Question

Imagine that we have 2\*\*48 text files. Explain how can we find which files are the same.

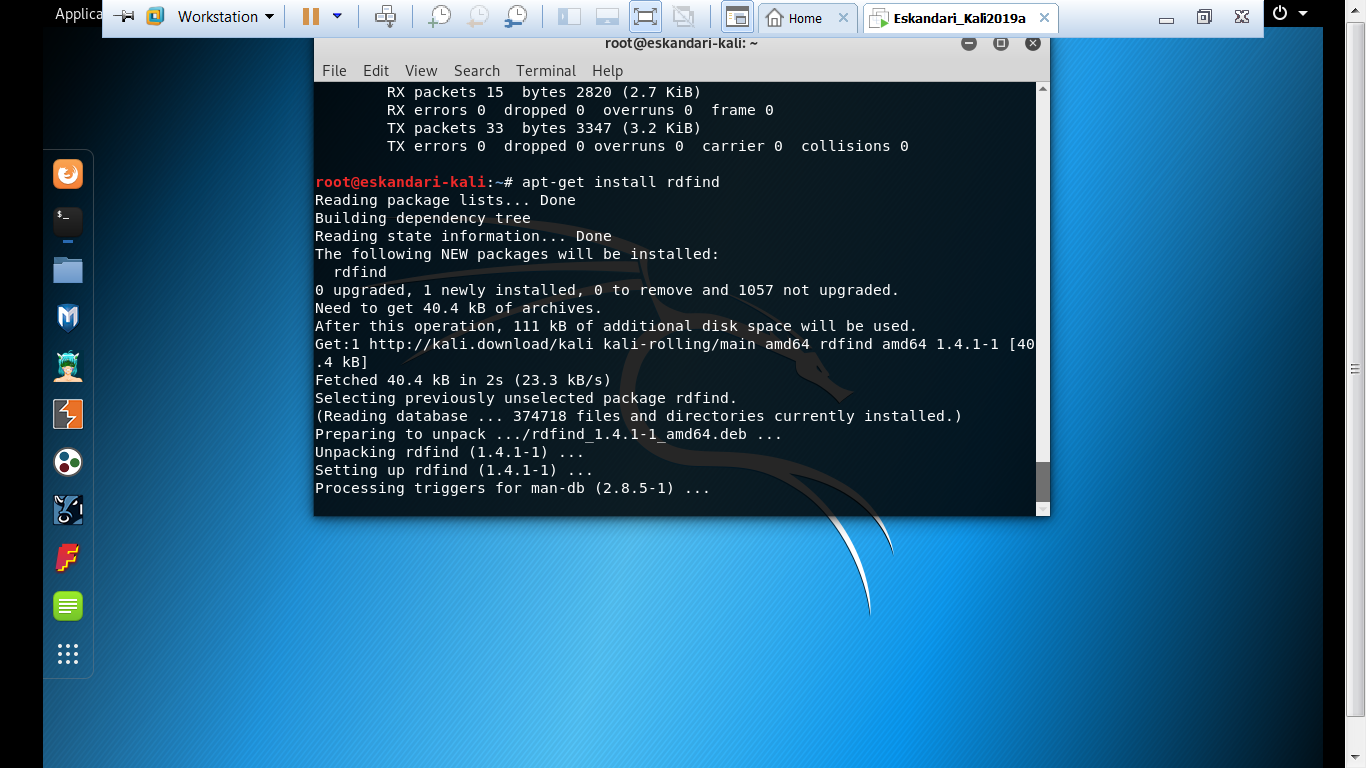
**Method 1-Short Answer**

Rdfind comes from redundant data find. It is a free tool used to find duplicate files across or within multiple directories. It uses checksum and find duplicates based on file contains not only names.

