

1 Extensions for RubyZip

```
"goi/zipx.rb" 1a ≡
# rubyzip extensions
<License 1b>
require 'rubygems'
require 'zip/zip'
require 'zip/zipfilesystem'
require 'fileutils'
require 'goi/tools'
<definitions 2b,... >
module Goi
  module_function
  <Goi functions 1c,... >
end
◇
```

```
<License 1b> ≡
# Copyright (C) 20045, Javier Goizueta <javier@goizueta.info>
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
◇
```

Macro referenced in [1a](#).

1.1 Packing a complete directory

```
<Goi functions 1c> ≡
#Zip a directory (path) into a zip file ( zipfile )
#with options :
# - :nozips exclude .zip files ( default is false )
# - :base include the base directory name in the zip entries ( default is false )
# - if a block is passed it is used filter which files to zip
def zip_dir(path, zipfile, options={})
  exclude_zips=opt(options[:nozips],false)
  include_base=opt(options[:base],false)
  no_times =opt(options[:no_times],false)

  base = File.expand_path(path)
  #puts base
  base = File.dirname(base) if include_base
  #puts base

  File.delete zipfile if File.exist?( zipfile )
  patt = File.join path, '**/*'
  Zip::ZipFile.open(zipfile, Zip::ZipFile::CREATE) do |zp|
    Dir[patt].each do |fn|
      next if exclude_zips and fn[-4..-1].downcase==' .zip'
      next if File.expand_path(zipfile).downcase == File.expand_path(fn,path).downcase
      next if block_given? && !yield(File.expand_path(fn,path))
      entry = relative_path(fn, base)
      if File.directory?(fn)
        zp.mkdir entry
        zp.file .utime(File.mtime(fn), entry) unless no_times
      else
        zp.add entry, fn
        zp.file .utime(File.mtime(fn), entry) unless no_times
      end
    end
  end
end
```

```

    end
  end
  ◇

```

Macro defined by [1c](#), [2a](#).
 Macro referenced in [1a](#).

1.2 Unpacking a complete directory

```

<Goi functions 2a> ≡
def unzip_dir(zipfile, path, options={}, &onexists)
  no_times = opt(options[:no_times], false)
  FileUtils.mkdir_p path
  Zip::ZipFile.open(zipfile) do |zp|
    zp.each do |entry|
      if entry.directory?
        ext_fn = File.expand_path(entry.name, path)
        FileUtils.mkdir_p ext_fn
        begin
          File.utime(entry.mtime, entry.mtime, ext_fn) unless no_times
        rescue Errno::EACCES
          # --utime produces a permission error on Windows 98
        end
      end
    end
  zp.each do |entry|
    if entry.file?
      ext_fn = File.expand_path(entry.name, path)
      zp.extract entry.name, ext_fn, &onexists
      File.utime(entry.mtime, entry.mtime, ext_fn) unless no_times
    end
  end
end
  ◇

```

Macro defined by [1c](#), [2a](#).
 Macro referenced in [1a](#).

1.3 Fix timestamp problem

The rubyzip library (versions up to 0.5.5) doesn't save correct timestamps in newly created zip files. This fix works for rubyzip 0.5.5 and Ruby 1.8.2.

```

<definitions 2b> ≡
if ::VERSION=="1.8.2" || ::VERSION=="1.8.4" || ::VERSION=="1.8.5" || ::VERSION=="1.8.6"
  module Zip
    class ZipOutputStream
      def put_next_entry(entry, level = Zlib::DEFAULT_COMPRESSION)
        raise ZipError, "zip_stream_is_closed" if @closed
        newEntry = entry.kind_of?(ZipEntry) ? entry : (entry.respond_to?(:__getobj__) ? entry.__getobj__ :
ZipEntry.new(@fileName, entry.to_s))
        init_next_entry(newEntry)
        @currentEntry=newEntry
      end
    end
    class ZipEntry
      def time
        if @extra["UniversalTime"] && @extra["UniversalTime"].mtime
          @extra["UniversalTime"].mtime
        else
          @time
        end
      end
    end
  end
end

```

```

    def mtime
      self.time
    end
  end
end
end
end
◇

```

Macro defined by [2b, 3](#).
Macro referenced in [1a](#).

Las últimas distribuciones de la versión 1.8.2 de Ruby (edición 15 de "One click installer") han introducido un cambio en el comportamiento de `Array#pack` (la conversión a "C" de nil antes lo trataba como 0, ahora da error) que produce un error al a cerrar un archivo zip en el que se han modificado las horas si se usa la función `put_next_entry` anterior.

```

<definitions 3> ≡
  if ::VERSION=="1.8.2" || ::VERSION=="1.8.4" || ::VERSION=="1.8.5" || ::VERSION=="1.8.6"
  module Zip
    class ZipExtraField
      class UniversalTime
        def pack_for_local
          s = [@flag || 0].pack("C")
            @flag & 1 != 0 and s << [@mtime.to_i].pack("V")
            @flag & 2 != 0 and s << [@atime.to_i].pack("V")
            @flag & 4 != 0 and s << [@ctime.to_i].pack("V")
          s
        end

        def pack_for_c_dir
          s = [@flag || 0].pack("C")
            @flag & 1 == 1 and s << [@mtime.to_i].pack("V")
          s
        end
      end
    end
  end
end
end
◇

```

Macro defined by [2b, 3](#).
Macro referenced in [1a](#).

2 Tests

```

Goi::zip_dir '.', 'prueba.zip'
Goi::unzip_dir 'prueba.zip', '../xzip'

```

3 Índices

3.1 Archivos

"goi/zipx.rb" Defined by [1a](#).

3.2 Fragmentos

<Goi functions [1c, 2a](#)> Referenced in [1a](#).

<License [1b](#)> Referenced in [1a](#).

<definitions [2b, 3](#)> Referenced in [1a](#).

3.3 Identificadores