Deduplication Meets Innovation

Kellen Haas and Patrick "Bart" Johnson
CPSC 4240

ABSTRACT

Our project, and in turn, this paper focus on file systems and the data stored in these systems. Storage availability and free storage space impacts everyone at every level of life. The script designed and outlined in this document is designed to make a system administrator's life easier by finding and identifying the duplicate files, and then storing them in a secondary location in the filesystem to be deleted (or kept). Convenience was the main goal of this project, allowing system administrators the ability to contain all duplicate files stored on the filesystem in one easily accessible location. The system administrator can then go through that secondary location and remove or keep the files listed as duplicates.

This program was designed and created because everyone suffers from storage shortages due to duplication of files. Sometimes this duplication is on purpose, but for the most part it is usually accidental. A system administrator needs to tackle the foreboding task of cleaning up the filesystem before it becomes a big problem creating a bottleneck in the filesystem. A helpful script would largely benefit system administrators helping them find the duplicate files and then decide the fate of those files.

1 INTRODUCTION

As a system administrator, file systems give us a place to store all of our files and data. However, this virtual storage space has a dark side, and duplicate files love to hide in the shadows. If the storage on a device fills up, it is up to the system administrator to free up that storage or upgrade the company to a larger storage plan. Whenever someone runs out of storage, they usually try to find the largest items that can be deleted to free up the most storage with the least amount of effort required on their end. We have been conditioned to want the biggest impact the fastest. If there was a script that could be run that does a massive impact all at once that means the bigger applications and files that would be deleted to just make room get to remain possibly useful. We started with a homemade approach to creating our own script, but quickly realized that with the time frame being so short it would be best to modify existing code that is free for download online and documented later in this paper. After we got the code we modified it majorly. One of the most important changes made was where instead of deleting the files the script moved them to a directory that the script makes (if it is non existent.) This way the system administrator can go through all the duplicate files in one place to decide if there was any information worth salvaging on them or whatever his reason would be for keeping them. Since we create the directory every time it is run they could delete it without reading it anytime they wish. Overall all of our initial tests were good and if we had the time of a regular semester this code would be very cool and definitely something anyone would be able to use.

2 BACKGROUND

In research for this idea of date duplication and the need for deduplication to take place it stands to reason that we must understand why this is happening and could it actually cause the problems we think it could cause. Looking into this more and doing research we found this quote from TechChannel, "Information Technology Intelligence Corp.'s (ITIC) 2018 Global Server Hardware, Server OS Reliability Survey (ibm.co/2LCX6gz), which surveyed more than 800 customers worldwide, found that 59 percent of respondents cited human error as the No. 1 cause of unplanned downtime" [3]. Knowing duplication of files can have a severe impact on a company, as a system administrator we need to always remain aware of even the smallest issues because they can lead to far bigger issues in the future.

Some more research revealed this big idea and thought of how duplicate files actually are to a big company. The article was listing all of the reasons duplicate data was bad, but one of the big ideas was that the company was paying to store the same data twice [1]. This article illuminated the potential money this could be costing a company for wasted storage. The bigger the company the more employees that can potentially waste chunks of storage for duplicate files or data. We know duplication is happening and we are aware that duplicates are costing companies money and storage space to keep, so how do we get rid of all the duplicates?

We found an article that describes a simple, yet powerful way to take care of the duplicate data problem that they like to call data deduplication. It is introduced as a solution to data duplication caused by human error. According to Tye,

> "To counteract the problems outlined, the solution is a process called, unsurprisingly, data deduplication. This is a blend of human insight, data processing and algorithms to help identify potential duplicates based on likelihood scores and common sense to identify where records look like a close match. This process involves analysing your database to identify potential duplicate records, and unravelling these to identify definite duplicates" [2].

This suggests the exact method we are using and goes even further providing more insight with regards to the necessity of sorting duplicate files for easier cleanup. Similarly, Tye states, "Deduplication rules also need to be implemented, based on your own unique data issues, in order to create a bespoke deduplication strategy. The rules should take into account your decisions about how 'strict' you want to be with your deduplication, in terms of maintaining the balance between losing valuable customer data and having a clean database" [2]. The idea to create a temporary archive in the "tmp" directory, allowing the system administrator to parse through the duplicates at a time convenient to them, stemmed largely from Tye's article.

