

# Package ‘zip’

March 11, 2019

**Title** Cross-Platform 'zip' Compression

**Version** 2.0.1

**Author** Gábor Csárdi, Kuba Podgórski, Rich Geldreich

**Maintainer** Gábor Csárdi <csardi.gabor@gmail.com>

**Description** Cross-Platform 'zip' Compression Library. A replacement for the 'zip' function, that does not require any additional external tools on any platform.

**License** CC0

**LazyData** true

**URL** <https://github.com/r-lib/zip#readme>

**BugReports** <https://github.com/r-lib/zip/issues>

**RoxygenNote** 6.1.1

**Suggests** covr, processx, R6, testthat, withr

**Encoding** UTF-8

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-03-11 18:20:03 UTC

## R topics documented:

unzip . . . . .	2
unzip_process . . . . .	3
zip . . . . .	3
zip_list . . . . .	6
zip_process . . . . .	6
<b>Index</b>	<b>8</b>

---

`unzip`*Uncompress 'zip' Archives*

---

**Description**

`unzip()` always restores modification times of the extracted files and directories.

**Usage**

```
unzip(zipfile, files = NULL, overwrite = TRUE, junkpaths = FALSE,
      exdir = ".")
```

**Arguments**

<code>zipfile</code>	Path to the zip file to uncompress.
<code>files</code>	Character vector of files to extract from the archive. Files within directories can be specified, but they must use a forward slash as path separator, as this is what zip files use internally. If <code>NULL</code> , all files will be extracted.
<code>overwrite</code>	Whether to overwrite existing files. If <code>FALSE</code> and a file already exists, then an error is thrown.
<code>junkpaths</code>	Whether to ignore all directory paths when creating files. If <code>TRUE</code> , all files will be created in <code>exdir</code> .
<code>exdir</code>	Directory to uncompress the archive to. If it does not exist, it will be created.

**Permissions**

If the zip archive stores permissions and was created on Unix, the permissions will be restored.

**Examples**

```
## Some files to zip up
dir.create(tmp <- tempfile())
cat("first file", file = file.path(tmp, "file1"))
cat("second file", file = file.path(tmp, "file2"))

zipfile <- tempfile(fileext = ".zip")
zipr(zipfile, tmp)

## List contents
zip_list(zipfile)

## Extract
tmp2 <- tempfile()
unzip(zipfile, exdir = tmp2)
```

---

`unzip_process`*Class for an external unzip process*

---

**Description**

`unzip_process()` returns an R6 class that represents an unzip process. It is implemented as a subclass of [processx::process](#).

**Usage**

```
unzip_process()
```

**Value**

An `unzip_process` R6 class object, a subclass of [processx::process](#).

**Using the `unzip_process` class**

```
up <- unzip_process()$new(zipfile, exdir = ".")
```

See [processx::process](#) for the class methods.

Arguments:

- `zipfile`: Path to the zip file to uncompress.
- `exdir`: Directory to uncompress the archive to. If it does not exist, it will be created.

**Examples**

```
ex <- system.file("example.zip", package = "zip")
tmp <- tempfile()
up <- unzip_process()$new(ex, exdir = tmp)
up$wait()
up$get_exit_status()
dir(tmp)
```

---

`zip`*Compress Files into 'zip' Archives*

---

**Description**

`zipr` and `zip` both create a new zip archive file.

**Usage**

```
zip(zipfile, files, recurse = TRUE, compression_level = 9)
```

```
zipr(zipfile, files, recurse = TRUE, compression_level = 9)
```

```
zip_append(zipfile, files, recurse = TRUE, compression_level = 9)
```

```
zipr_append(zipfile, files, recurse = TRUE, compression_level = 9)
```

**Arguments**

zipfile	The zip file to create. If the file exists, zip overwrites it, but zip_append appends to it.
files	List of file to add to the archive. See details below about absolute and relative path names.
recurse	Whether to add the contents of directories recursively.
compression_level	A number between 1 and 9. 9 compresses best, but it also takes the longest.