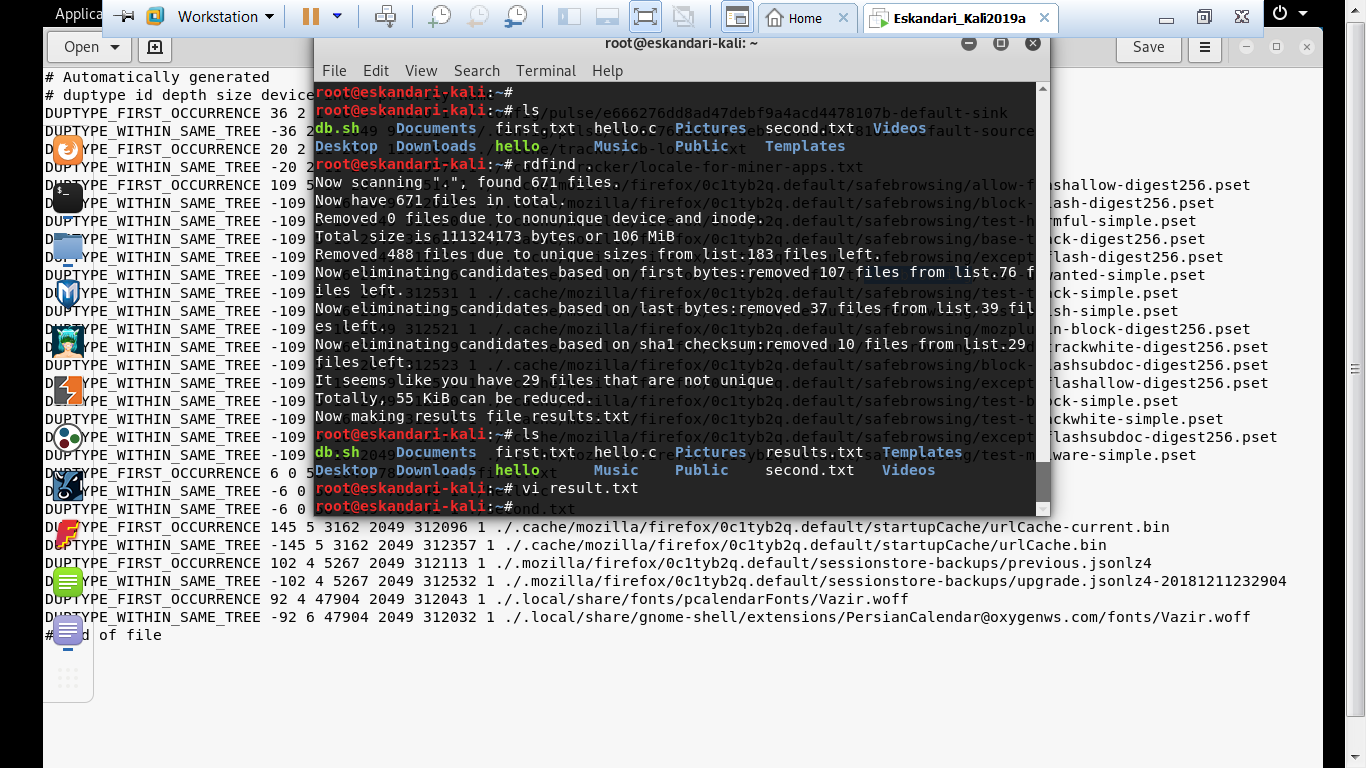
Rdfind uses algorithm to classify the files and detects which of the duplicates is the original file and considers the rest as duplicates.