3 MOTIVATION AND OBJECTIVES

We are trying to simply create a way to delete duplicate files and clutter to make a user's life and experience easier. At first it was just creating something completely from scratch but due to time constraints we needed to just modify something someone already made and just enhance it to make it better for our goals. As we saw earlier in the research it is really important to have the system administrator make the final call on the files to be kept and the files to be deleted. This is why we created the temp directory that will store the duplicate files where the sys admin can come through and delete what they see fit. This needs to be created to stop the storage problem that if kept unchecked will cause companies storage bills to continuously increase.

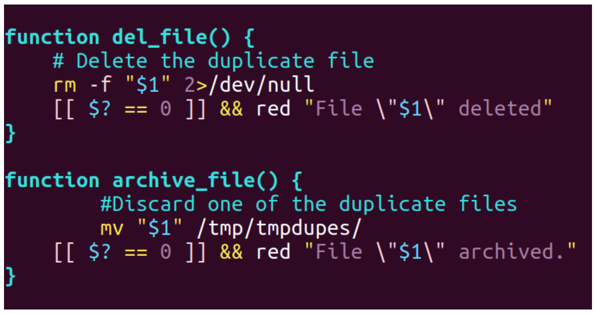
4 METHODOLOGY AND DESIGN

Originally, we attempted to write our own bash script from scratch to accomplish the goal of removing duplicate files. We discovered that our original idea was a bit ambitious given the small time frame of a summer class. The first bash script utilized the function "fdupes" to find duplicate files inside of a single directory. The function did work and would then send the file path of the pairs of duplicate files to a text document sorted by storage size. This was a decent place to start, but we were unable to completely figure out how to then remove or move the files once they were identified. Being unfamiliar with writing bash scripts and bash script syntax, we decided to pivot in a different direction. We decided to utilize a bash script that finds and deletes duplicate files as a sort of template [5].

4.1 Overview of the Design

The following screenshot contains two functions:

- function del_file() - original function from Thomas' article
- function archive_file() - added function that we created

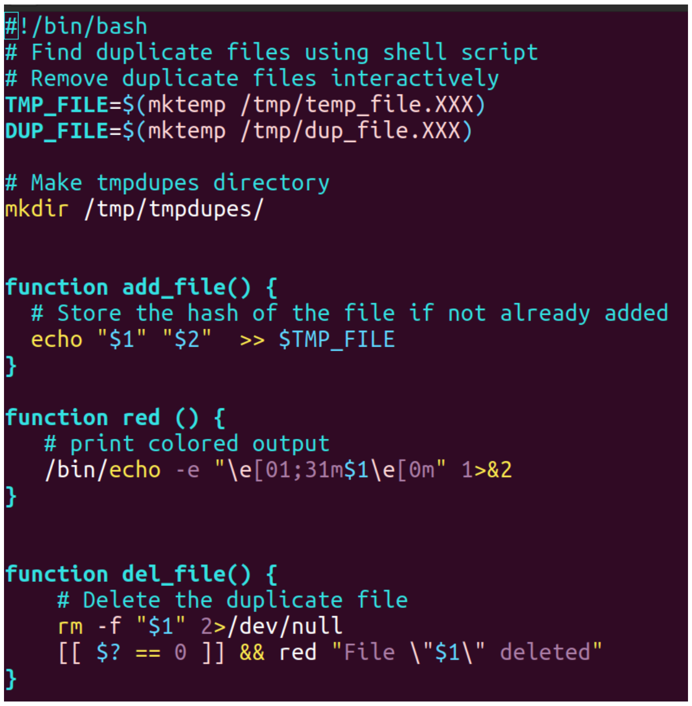


```
function del_file() {
    # Delete the duplicate file
    rm -f "$1" 2>/dev/null
    [[ $? == 0 ]] && red "File \"$1\" deleted"
}

function archive_file() {
        #Discard one of the duplicate files
        mv "$1" /tmp/tmpdupes/
    [[ $? == 0 ]] && red "File \"$1\" archived."
}
```

