dlFindDuplicates

Jos De Laender

1-september-2014

Inhoud

1 Glossary

- http://en.wikipedia.org/wiki/Hard_link : In computing, a hard link is a directory entry that associates a name with a file on a file system. The term is used in file systems which allow multiple hard links to be created for the same file. This has the effect of creating multiple names for the same file, causing an aliasing effect: e.g. if the file is opened by one of its names, and changes are made to its content, then these changes will also be visible when the file is opened by an alternative name.
- http://en.wikipedia.org/wiki/Lua_(programming_language) : is a lightweight multi-paradigm programming language designed as a scripting language. A Lua interpreter is embedded in dlFindDuplicates, i.e. you don't need to have it installed.

- http://en.wikipedia.org/wiki/MD5 : A MD5 checksum or hash sum is a 'signature' that is calculated over data, in the first place to protect it against unwanted alterations. In dlFindDuplicates it is used to determine a unique (*) signature of the file.
  (*) With high probability. Files can be constructed to be different and still have the same MD5.

2 Introduction

Due to copying around files on your disk, in a desperate attempt to organize them, you often end up with files that are just duplicates in different directories. Here comes dlFindDuplicates at rescue. It does so in 3 consecutive steps :

Step 1 : Search the duplicate files on selected directories. Duplicates are sieved in following order :

1. Files that are hard linked to each other are per definition the same, and no further processing power is spoiled on finding out they are equal.
2. Filesize. Files that are unequal in length cannot be the same.
3. MD5 Sum over the first megabyte of data, in order to break fast the tie on files of equal length.
4. MD5 Sum over the whole file.

Step 2 : Present each group of equal files as input to a Lua script that determines which files of the group should be selected or not. This offers the advanced user an enormous amount of flexibility in selecting which files he wants to action upon. The standard user is offered a number of predefined Lua scripts ('Select oldest', 'Select newest' …) or can select manually.

Step 3 : Action on the selected files in 3 possible ways :

1. Delete the selected files.

2. Hard link the selected files. This way the files can stay in 2 different directories (i.e. photos per year and photos per theme) yet taking only the disk space of one file. Note that hard linking is only possible on the same disk partition.
3. Move the selected files, which is rather pointless, except if you want to be very cautious and first have a second look to the moved files before ultimately deleting them.

3 Getting and installing dlFindDuplicates

3.1 Windows

Note that at the time of writing only Windows 64 bits executables are distributed !

Visit http://sourceforge.net/projects/dlfindduplicates/files and grasp the dlFindDuplicates_Installer.exe of the latest version. Install it as any other Windows program, i.e. click-click-click.

In order to fulfill the GPL license requirements, I will at the end describe how to build the Windows version, but you probably will never do that.

3.2 Linux

I'm working on XUbuntu 14.04. I assume that the description will be reasonably valid for other distributions as well. All feedback welcome.

Note : Building is only tested on a 64 bit linux machine, but there's no a priori reason why it would not work on a 32 bit machine. All feedback welcome.

3.2.1 Prerequisites

- Python 2.7.6
- gcc/g++ 4.8.1 or higher.
- Qt (development version) 4.8.6 or higher. Not Qt5 tested.
- liblua (development version) 5.2.0 or higher.
- mercurial

