***ddeedduupp* v0.1a**

**User Manual**

April 2, 2016

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

Table of Contents

[1 License 3](#_Toc27927211)

[2 Introduction and Overview 3](#_Toc27927212)

[2.1 Brief Description of *ddeedduupp* Functionality 3](#_Toc27927213)

[2.2 Definitions 3](#_Toc27927214)

[2.3 Detailed Description of *ddeedduupp* Functionality 4](#_Toc27927215)

[2.3.1 Invocation 4](#_Toc27927216)

[2.3.2 General Notes and Cautions 5](#_Toc27927217)

[2.3.3 Issuing Commands to Remove Duplicates 5](#_Toc27927218)

[2.3.4 Commands that Exit the Program 5](#_Toc27927219)

[3 Invocation Example 8](#_Toc27927220)

[4 Technical Description and Internal Operation of *ddeedduupp* 8](#_Toc27927221)

[4.1 Heap Allocation 8](#_Toc27927222)

[4.2 RAM Data Structures 8](#_Toc27927223)

[4.3 Approximate Limits on File Set 8](#_Toc27927224)

[4.4 Approximate Initialization Time 8](#_Toc27927225)

[4.5 Method of Moving Files from *tgt* to *dup* 8](#_Toc27927226)

[4.6 Use on a Changing File Set 8](#_Toc27927227)

[4.7 Use on a Networked Drive 8](#_Toc27927228)

[4.8 Redistributing *ddeedduupp* 8](#_Toc27927229)

[4.9 Rebuilding *ddeedduupp* from Source Code 9](#_Toc27927230)

[4.10 Modifying *ddeedduupp* 9](#_Toc27927231)

[4.11 Redistributing modified *ddeedduupp* 9](#_Toc27927232)

# License

# Introduction and Overview

## Brief Description of *ddeedduupp* Functionality

*ddeedduupp* is a simple *Windows* and *\*nix* tool for identifying and selectively removing duplicate files. 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* requires that the pool of files from which duplicates are to be selectively removed are staged in a single directory on a single disk volume.

*ddeedduupp* supports only ASCII characters in file names, and does not support Unicode. It is not known how the program will behave on operating systems where file names may contain non-ASCII characters.

*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.

Unlike other more sophisticated deduplication programs, *ddeedduupp* has no notion of files which are nearly identical. *ddeedduupp* will identify and remove only files which are exactly identical.

## 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 1: 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

### General Notes and Cautions

1. On \*nix systems, *ddeedduupp* does not follow symlinks (“soft” links). However, it does follow “hard” links.
2. *ddeedduupp* makes the assumption that it is the only program modifying the filesystem of the target directory. Any modification of the filesystem of the target directory while *ddeedduupp* is running may have unpredictable results, including loss of data.
3. *ddeedduupp* commands that remove duplicates are not undoable. The duplicate files are deleted permanently.
4. *ddeedduupp* will not remove the last file of a set of duplicates (such an operation must be done manually, using some method other than *ddeedduupp*).

### Invocation

From a shell or DOS command prompt, *cd* (change directory) to the working directory in which deduplication operations are to be performed. Then, run the program by typing *ddeedduupp*.

At the present time, *ddeedduupp* supports no command-line options.

Immediately on invocation, *ddeedduupp* will identify all files and directories that are descendants of the current working directory, then calculate the SHA-512 cryptographic hash of every file. This operation may take in some cases hours or days.

After *ddeedduupp* has calculated the SHA-512 cryptographic hash of every file, it will display a command prompt and accept commands to delete duplicates.

### General Operation, Current Reference Directory, and Current Reference Set

*ddeedduupp* divides the set of files on which it is invoked into two sets:

* The current reference set.
* All other files.

Note that since the sets above are mutually exclusive and mutually exhaustive, identifying the current reference set also uniquely identifies the set of all other files.

When a command is provided to *ddeedduupp*, the information provided is:

* The identity of the current reference set.
* Which operation to perform.
* Options which modify the operation.

### Current Reference Directory

Current Reference Set

### Issuing Commands to Remove Duplicates

*ddeedduupp* commands are divided into five categories:

* Commands that exit the program.
* Commands that display information.
* Commands that change the current reference directory.
* Commands that remove duplicates with respect to files.
* Commands that remove duplicates with respect to directories.

Each command must be followed by the *ENTER* key.

Each of these families of commands is described in the following sections.

The general format of each command issued is:

*command* [*options*] [*--*] [*targets*]

where

* *command* is the issued command.
* *options* are options to modify the command.
* *--* is an optional separator.
* *targets* is the list of targets. Lists and ranges are supported, i.e. “1-3 4 29 9-12”

