***ddeedduupp* v0.1**

**User Manual**

April 2, 2016

David T. Ashley ([dashley@gmail.com](mailto:dashley@gmail.com))

Table of Contents

[1 License 3](#_Toc447454215)

[2 Introduction and Overview 3](#_Toc447454216)

[2.1 Brief Description of *ddeedduupp* Functionality 3](#_Toc447454217)

[2.2 Definitions 3](#_Toc447454218)

[2.3 Detailed Description of *ddeedduupp* Functionality 4](#_Toc447454219)

[3 Invocation Example 6](#_Toc447454220)

[4 Technical Description and Internal Operation of *ddeedduupp* 6](#_Toc447454221)

[4.1 Heap Allocation 6](#_Toc447454222)

[4.2 RAM Data Structures 6](#_Toc447454223)

[4.3 Approximate Limits on File Set 6](#_Toc447454224)

[4.4 Approximate Initialization Time 6](#_Toc447454225)

[4.5 Method of Moving Files from *tgt* to *dup* 6](#_Toc447454226)

[4.6 Use on a Changing File Set 7](#_Toc447454227)

[4.7 Use on a Networked Drive 7](#_Toc447454228)

[4.8 Redistributing *ddeedduupp* 7](#_Toc447454229)

[4.9 Rebuilding *ddeedduupp* from Source Code 7](#_Toc447454230)

[4.10 Modifying *ddeedduupp* 7](#_Toc447454231)

[4.11 Redistributing modified *ddeedduupp* 7](#_Toc447454232)

# License

# Introduction and Overview

## Brief Description of *ddeedduupp* Functionality

*ddeedduupp* is a simple *Windows* and *\*nix* tool for identifying and selectively removing duplicate files within a single directory and its subdirectories on a single disk volume. It was originally devised by the author to eliminate duplicate photos and documents.

*ddeedduupp* is written in ANSI C and should compile on virtually any platform. Full source code and a *Windows* binary are provided. For non-Windows systems, the end user will have to rebuild the program from source code.

*ddeedduupp* is designed for infrequent use, and so is not efficient if it is used frequently on large sets of files. Immediately on each invocation, *ddeedduupp* calculates the SHA-512 cryptographic hash of every file in the target directory[[1]](#footnote-1): an operation that can take hours or perhaps even days on large sets of files. After the initial calculation of the SHA-512 hashes, the operations performed by *ddeedduupp* are very quick.

*ddeedduupp* is single-threaded. (A substantial performance increase during the initial SHA-512 calculation might be possible on some platforms if a multi-threaded model were adopted, but this has not been evaluated.)

Although the possibility of two files with different contents having the same SHA-512 hash is remote[[2]](#footnote-2), *ddeedduupp* compares files before deletion and will not ultimately treat a file with the same SHA-512 hash but different contents as a duplicate.

## Definitions

On invocation, the first step performed by *ddeedduupp* is to identify the files and directories in the target directory. Internally, *ddeedduupp* forms a tree.

The notion of a tree to represent a computer file system is probably familiar to most computer users. Only directories (not files) may have children, and the children may be either files or directories.



Figure : Example Tree (from Wikipedia article https://en.wikipedia.org/wiki/Tree\_structure)

Figure 1 (copied from [this](https://en.wikipedia.org/wiki/Tree_structure) Wikipedia article) is a prototypical tree to illustrate the nomenclature for the relationships between nodes used in this document and in the output from *ddeedduupp*.

* *Node*: Each filesystem object (directory, file) is a node. *Encyclopaedia*, *Science*, *Culture*, *Art*, and *Craft* are nodes.
* *Child*, *Children*: Only a directory may have children. *Science* is a child of *Encyclopaedia*, and *Craft* is a child of *Culture*.
* *Parent*: The directory containing the directory or file. *Culture* is a parent of *Art*, and *Encyclopaedia* is a parent of *Science*.
* *Root*: A node with no parent. Only *Encyclopaedia* is the root.
* *Leaf*, *Leaf Node*: A node with no children. Every file is a leaf node (because files may not have children). Only empty directories are leaf nodes.
* *Sibling*: A node with the same parent. *Art* and *Craft* are siblings.
* *Uncle*: A sibling of a node’s parent. *Science* is an uncle of *Art*.
* *Ancestor*: A node that can be reached by traveling up the tree. *Craft* has the ancestors *Culture* and *Encyclopaedia*.
* *Descendent*: A node that can be reached by traveling down the tree. *Encyclopaedia* has all other nodes as its descendents. *Culture* has *Art* and *Craft* as its descendants.