**Details**

zipr\_append and zip\_append append compressed files to an existing 'zip' file.

**Value**

The name of the created zip file, invisibly.

**Permissions**

zipr() (and zip(), zipr\_append(), etc.) add the permissions of the archived files and directories to the ZIP archive, on Unix systems. Most zip and unzip implementations support these, so they will be recovered after extracting the archive.

Note, however that the owner and group (uid and gid) are currently omitted, even on Unix.

**Relative paths**

The different between zipr and zip is how they handle the relative paths of the input files.

For zip (and zip\_append), the root of the archive is supposed to be the current working directory. The paths of the files are fully kept in the archive. Absolute paths are also kept. Note that this might result non-portable archives: some zip tools do not handle zip archives that contain absolute file names, or file names that start with ../ or ./ . This behavior is kept for compatibility, and we suggest that you use zipr and zipr\_append for new code.

E.g. for the following directory structure:

```
foo
  bar
    file1
```

```

  bar2
    file2
foo2
  file3

```

Assuming the current working directory is foo, the following zip entries are created by zip:

```

zip("x.zip", c("bar/file1", "bar2", "../foo2"))
zip_list("x.zip")$filename
#> bar/file1
#> bar2
#> bar2/file2
#> ../foo2
#> ../foo2/file3

```

For zipr (and zipr\_append), each specified file or directory in files is created as a top-level entry in the zip archive. We suggest that you use zip and zip\_append for new code, as they don't create non-portable archives. For the same directory structure, these zip entries are created:

```

zipr("x.zip", c("bar/file1", "bar2", "../foo2"))
zip_list("x.zip")$filename
#> file1
#> bar2
#> bar2/file2
#> foo2
#> foo2/file3

```

## Examples

```

## Some files to zip up
dir.create(tmp <- tempfile())
cat("first file", file = file.path(tmp, "file1"))
cat("second file", file = file.path(tmp, "file2"))

zipfile <- tempfile(fileext = ".zip")
zipr(zipfile, tmp)

## List contents
zip_list(zipfile)

## Add another file
cat("third file", file = file.path(tmp, "file3"))
zipr_append(zipfile, file.path(tmp, "file3"))
zip_list(zipfile)

```

---

zip_list	<i>List Files in a 'zip' Archive</i>
----------	--------------------------------------

---

**Description**

List Files in a 'zip' Archive

**Usage**

```
zip_list(zipfile)
```

**Arguments**

zipfile            Path to an existing ZIP file.

**Value**

A data frame with columns: filename, compressed\_size, uncompressed\_size, timestamp, permissions.

---

zip_process	<i>Class for an external zip process</i>
-------------	--

---

**Description**

zip\_process() returns an R6 class that represents a zip process. It is implemented as a subclass of [processx::process](#).

**Usage**

```
zip_process()
```

**Value**

A zip\_process R6 class object, a subclass of [processx::process](#).

**Using the zip\_process class**

```
zp <- zip_process()$new(zipfile, files, recurse = TRUE)
```

See [processx::process](#) for the class methods.

Arguments:

- zipfile: Path to the zip file to create.
- files: List of file to add to the archive. Each specified file or directory in is created as a top-level entry in the zip archive.
- recurse Whether to add the contents of directories recursively.

**Examples**

```
dir.create(tmp <- tempfile())
write.table(iris, file = file.path(tmp, "iris.ssv"))
zipfile <- tempfile(fileext = ".zip")
zp <- zip_process()$new(zipfile, tmp)
zp$wait()
zp$get_exit_status()
zip_list(zipfile)
```

# Index

`processx::process`, 3, 6

`unzip`, 2

`unzip_process`, 3

`zip`, 3

`zip_append(zip)`, 3

`zip_list`, 6

`zip_process`, 6

`zipr(zip)`, 3

`zipr_append(zip)`, 3