" objects.

Details

A "mixedBlock" object describes a block of binary data that consists of a series of sub-blocks. Each sub-block can be any type of "memBlock" object.

Value

A "mixedBlock" object.

Author(s)

Paul Murrell

See Also

memFormat readFormat memBlock atomicBlock vectorBlock lengthBlock markedBlock

Examples

```
# A line of text followed by a four-byte integer
mixedBlock(ASCIIline, integer4)
```

---

print.rawBlock *Print Method for Block of Binary Data*

Description

This function displays a "rawBlock" object. Each line of output contains a binary offset, the binary data in a raw machine format, and an interpretation of the data in a human-readable format. The object contains parameters controlling the format of the display, some of which may be overridden in the call to print.

Usage

```
## S3 method for class 'rawBlock'
print(x, width = NULL, machine = NULL,
      sep1 = "  :  ", sep2 = "  |  ",
      showOffset = TRUE, showHuman = TRUE,
      page = FALSE, ...)
```

Arguments

x	A "rawBlock" object.
width	The number of bytes to display per line of output.
machine	The machine format to display; either "hex" or "binary".
sep1	A separator to insert between the block offset and the machine format.
sep2	A separator to insert between the machine format and the human-readable format.
showOffset	If FALSE, the column of offsets is not printed.
showHuman	If FALSE, the column of human-readable format is not printed.
page	If TRUE the output is sent to the file viewer specified by getOption("pager").
...	Other arguments to print.

Details

If either width or machine is NULL, the relevant value is taken from the "rawBlock" object. The human-readable format is taken from the "rawBlock" object.

Author(s)

Paul Murrell

See Also

readRaw as.character.rawBlock

Examples

```
fileBlock <- readRaw(hexViewFile("rawTest.txt"))
print(fileBlock)
print(fileBlock, width=8)
print(fileBlock, machine="binary", width=4)
```

---

print.rawFormat *Print Method for Binary File Format*

Description

This function displays a "rawFormat" object. Each line of output contains a binary offset, the binary data in a raw machine format, and an interpretation of the data in a human-readable format. The format consists of one or more sub-blocks and a heading line is added for each block. The object contains parameters controlling the format of the display, some of which may be overridden in the call to print.

Usage

```
## S3 method for class 'rawFormat'
print(x, sep1 = "  :  ", sep2 = "  |  ",
                  blockHead = TRUE, blockChar = "=", page = FALSE, ...)
```

Arguments

x	A "rawFormat" object.
sep1	A separator to insert between the format offset and the machine format.
sep2	A separator to insert between the machine format and the human-readable format.
blockHead	A logical indicating whether to print a header between blocks of the format.
blockChar	The character used as a prefix to the block names for printing headers between blocks.
page	If TRUE the output is sent to the file viewer specified by getOption("pager").
...	Other arguments to print.

Details

Information on the number of bytes on each line, the machine representation of each byte and the human-readable format are all taken from the taken from the "rawBlock" elements of the "rawFormat" object. Consequently each block can have a quite different appearance. Considerable effort is made to line up the separators across all blocks within the format.

Author(s)

Paul Murrell

See Also

readFormat as.character.rawFormat

Examples

```
fileFormat <- readFormat(hexViewFile("rawTest.int"),
                         memFormat(int1=integer4, int2=integer4))
print(fileFormat)
print(fileFormat, sep2=":")
```

readEViews *Read an Eviews File*

Description

This function reads a file in Eviews format (Eviews is an econometrics package).

Usage

```
readEViews(filename, as.data.frame = TRUE)
```

Arguments

filename	The name of the file.
as.data.frame	If TRUE the result is a data frame; otherwise a list of variables is returned.

Details

This function is just a demonstration of how the functions in this package can be used to read a complex binary format. It has been tested on a few sample files (and works), but there is no guarantee it will work for all Eviews files (this is not helped by the fact that it is based on reverse-engineering information about the Eviews format, NOT an official description of the format.

Value

Either a data frame or a list of variables.

Author(s)

Paul Murrell

References

http://www.eviews.com/ and http://www.ecn.wfu.edu/~cottrell/eviews_format/

Examples

```
readEViews(hexViewFile("data4-1.wf1"))
```

readFormat *Read a Binary File*

Description

Read the raw binary content of a file using a description of the binary format.

Usage

```
readFormat(file, format, width = NULL, offset = 0, machine = "hex",
           flatten = TRUE)
```

Arguments

file	The name of a file or a connection.
format	A "memFormat" object.
width	The number of bytes to print per row when displaying the file.
offset	An offset within the file to start reading.
machine	How to print each byte when displaying the file; either "hex" or "binary".
flatten	If TRUE the list of blocks created from the "memFormat" description are flattened to a list of depth 1.

Details

This function uses a "memFormat" description to read the raw binary content of a file and interpret sub-blocks of the file as distinct (blocks of) values.

The "memFormat" can described a nested structure of blocks. The flatten argument is used to convert nested format structures to a flat (depth of one) structure.

The format is always flattened for display, but extracting

Value

A "rawFormat" object, which is a list:

blocks	A list (of lists) of "rawBlock" objects.
offset	The offset in the file where reading began.
nbytes	The number of bytes read from the file.

