

Package ‘uncompress’

April 19, 2009

Version 1.31

Date 2008-11-16

Title Uncompress

Author Nicholas Vinen

Maintainer Nicholas Vinen <hb@rp.x256.org>

Description This R Add-On provides the function `uncompress(raw)` for decompressing .Z files. It returns the decompressed data as a raw (binary) string. The compressed data is passed in in the same format. It returns NULL if the input data is invalid.

License Unlimited

Repository CRAN

Date/Publication 2008-11-16 14:52:07

R topics documented:

<code>rawToLines</code>	1
<code>uncompress</code>	3
<code>uncompressToLines</code>	4

Index	7
--------------	---

<code>rawToLines</code>	<i>Convert text encoded as a raw binary vector to a string vector, one line per entry</i>
-------------------------	---

Description

This function takes text data encoded as a raw binary vector, splits it according to line breaks (LF [Unix] or CR/LF [Windows] style), and returns a string vector with one line per entry. If there is a blank line at the end of the file, it is included. Optionally, you can skip a number of lines from the start of the file by passing an integer as the second argument. You can also, optionally, limit the number of lines that will be returned by passing an integer as the third argument.

Usage

```
rawToLines(data, start_line = 0, max_line_count = 999999999)
```

Arguments

`data` The raw binary data to split.

`start_line` The index of the first line to return. Defaults to 0. In effect, processing of the data begins after skipping this many lines from the start. If it is greater than the number of lines in the file, an error will be returned.

`max_line_count` If more than this many lines would be returned, the rest are ignored. Otherwise it has no effect. The default is 999999999.

Examples

```
library("uncompress")
## Not run:

## Example 1 - load a file as binary data, then split it into strings by line.
handle <- file("file.txt", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
text_lines <- rawToLines(data)
print(text_lines)

## Example 2 - the same, except does the conversion 1000 lines at a time.
handle <- file("file.txt", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
start <- 0
while( 1 ) {
  text_lines <- rawToLines(data, start, 1000)
  if( length(text_lines) == 0 )
    break;
  print(text_lines);
  if( length(text_lines) < 1000 )
    break;
  start <- start + 1000
}

## End(Not run)
```

`uncompress`*Uncompress .Z file data*

Description

Uncompresses data compressed with the "compress" utility. Data pass in and returned are both raw binary strings. Invalid compressed data results in an error being returned.

Usage

```
uncompress(data)
```

Arguments

`data` The raw binary data to decompress.

Examples

```
library("uncompress")
## Not run:

## Example 1 - extracting text data from a .Z file.
handle <- file("file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
uncomp_data <- uncompress(data)
# At this point you might want to check if uncomp_data is NULL in case the
# file is corrupt.
# This is assuming it's Unix-style text (i.e. \n separates lines).
# If it's Windows-style text you may need to use "\r\n".
lines <- strsplit(rawToChar(uncomp_data), "\n")
print(lines)

## Example 2 - extracting text data from a .Z file at an HTTP host.
handle <- url("http://host/path/file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
uncomp_data <- uncompress(data)
# At this point you might want to check if uncomp_data is NULL in case the
# file is corrupt.
# This is assuming it's Unix-style text (i.e. \n separates lines).
# If it's Windows-style text you may need to use "\r\n".
lines <- strsplit(rawToChar(uncomp_data), "\n")
print(lines)

## Example 3 - downloading a .Z file, then uncompressing it.
```

```

# Mode is important, in Windows if you don't use binary mode the compressed
# data will be corrupted.
download.file("http://host/path/file.Z", "file.Z", mode="wb")
handle <- file("file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
uncomp_data <- uncompress(data)
# At this point you might want to check if uncomp_data is NULL in case the
# file is corrupt.
# This is assuming it's Unix-style text (i.e. \n separates lines).
# If it's Windows-style text you may need to use "\r\n".
lines <- strsplit(rawToChar(uncomp_data), "\n")
print(lines)

## Example 4 - downloading a .Z file, writing the uncompressed data to a
## local file.
handle <- url("http://host/path/file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
uncomp_data <- uncompress(data)
handle <- file("file.Z", "wb")
writeBin(uncomp_data, handle)
close(handle)

## End(Not run)

```

uncompressToLines *Uncompress data and return it as a string vector*

Description

This function is a wrapper for `uncompress()` and `rawToLines()`. First it uncompresses the raw binary vector passed as the first argument, then it splits it according to line breaks (LF [Unix] or CR/LF [Windows] style), and returns a string vector with one line per entry. If there is a blank line at the end of the file, it is included. Optionally, you can skip a number of lines from the start of the file by passing an integer as the second argument. You can also, optionally, limit the number of lines that will be returned by passing an integer as the third argument.

Usage

```
uncompressToLines(data, start_line = 0, max_line_count = 999999999)
```

Arguments

`data` The raw compressed binary data.

`start_line` The index of the first line to return. Defaults to 0. In effect, processing of the data begins after skipping this many lines from the start. If it is greater than the number of lines in the file, an error will be returned.

`max_line_count` If more than this many lines would be returned, the rest are ignored. Otherwise it has no effect. The default is 999999999.

Examples

```
library("uncompress")
## Not run:

## Example 1 - uncompress a file, then split it into strings by line.
handle <- file("file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
text_lines <- uncompressToLines(data)
print(text_lines)

## Example 2 - the same, except does the conversion 1000 lines at a time.
## Note that this will uncompress all the data for each call!
## Example 3 is a much better way to do it.
handle <- file("file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
start <- 0
while( 1 ) {
  text_lines <- uncompressToLines(data, start, 1000)
  if( length(text_lines) == 0 )
    break;
  print(text_lines);
  if( length(text_lines) < 1000 )
    break;
  start <- start + 1000
}

## Example 3 - Smarter version of Example 2
handle <- file("file.Z", "rb")
# The size here is arbitrary, it should be large enough for most files,
# adjust as necessary.
data <- readBin(handle, "raw", 99999999)
close(handle)
data <- uncompress(data);
start <- 0
while( 1 ) {
  text_lines <- rawToLines(data, start, 1000)
  if( length(text_lines) == 0 )
    break;
```

```
print(text_lines);
if( length(text_lines) < 1000 )
  break;
start <- start + 1000
}

## End(Not run)
```

Index

*Topic **file**

rawToLines, [1](#)

uncompress, [2](#)

uncompressToLines, [4](#)

rawToLines, [1](#)

uncompress, [2](#)

uncompressToLines, [4](#)