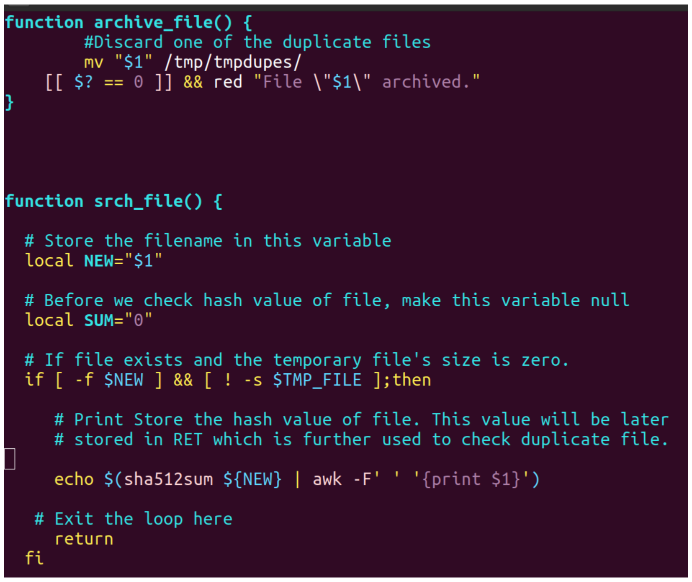
These functions and screenshot are part of the file we modified, delete was already there and we added archive. This is where we were able to edit the file and make necessary changes to ask for correct user input, and then added new functionality to move duplicate files to an archive in the system's "tmp" directory, allowing a system administrator to then go back through the archived files and delete specific ones. The system administrator would also have the option to just delete the entire archive in the "tmp" directory if they know there is nothing that they potentially need to keep. The code itself uses the "awk" command to scan the files in the directories specified and find matches for a pattern. In this project matching patterns would be having the same file name or the same contents. The program utilizes the "find" command and scans the SHA512 checksums of the directories. It also uses the hash values of the files to find duplicates.

Ideal functionality would have included the ability to search an entire drive or filesystem all at once, but this will require a fairly large amount of work. With the option to search the entire filesystem, the system administrator would also be able to create a list of directories or specific files that they do not want removed or archived. This additional step would be necessary if the script is going to be automated.