**Method 1-Long Answer**

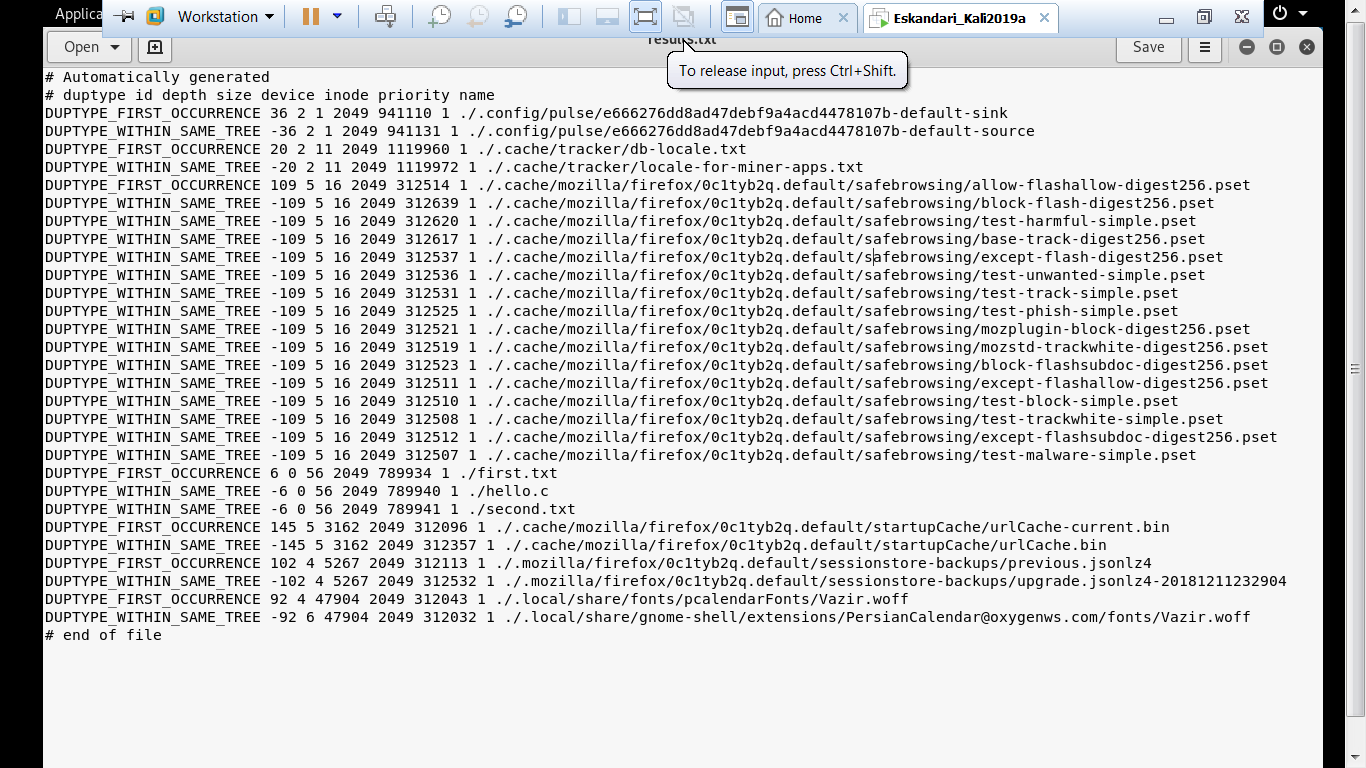
1. install **rdfind** in Linux



Now, we run **rdfind** on a directory, by simply typing **rdfind** and the target directory. Here is an example:



As you can see rdfind will save the results in file called results.txt located in the same directory from where you ran the program. The file contains all the duplicate files that rdfind has found.



Method 2- Short Answer:

We should store the hash value of each file in a database and then query the database for duplicate hash values. It results in duplicate files along with some other distinct files which their hash values collide with each other. Then, we could prune the result to remove collision cases.

Method 2- Long Answer:

1. Make a table in a database, for example, table Files\_Info with three columns (ID, Path , Hash)
2. for each file, first we should find its hash value (for example, by running one of the MD5 or SHA1 hash functions), and then Insert file path and its Hash to the above table. SHA1 is preferred since the probability of collision occurrence is less than MD5.

Commands in Linux:

For example, you can download coreutil via commands below: (it is a library which contains most of hash functions)

|  |
| --- |
| % dpkg -L coreutils | grep '[0-9]sum$' |

Now, you have these hash functions:

|  |
| --- |
| /usr/bin/sha224sum  /usr/bin/sha512sum  /usr/bin/md5sum  /usr/bin/sha1sum  /usr/bin/sha256sum  /usr/bin/sha384sum |

Choose one of them to hash files. We prefer the sha1sum.

Step 2, make the before mentioned table in MySQL and then use the following bash to store the hash value of each file in the table.

|  |
| --- |
| #!/bin/bash  DB\_USER='razieh';  DB\_PASSWD='eskandari';  DB\_NAME='test';  TABLE='Files\_Info'  for filename in /Directory/\*.txt; do  Hash=` sha1sum ${filename}`  mysql --user=$DB\_USER --password=$DB\_PASSWD $DB\_NAME << EOF  INSERT INTO $TABLE (id, Path, hash) VALUES (NULL, "$filename", "$Hash");EOF  done |

1. Step 3: finally, running this query in MySQL would return the duplicate files:

**SELECT** Hash,**COUNT**(Hash)

**FROM** Files\_Info

**GROUP BY** Hash

**HAVING COUNT**(Hash) > 1;

1. Step 4: since it is probable that the above hash function could have a collision among 2^24 hash values, we should elaborate more on results. So, we check files obtained in step 3 once again to ensure duplication. Some of them may be eliminated in this step.

The algorithm used for this step is not very important, because the number of candidate files is not too many.