3.2.2 Fetching and building

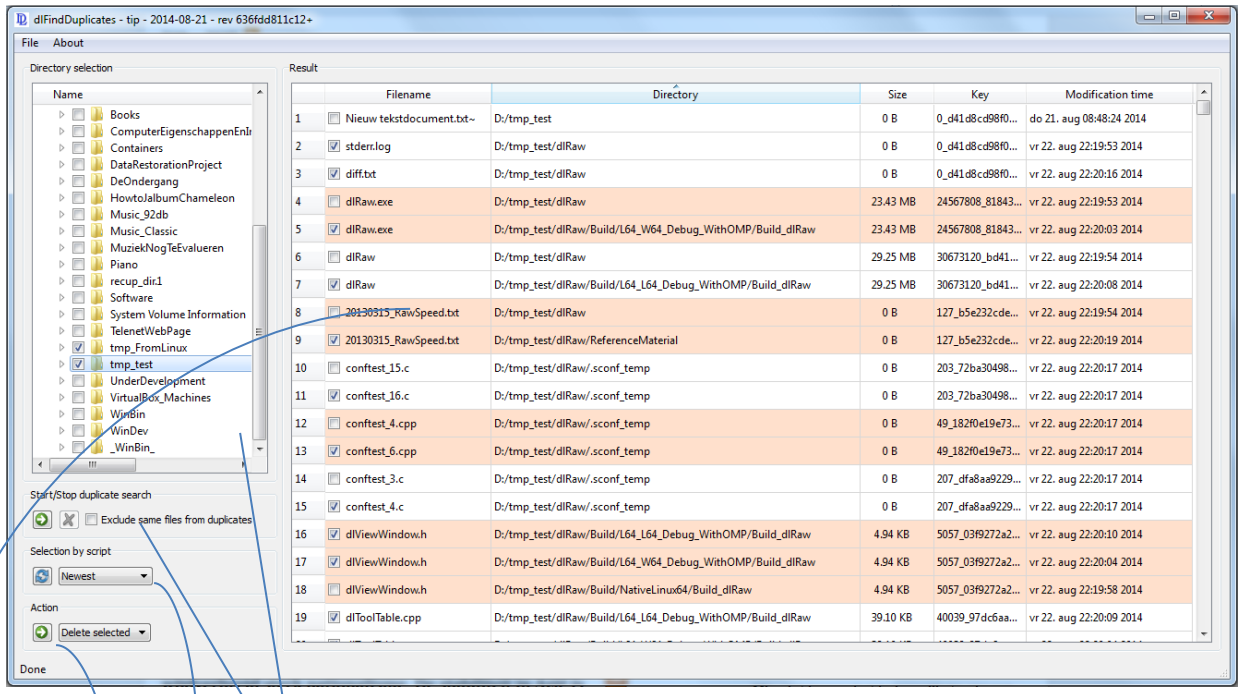
Go to an appropriate directory and issue following commands :

```
➢ hg clone http://hg.code.sf.net/p/dlfindduplicates/code dlFindDuplicates
➢ cd dlFindDuplicates
➢ ./scons.py -Q --dlBuildConfFile=BuildConfs/L64_L64_Debug_WithOMP.py
➢ sudo ./scons.py -Q --dlBuildConfFile=BuildConfs/L64_L64_Debug_WithOMP.py install
```

4 Using dlFindDuplicates

4.1 Regular use

Start it the usual way using the menu entry (with administrator / root rights in case the files you want to access require so). I will explain main functionalities via the screenshot :



- Select here the directories you want to search for duplicates
- Start (and if needed cancel) the search for duplicates. The checkbox allows you to exclude groups of files that are the same because of hard linked out of the result.
- Select one of the Lua scripts that generates a selection amongst the duplicate files. “Newest” (a standard available script) for instance would select the newest files, in order to keep the original. The button reloads the directory that contains the scripts (*) such that you can add your own scripts as well.
- Launch the indicated action (delete, move (**), link) on the selected files.
- Results found. They can be sorted on filename, directory name, size, modification time and key (***). Also you can do here manual selections on top of what the script did already for you. The alternating colors represent a group of files that are duplicates.

(*) The scripts must have ‘.lua’ extension and located under

Windows : User => UserName => .dlFindDuplicates => SelectScripts

Linux : ~/.dlFindDuplicates/SelectScripts

(**) Moving is done to :

Window : User => UserName => .dlFindDuplicates => TimeStamp_Moved

Linux : ~/.dlFindDuplicates/TimeStamp_Moved

(***) The key is of the form X_Y_Z where

X : size of the file

Y : MD5 over the first MB, or “X” in case it was not calculated.

Z : MD5 over the full file, or “X” in case it was not calculated.

4.2 Use of the script interface

In order to select within groups of equal files, a Lua scripting interface is used. The scripts must have ‘.lua’ extension and must be located under

Windows : User => UserName => .dlFindDuplicates => SelectScripts

Linux : ~/.dlFindDuplicates/SelectScripts

For the full capabilities of Lua, I will refer to www.lua.org, however the simple examples (with the explanation) will likely be a good start for everyday use.