Author(s)

Paul Murrell

See Also

viewFormat memFormat as.character.rawFormat print.rawFormat readRaw readBin

Examples

```
fileFormat <- readFormat(hexViewFile("rawTest.int"),
                         memFormat(int1=integer4, int2=integer4))
blockValue(fileFormat$blocks$int2)

fileFormat <- readFormat(hexViewFile("rawTest.int"),
                         memFormat(integers=vectorBlock(integer4, 20)))
blockValue(fileFormat$blocks$integers)
```

---

readRaw *Read the Raw Binary Content of a File*

Description

Read the contents of a file as bytes and create an object containing the raw data, plus optionally an interpretation of the bytes as numeric values, plus parameters controlling how to display the data.

Usage

```
readRaw(file, width = NULL, offset = 0, nbytes = NULL,
        machine = "hex", human = "char",
        size = switch(human, char = 1, int = 4, real = 8),
        endian = .Platform$endian, signed = TRUE)
```

Arguments

file	The name of a file or a connection.
width	The number of bytes to print per row when displaying the data.
offset	An offset within the file to start reading.
nbytes	The number of bytes to read from the file. NULL means read the whole file.
machine	How to print each byte when displaying; either "hex" or "binary".
human	How to print a human-readable form of the data; either "char" (an ASCII character), "int" (an integer), or "real" (a floating point number).
size	The number of bytes used to generate each value when interpreting the raw binary as character or numeric data.
endian	The endianness of the binary data; used when interpreting bytes as numeric values.
signed	Whether the bytes should be interpreted as a signed numeric value.

Details

Each individual byte is printed in the appropriate machine format, but there is only one value printed in the appropriate human format for every size bytes. Consequently, the width must be a multiple of the size.

Value

An object of class "rawBlock".

Author(s)

Paul Murrell

See Also

viewRaw readBin as.character.rawBlock print.rawBlock blockValue readFormat

Examples

```
readLines(hexViewFile("rawTest.txt"))

fileBlock <- readRaw(hexViewFile("rawTest.txt"), width=8)
blockValue(fileBlock)

fileBlock <- readRaw(hexViewFile("rawTest.int"), human="int")
blockValue(fileBlock)
```

---

vectorBlock *Create a vectorBlock Object*

Description

These functions create a "vectorBlock" or lengthBlock object, which are a descriptions of a block of binary data. These can be used as part of a description of a binary format.

Usage

```
vectorBlock(block = ASCIIchar, length = 1)
lengthBlock(length = integer4, block = ASCIIchar)
```

Arguments

block	An object derived from the "memBlock" class, e.g., an "atomicBlock" object.
length	The number of block objects in the overall binary block.

Details

These objects describe a block of binary data that consists of a repeating sub-block. The sub-block can be any type of "memBlock" object.

There is also a predefined ASCIIline block, which is a block of single-byte characters terminated by a newline character.

Value

A "vectorBlock" or lengthBlock object.

Author(s)

Paul Murrell

See Also

memFormat readFormat memBlock atomicBlock mixedBlock markedBlock

Examples

```
# A block of 20 four-byte blocks
# which are interpreted as integer values
vectorBlock(integer4, 20)
```

---

viewFormat *View a Binary File*

Description

Displays the raw bytes of a file like a hex editor, showing offsets within the file, raw bytes in binary or hexadecimal form, and a human-readable representation of the bytes as either ASCII characters, integers, or real values. The file is broken up into blocks according to a supplied file format specification.

Usage

```
viewFormat(..., page = FALSE)
```

Arguments

...	Arguments passed to the function readFormat, most importantly, a "memFormat" describing the file format.
page	If TRUE, the output is sent to the application set up to display text files as per getOption("pager").

Details

This function is only called for its side-effect, which is to display the file.

Author(s)

Paul Murrell

See Also

readFormat viewRaw

Examples

```
viewFormat(hexViewFile("rawTest.int"),
           memFormat(int1=integer4, int2=integer4))
viewFormat(hexViewFile("rawTest.int"),
           memFormat(integers=vectorBlock(integer4, 20)))
```

---

viewRaw *View the Raw Binary Content of a File*

Description

Displays the raw bytes of a file like a hex editor, showing offsets within the file, raw bytes in binary or hexadecimal form, and a human-readable representation of the bytes as either ASCII characters, integers, or real values.

Usage

```
viewRaw(..., page = FALSE)
```

Arguments

...	Arguments passed to the function readRaw.
page	If TRUE, the output is sent to the application set up to display text files as per getOption("pager").

Details

This function is only called for its side-effect, which is to display the file.

Author(s)

Paul Murrell

See Also

readRaw viewFormat

Examples

```
viewRaw(hexViewFile("rawTest.txt"), width=8)
viewRaw(hexViewFile("rawTest.txt"), machine="binary", width=4)

# UNICODE text
# rawTest.unicode created using Notepad on Windows
viewRaw(hexViewFile("rawTest.unicode"), width=8)

viewRaw(hexViewFile("rawTest.int"), human="int")
viewRaw(hexViewFile("rawTest.real"), human="real", width=8, endian="big")
```

Index