AndyPatterns

Cloning Directories in Ruby using Hard Links

Hard links exist under windows 7 so you can clone huge directories or files without taking up any extra disk space. Both the original and the copy are equal and apps can't tell the difference between them - because we are using true hard links (not shortcuts or symbolic links).

The dos command for cloning a file is simply

mklink /H original clone

If you delete the original file, the clone is still there and indistinguishable from the original file. You can clone as many times as you like and no extra disk space is used.

I'm not aware of a way of recursively hard linking directories (files as hard links, sub directories as real directories) in windows ~~or linux~~ using the standard commands, so here is a recursive directory cloning utility written in Ruby. It uses the FileUtils.ln method to do the file cloning. This works under both windows 7 and linux.

```
# ln_r
# Copy a directory recursively creating hardlinks for files and real dirs for directories
# Andy Bulka, Reilly Beacom
# version 1.5

require 'fileutils'
require 'optparse'

def ln_r source, target, options = {:verbose => true, :report => true, :countsize => false,
options[:deletetarget] => true}

  verbose = options[:verbose]
  puts "ln_r copying and hard linking from #{source} to #{target}" if verbose
  puts "..." if verbose

  raise "source not a directory" if not File.directory?(source)

  # Add trailing slash
  source = File.join(source, "")
  target = File.join(target, "")

  # Ensure target dir exists before we start and delete destination files
  FileUtils.mkdir_p target
  FileUtils.rm_r Dir.glob(File.join(target, '/*')) if options[:deletetarget]

  total_file_sizes = 0
  Dir.glob(File.join(source, '**/*')).each do | source_path |

    target_path = source_path.gsub Regexp.new("^" + source), target
    if File.file? source_path

      FileUtils.mkdir_p File.dirname(target_path)
      FileUtils.ln source_path, target_path
      total_file_sizes += File.size(source_path) if options[:countsize]
      puts "created hard link #{target_path}  (source: #{source_path}" if verbose

    else
      FileUtils.mkdir_p target_path
      puts "created directory " + target_path if verbose
    end
  end
  puts "Done copying/linking." if verbose

  def number_with_delimiter(number, delimiter=",")
    number.to_s.gsub(/(\d)(?=(\d\d\d)+(?!\d))/, "\\1#{delimiter}")
  end
  puts "Bytes saved by linking: #{number_with_delimiter(total_file_sizes/1024000)} Mb" if
options[:countsize]

  if options[:report]
    puts
    puts "---- RESULT: SOURCE DIRECTORY"
    puts Dir.glob(File.join(source, '/**/*'))
    puts "---- TARGET DIRECTORY"
    puts Dir.glob(File.join(target, '/**/*'))
    puts "---- RESULT END"
    puts
  end

end



# This hash will hold all of the options
# parsed from the command-line by
# OptionParser.
options = {}

optparse = OptionParser.new do|opts|
  # Set a banner, displayed at the top
  # of the help screen.
  opts.banner = "Usage: ln_r.rb [options] source_dir target_dir"

  # Define the options, and what they do
  options[:verbose] = false
  opts.on( '-v', '--verbose', 'Output more information' ) do
    options[:verbose] = true
  end

  options[:test] = false
  opts.on( '-t', '--test', 'Run test copy on test data dirs - Andy only' ) do
    options[:test] = true
  end

  options[:report] = false
  opts.on( '-r', '--report', 'Display directory of source and target dirs after finish' ) do
    options[:report] = true
  end

  options[:countsize] = false
  opts.on( '-c', '--countsize', 'Display bytes saved by using hard linking' ) do
    options[:countsize] = true
  end

  options[:deletetarget] = true
  opts.on( '-d', '--dontdeletetarget', 'Dont rm -r * target directory first' ) do
    options[:deletetarget] = false
  end

  #options[:logfile] = nil
  #opts.on( '-l', '--logfile FILE', 'Write log to FILE' ) do|file|
  #  options[:logfile] = file
  #end

  # This displays the help screen, all programs are
  # assumed to have this option.
  opts.on( '-h', '--help', 'Display this screen' ) do
```

```
    puts opts
    exit
  end
end

# Parse the command-line. Remember there are two forms
# of the parse method. The 'parse' method simply parses
# ARGV, while the 'parse!' method parses ARGV and removes
# any options found there, as well as any parameters for
# the options. What's left is the list of files to resize.
optparse.parse!

puts "Being verbose" if options[:verbose]
#puts "Logging to file #{options[:logfile]}" if options[:logfile]

if options[:test]
  ln_r "LinkTests1/dirA", "LinkTests1/dirB", options
  ln_r "LinkTests2/dirA", "LinkTests2/dirB/MyCopy/Fred", options
  exit
end

if ARGV.length < 2
  puts optparse.help
  exit
end

ln_r ARGV[0], ARGV[1], options
```

You can invoke it using:

ruby ln_r.rb [options] dir1 dir2 ...

-v, --verbose Output more information
-t, --test Run test copy on test data dirs - Andy only
-r, --report Display directory of source and target dirs after finish
-c, --countsize Display bytes saved by using hard linking
-d, --dontdeletetarget Dont rm -r * target directory first
-h, --help Display this screen

Source code

Update Feb 2013

Its true that you can achieve the above in linux with

cp -lr from to

where -r means recursive copy and -l mean use linking.

Under windows, the COPY isn't so smart and so the above ruby script may be of help. Alternatively you can use a port of the linux cp command under windows - and it seems to work OK. See Port of the most important GNU utilities to Windows

Comments

Posted by Reilly on Dec 20th, 2011
The unix/linux/posix command is:

cp -al dir1 dir2

"copy [cp] with archive [a] (recursive) and link [l]"

Posted by admin on Dec 20th, 2011
I wonder if Win 7 / NTFS has similar flags on the DOS copy command?