## Detailed Description of *ddeedduupp* Functionality

*ddeedduupp* has the following behavior and provides the following functionality:

* Before invocation of *ddeedduupp*, a directory must be staged with exactly two children:
  + A subdirectory named *tgt*, containing the content to be analyzed for duplicate files. This directory may contain an arbitrary number of descendants.
  + A subdirectory named *dup*. When *ddeedduupp* removes duplicates, it places them in the *dup* directory with exactly the same relative path as they had within the *tgt* directory. The purpose of the *dup* subdirectory is to help guard against mistakes: *ddeedduupp* will only move files into the *dup* subdirectory; the user of the program has to take the final action to delete the files. Two elements of the program’s behavior should be noted:
    - *ddeedduupp* does not consider the contents of *dup* when offering deletion options to the user: the program will not delete the final instance of a file from *tgt*, even if *dup* also contains one or more instances of the file.
    - *ddeedduupp* will never under any circumstances offer the user the option of deleting the last instance of a file from *tgt*.
* *tgt* and *dup* and their descendants must be part of the same disk volume. (For the sake of speed, *ddeedduupp* moves rather than copies files from *tgt* to *dup*, and the move operation can only be performed within the same disk volume.)
* *ddeedduupp* must be invoked from the directory containing *tgt* and *dup* (i.e. at the time *ddeedduup* is invoked, the current working directory must be this directory). (There are a variety of ways to invoke *ddeedduupp* so that this constraint is met. The most common is to run *ddeedduupp* from a command prompt where the working directory is the directory containing *tgt* and *dup*.)
* Upon invocation, *ddeedduupp*:
  + Verifies that *tgt* and *dup* exist and are the only directories.
  + Scans the *tgt* directory and all descendants, and for each descendant file obtains the file size and calculates the SHA512 cryptographic hash.
  + Presents the user with information about the total number of duplicates that exist within *tgt* and the approximate space that would be saved by eliminating them.
  + Allows the user the option of navigating into directories in *tgt*. When a user navigates into a directory, information about duplicates in the directory is displayed.
  + As the user navigates into directories, the user is provided with options for the treatment of files within the directory that are duplicates (of other files within *tgt*), and for directories that contain at least one duplicate file. (The user is provided with no options for files that are not duplicates or for directories that do not contain duplicates.)
  + For files that are duplicates, the user is presented with the following options:
    - To move the file to *dup*.
    - To set the file as *globally authoritative*. All duplicate files within *tgt* that are not directory duplicates will be moved to *dup*.
    - To set the file as *locally authoritative*. All directory duplicates will be moved to *dup*.
    - To set the file as *ancestrally authoritative*. All duplicate descendants will be moved to *dup*.
    - To set the file as *sibling authoritative*. All duplicates within the siblings of the directory containing the file are moved to *dup*.
    - To set the file as *sibling ancestrally authoritative*. All duplicates within the siblings of the directory containing the file, as well as within the descendants of those siblings, are moved to *dup*.
  + For directories containing directory duplicates or with siblings that contain directory duplicates, *ddeedduupp* offers the following options:
    - *Local removal*: resolve directory duplicates only within the directory being manipulated.
    - *Local and descendant removal*: resolve directory duplicates in the directory being manipulated and all its descendants. (*Note:* if this option is chosen when manipulating the *tgt* directory, the effect is global directory duplicate removal.)
    - *Sibling removal*: resolve directory duplicates in the siblings of the directory being manipulated.
    - *Sibling and descendant removal:* resolve directory duplicates in the siblings of the directory being manipulated, and in the descendants of those siblings.
  + For directories containing duplicates that are not directory duplicates, *ddeedduupp* offers the following options:
    - *A*ll duplicates in the directory are moved to *dup*.
    - To set the directory as *globally authoritative*. All files outside the directory that are duplicates of any of those in the directory are removed.
    - To set the directory as *ancestrally authoritative*. All files in the descendants of the directory that are duplicates of those in the directory are removed.
    - To set the directory as sibling authoritative. All files in the siblings of the directory that are duplicates of those in the directory are removed.
    - To set the directory as *sibling ancestrally authoritative*. All files in the siblings of the directory containing the file that are duplicates of the files in the directory are removed, as well as duplicates within the descendants of those siblings.

# Invocation Example

# Technical Description and Internal Operation of *ddeedduupp*

## Heap Allocation

## RAM Data Structures

## Approximate Limits on File Set

## Approximate Initialization Time

## Method of Moving Files from *tgt* to *dup*

## Use on a Changing File Set

## Use on a Networked Drive

## Redistributing *ddeedduupp*

## Rebuilding *ddeedduupp* from Source Code

## Modifying *ddeedduupp*

## Redistributing modified *ddeedduupp*

1. Calculating the SHA-512 cryptographic hash of every file allows *ddeedduupp* to identify duplicates very quickly. [↑](#footnote-ref-1)
2. *Remote* is probably not a strong enough word. No SHA-512 hash collisions have *ever* been identified, and 2512 is approximately 10154. For comparison, the number of atoms in the observable universe is estimated to be 1082, a *much* smaller number. [↑](#footnote-ref-2)