### Command Options

Table 1: Command Options

|  |  |  |
| --- | --- | --- |
| **Command Option Number** | **Command** | **Action** |
| 1 | *-interactive* | Causes the user to be prompted for the deletion of each file. |
| 2 | *-dryrun* | Causes the command to be executed with no effect on the file system. |
| 3 | *-notouch\_crd* | Causes no duplicates to be removed from the current reference directory. |
| 4 | *-notouch\_crd\_descendants* | Causes no duplicates to removed from the descendants of the current reference directory. |
| 5 | *-setinclude\_crd* | Causes the current reference directory to be included in the current reference set.  This option is at this time never necessary, because the current reference directory is always included by default. |
| 6 | *-setexclude\_crd* | Causes the current reference directory to be excluded from the current reference set. |
| 7 | *-setinclude\_crd\_descendants* | Causes the descendants of the current reference directory to be included in the current reference set. |
| 8 | *-setexclude\_crd\_descendants* | Causes the descendants of the current reference directory to be excluded from the current reference set. |
| 9 | *-setinclude\_crd\_ancestors* | Causes the ancestors of the current reference directory to be included in the current reference set. |
| 10 | *-setexclude\_crd\_ancestors* | Causes the ancestors of the current reference directory to be excluded from the current reference set. |
| 11 | *-setinclude\_crd\_siblings* | Causes the siblings of the current reference directory to be included in the current reference set. |
| 12 | *-setexclude\_crd\_siblings* | Causes the siblings of the current reference directory to be excluded from the current reference set. |

### Commands that Exit the Program

Any of the commands listed in Table 2 can be used to exit *ddeedduupp*.

Table 2: Commands that Exit ddeedduupp

|  |  |  |
| --- | --- | --- |
| **Command Number** | **Command** | **Action** |
| 1 | *q* | Exits *ddeedduupp*. |
| 2 | *quit* | Same as *q*, above. |
| 3 | *exit* | Same as *q*, above. |
| 4 | *stop* | Same as *q*, above. |
| 5 | *abort* | Same as *q*, above. |
| 6 | *halt* | Same as *q*, above. |
| 7 | *^C (break)* | If *ddeedduupp* is performing a long-running operation (calculating SHA-512 hashes, for example), will cause an orderly exit as soon as possible.  At the *ddeedduupp* prompt, will cause an orderly exit immediately. |

#### Commands that Display Information

Commands that display information are described in Table 3.

Table 3: Commands that Display Information

|  |  |  |
| --- | --- | --- |
| **Command Number** | **Command** | **Action** |
| 8 | *help* | Displays help and general information about the *ddeedduupp* program. |
| 9 | *?* | Displays help about the current context, including the operations available and the number of duplicates present in the current reference directory and its descendants. |
| 10 | *ls* | Displays information about the current reference directory, including about the parent and children. This is typically used before changing the reference directory. |
| 11 | *dir* | Same as *ls*, immediately above. |

#### Commands that Change the Current Reference Directory

Commands that change the current reference directory are described in Table 4.

Table 4: Commands the Change the Current Reference Directory

|  |  |  |
| --- | --- | --- |
| **Command Number** | **Command** | **Action** |
| 12 | *cd /* | Changes the reference directory to the root directory on which *ddeedduupp* was invoked. |
| 13 | *cd ..* | Changes the reference directory to the parent directory. It is not possible to ascend above the root directory on which *ddeedduupp* was invoked. |
| 14 | *cd n* | Changes the reference directory to the child of the reference directory identified with the integer *n*.  “ls” (see Table 3) can be used to display a list of all the child directories with which “cd n” may be used, along with the corresponding integers. |

#### Commands that Remove Duplicates with Respect to Files

Option -includedescendants

Action restrictions:

* -notouch\_cwd
* -notouch\_descendants

Set membership modifications:

* -setinclude\_descendants

rdf\_pri <filespec>

rdf\_sec

rdd\_pri

rdd\_sec

#### Commands that Remove Duplicates with Respect to Directories

#### Commands that Remove Duplicate Files

The commands that remove duplicate files are described in Table 5.

Commands that remove duplicate files can be categorized along the following orthogonal axes:

* Which set of files is specified. Possibilities are:
  + Files in the current ***r***eference ***d***irectory ***o***nly (mnemonic: *rdo*).
  + Files in the current ***r***eference ***d***irectory and all its ***d***escendants (mnemonic: *rdd*).
  + Files in the current ***r***eference ***d***irectory and all its ***a***ncestors (mnemonic: *rda*).
  + Files in the ***d***escendants [***on***ly] of the current reference directory (mnemonic: *don*).
  + Files in the ***a***ncestors [***on***ly] of the current reference directory (mnemonic: *aon*).
* What operation should be performed. Possibilities are:
  + ***Pri***mary (mnemonic: *pri*): Every file that is outside the set that is a duplicate of at least one file within the set will be deleted. No file within the set will be deleted.
  + ***Sec***ondary (mnemonic: *sec*): Every file that is within the set that is a duplicate of at least one outside the set will be deleted. No file outside the set will be deleted.
  + ***Inn***er (mnemonic: *inn*): Within the set, all but one of each duplicate file will be deleted.[[3]](#footnote-3) No file outside the set will be deleted.
  + ***Out***er (mnemonic: *out*): Outside of the set, all but one of each duplicate file will be deleted.3 No file within the set will be deleted.
* Interactivity. Possibilities are:
  + Prompt the user about the deletion of each duplicate file. (Mnemonic for prompting: *int*.)
  + Perform a dry run only so that the user can determine what *ddeedduupp* would actually do. (Mnemonic for dry run: *dryrun*.)

Each of the commands in Table 5 is a concatenation of “rd” with the mnemonic for the selection along each of the orthogonal axes identified above (40 commands in all).

Table 5: Commands that Remove Duplicate Files

# 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)
3. There is unfortunately no provision for the user to specify which of the duplicates is to be preserved, or to allow more than one duplicate to be preserved. Some types of desired behavior can be obtained by changing the set or by performing operations interactively. [↑](#footnote-ref-3)