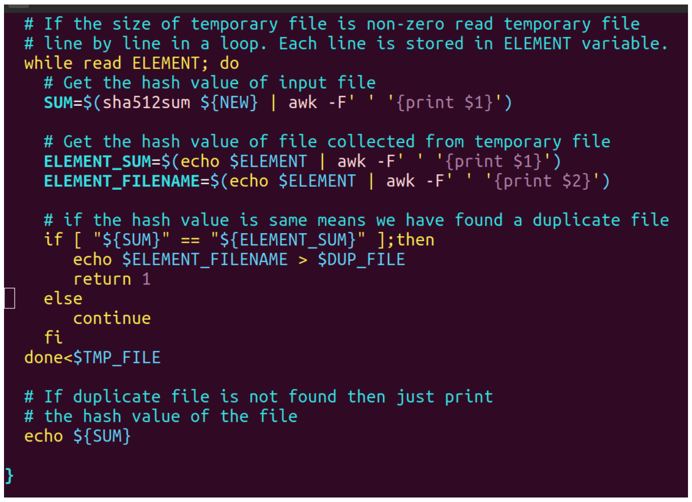
5 ANALYSIS AND RESULTS



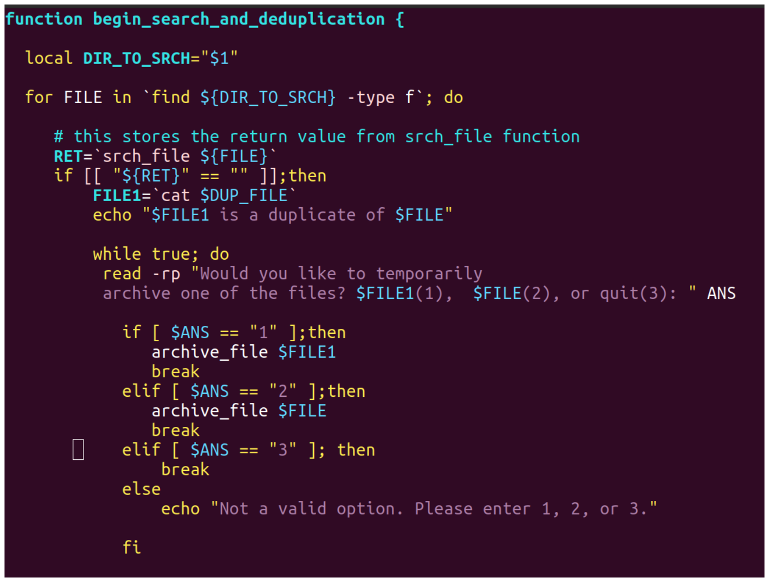
```
#!/bin/bash
# Find duplicate files using shell script
# Remove duplicate files interactively
TMP_FILE=$(mktemp /tmp/temp_file.XXX)
DUP_FILE=$(mktemp /tmp/dup_file.XXX)

# Make tmpdupes directory
mkdir /tmp/tmpdupes/


function add_file() {
  # Store the hash of the file if not already added
  echo "$1" "$2"  >> $TMP_FILE
}

function red () {
   # print colored output
   /bin/echo -e "\e[01;31m$1\e[0m" 1>&2
}


function del_file() {
    # Delete the duplicate file
    rm -f "$1" 2>/dev/null
    [[ $? == 0 ]] && red "File \"$1\" deleted"
}
```



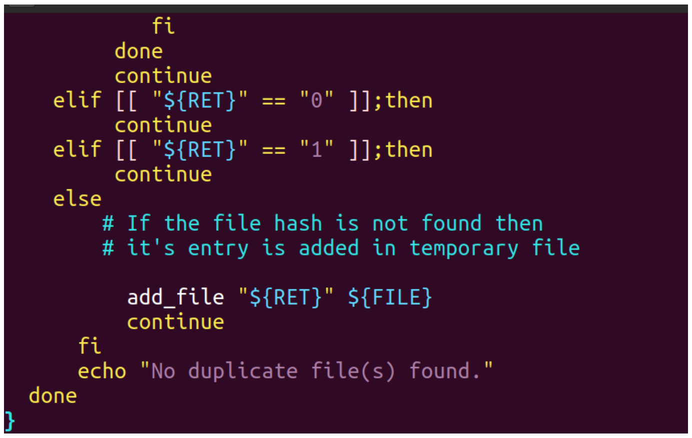
```
function archive_file() {
        #Discard one of the duplicate files
        mv "$1" /tmp/tmpdupes/
    [[ $? == 0 ]] && red "File \"$1\" archived."
}




function srch_file() {

  # Store the filename in this variable
  local NEW="$1"

  # Before we check hash value of file, make this variable null
  local SUM="0"

  # If file exists and the temporary file's size is zero.
  if [ -f $NEW ] && [ ! -s $TMP_FILE ];then

     # Print Store the hash value of file. This value will be later
     # stored in RET which is further used to check duplicate file.

     echo $(sha512sum ${NEW} | awk -F' ' '{print $1}')

   # Exit the loop here
     return
  fi
```