4.2.1 Example 1 : select newest files from a group > 10MB

4.2.1.1 Code

```
--------------------------------------------------------------------------------
--
-- dlFindDuplicates
--
-- Copyright (C) 2014 Jos De Laender <jos@de-laender.be>
--
-- This file is part of dlFindDuplicates.
--
-- License : see file "License".
--
--------------------------------------------------------------------------------

Title       = "Newest > 10MB"
Description = "Select newest from group, but leave < 10MB untouched."

function Select(Files,Times,Size)
  -- Files : table of filenames.
  -- Times : corresponding table of modification times.
  -- Size  : filesize.
  local Selected = {}
  Pivot = 1
  for i=2,#Files do
    if Times[i] < Times[Pivot] then
      Pivot = i
    end
  end
  for i=1,#Files do
    Selected[i] = (i ~= Pivot) and (Size > 10485760)
  end
  return Selected
end

--------------------------------------------------------------------------------
```

4.2.1.2 Explanation

- Title and Description are mandatory global variables and that will be used to display as text and tooltip in the selection box.
- Function Select is a mandatory function and that should return the selected files as a table of Boolean values. You can add any further function or variable as you want.
- The Select function takes 2 tables and an integer as argument
    - Files is a table with the full path filenames of the files in a group of equals. Hence the size of it is the number of files in the group.
    - Times is a table with modification times, expressed in 'unix time' (basically the number of seconds since January 1, 1970) of the corresponding file in the Files table.
    - Size is an integer argument representing the size of the files (which is per definition the same for all in the group).
- The remainder is a small algorithm to select on the basis of the newest files in a group but that are larger than 1MB. You should read the Lua manual, but if you know that tables are indexed from 1 on (not 0 based !) and that #Value is the length of a table, you probably can just read it like this.
- The Select function must return a table, equal in size to the Files and Times table, and that has on each position a Boolean value that says if the file should be selected or not. In order to be valid, at least one file in the group needs to stay unselected.

4.2.2 Example 2 : Select files that contain 'Build' in the path name

4.2.2.1 Code

```
--------------------------------------------------------------------------------
--
-- dlFindDuplicates
--
-- Copyright (C) 2014 Jos De Laender <jos@de-laender.be>
--
-- This file is part of dlFindDuplicates.
--
-- License : see file "License".
--
--------------------------------------------------------------------------------

Title       = "Contains 'Build'"
Description = "Select those containing 'Buid' in path"

function Select(Files,Times,Size)
  -- Files : table of filenames.
  -- Times : corresponding table of modification times.
  -- Size  : filesize.
  local Selected = {}
  for i=1,#Files do
    Selected[i] = (nil ~= string.find(Files[i],"Build"))
```

```
  end
  AllSelected = true
  for i=1,#Files do
    if not Selected[i] then
      AllSelected = false
      break
    end
  end
  if AllSelected then
    Selected[1] = false
  end
  return Selected
end

--------------------------------------------------------------------------------
```

4.2.2.2 Explanation

This example is only added to demonstrate that you can work also on the basis of the filenames. Otherwise it just contains the same elements as the first example. Mind the code at the end to verify that at least one of the files stays unchecked.

5 Development notes

5.1 Building for Windows

This chapter is here to make sure that, in spirit of the GPL, you can build the Windows executable yourself. Normally, you would however grasp it just from the site.

Building is done using the mxe cross-compiling environment. Get the mxe code as described on the mxe site. In that directory you issue the command :

- `make MXE_TARGETS=x86_64-w64-mingw32.static`

This will build a complete cross compilation environment. Now you go to an appropriate directory and issue following commands :

- `hg clone http://hg.code.sf.net/p/dlfindduplicates/code dlFindDuplicates`
- `cd dlFindDuplicates`

Edit the file ‘Buildconfs/L64_W64_Debug_WithOMP.py’ such that the DL_MXE_PATH variable is pointing to the the mxe directory where you made the cross-build environment.

- `./scons.py -Q --dlBuildConfFile=BuildConfs/L64_W64_Debug_WithOMP.py`

This will build now the executable, as well as the installer.