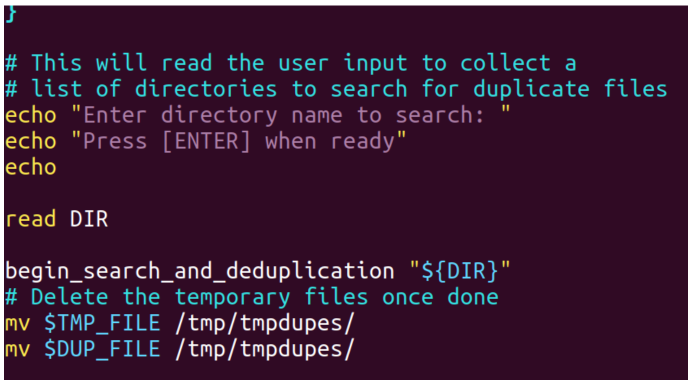
```
  # If the size of temporary file is non-zero read temporary file
  # line by line in a loop. Each line is stored in ELEMENT variable.
  while read ELEMENT; do
    # Get the hash value of input file
    SUM=$(sha512sum ${NEW} | awk -F' ' '{print $1}')

    # Get the hash value of file collected from temporary file
    ELEMENT_SUM=$(echo $ELEMENT | awk -F' ' '{print $1}')
    ELEMENT_FILENAME=$(echo $ELEMENT | awk -F' ' '{print $2}')

    # if the hash value is same means we have found a duplicate file
    if [ "${SUM}" == "${ELEMENT_SUM}" ];then
       echo $ELEMENT_FILENAME > $DUP_FILE
       return 1
    else
       continue
    fi
  done<$TMP_FILE

  # If duplicate file is not found then just print
  # the hash value of the file
  echo ${SUM}

}
```



```
function begin_search_and_deduplication {

  local DIR_TO_SRCH="$1"

  for FILE in `find ${DIR_TO_SRCH} -type f`; do

     # this stores the return value from srch_file function
     RET=`srch_file ${FILE}`
     if [[ "${RET}" == "" ]];then
         FILE1=`cat $DUP_FILE`
         echo "$FILE1 is a duplicate of $FILE"

         while true; do
          read -rp "Would you like to temporarily
          archive one of the files? $FILE1(1),  $FILE(2), or quit(3): " ANS

            if [ $ANS == "1" ];then
               archive_file $FILE1
               break
            elif [ $ANS == "2" ];then
               archive_file $FILE
               break
            elif [ $ANS == "3" ]; then
                break
            else
                echo "Not a valid option. Please enter 1, 2, or 3."

            fi
```



```
            fi
         done
         continue
    elif [[ "${RET}" == "0" ]];then
         continue
    elif [[ "${RET}" == "1" ]];then
         continue
    else
        # If the file hash is not found then
        # it's entry is added in temporary file

          add_file "${RET}" ${FILE}
          continue
      fi
      echo "No duplicate file(s) found."
  done
}
```



```
}

# This will read the user input to collect a
# list of directories to search for duplicate files
echo "Enter directory name to search: "
echo "Press [ENTER] when ready"
echo

read DIR

begin_search_and_deduplication "${DIR}"
# Delete the temporary files once done
mv $TMP_FILE /tmp/tmpdupes/
mv $DUP_FILE /tmp/tmpdupes/
```

6 CONCLUSION AND FUTURE WORK

We are able to find the duplicate files and move them to another directory for a system admin to look at. This means our goals have been reached and we are successful. Even though it is just two directories it can grow to be bigger. A big takeaway practically for us is that bash scripting is a lot harder than we thought. Our last article also has to do with future advancement of this project. It has a good suggestion to take into consideration when data deduplication is occurring and you are deleting all the duplicates through an algorithm. "..before you go the automated route, consider a safer option: Set your duplicate file finder to ignore files smaller than 20MB. That way, you'll have far fewer files to worry about, yet still free up a lot of space" [4]. Sorting based on size is something we are definitely interested in mostly where the user can customize and set the size they want to be notified if it is above. Another idea this sparked was a directory blacklist that will always prompt before moving or deleting that way directories that need to never be touched such as /etc, /bin/, and /sbin will never be touched without the system administrator's permission.

REFERENCES

1. Anon. 2017. Top 5 Reasons Why Duplicate Data is Bad for Business. *This is why duplicate data is bad for you* (2017).
2. George Tye. 2015. Why duplicate records are costly for your company - and what can be done about it. (August 2015). Retrieved July 18, 2021 from https://www.mycustomer.com/marketing/data/why-duplicate-records-are-costly-for-your-company-and-what-can-be-done-about-it
3. Laura DiDio. 2018. Human Error is Top Cause of Downtime. (October 2018). Retrieved July 18, 2021 from https://techchannel.com/SMB/10/2018/Human-Error-Top-Cause-of-Downtime
4. Lincoln Spector. 2013. Automatically delete a huge amount of duplicate files. (June 2013). Retrieved July 21, 2021 from https://www.pcworld.com/article/2039794/automatically-delete-a-huge-amount-of-duplicate-files.html
5. Thomas. 2020. How to find and remove duplicate files using shell script in Linux. (January 2020). Retrieved July 21, 2021 from https://www.golinuxcloud.com/find-remove-duplicate-files-hash-shell-script/