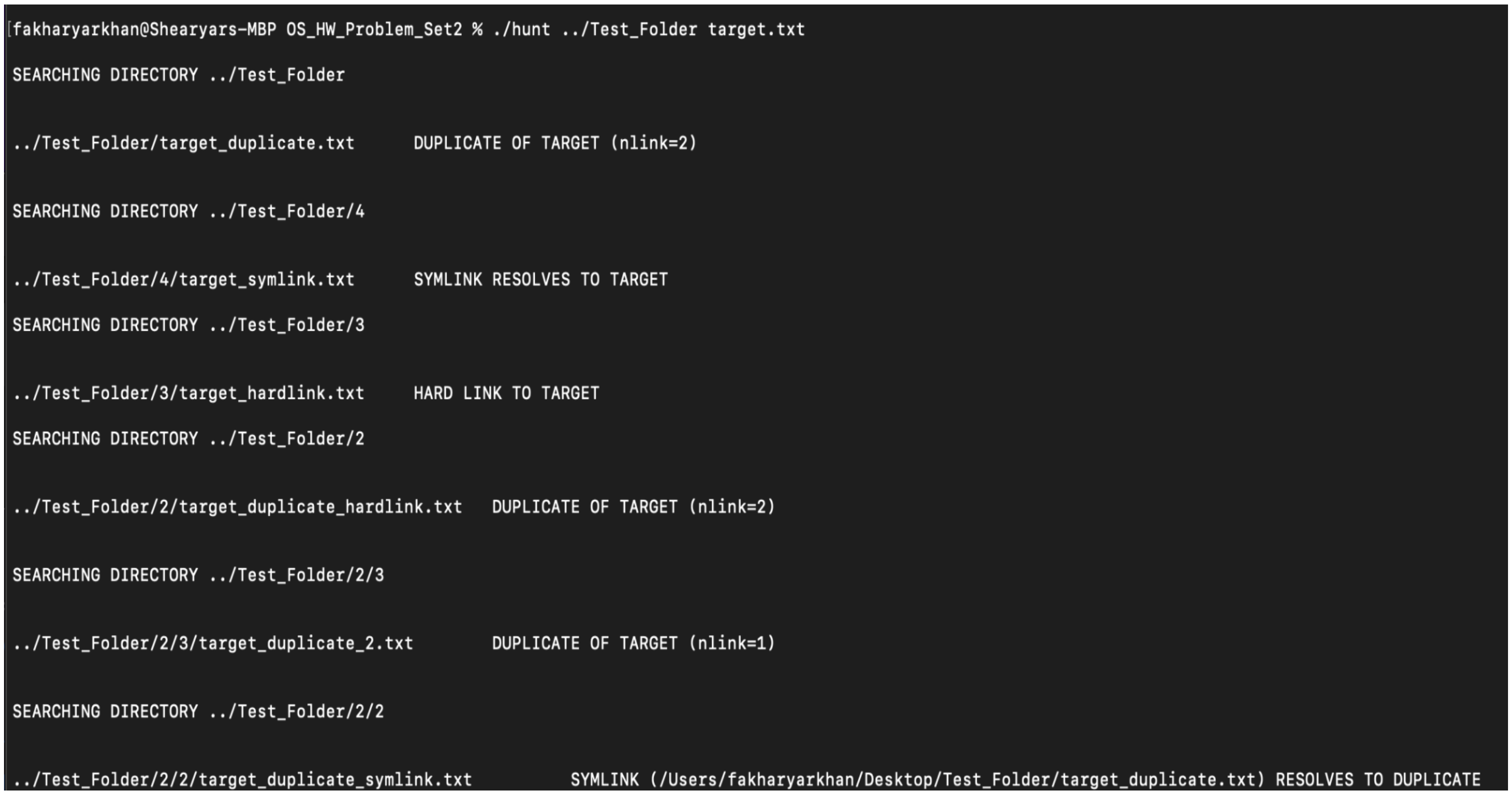
Hengnan Ma, Fakharyar Khan
Operating Systems
Professor Hakner
September 26th, 2023

Project #2: File Scavenger Hunt README

How it works: Suppose the executable file is named hunt. To run the program, use the command: ./hunt directory targetfile, where the directory is the path to the directory you want the program to search in and the target file is the name of a file in the current directory. Hunt will search through the specified directory and report back the names of all files that are byte for byte equal to the target file. It will also state if the file being searched is a duplicate, a symlink, a hard link to the target file, or a hardlink to a duplicate. Along its way, it may run into errors ranging from: accessing a directory without permission, reading a non-existent file, or hunting for links to non-file objects which the program will print coherent error reporting.

Example: The following example shows the typical use case and output generated by the program. In this example, we have a file called target.txt located in a folder on our desktop. We will call on our program to search through Test_Folder, a directory also on the desktop for any hardlinks, symlinks, etc. The file structure for Test_Folder along with the output from our program when used on this directory can be seen on the next page:



```
[fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set2 % ./hunt ../Test_Folder target.txt
SEARCHING DIRECTORY ../Test_Folder

../Test_Folder/target_duplicate.txt      DUPLICATE OF TARGET (nlink=2)

SEARCHING DIRECTORY ../Test_Folder/4

../Test_Folder/4/target_symlink.txt      SYMLINK RESOLVES TO TARGET
SEARCHING DIRECTORY ../Test_Folder/3

../Test_Folder/3/target_hardlink.txt     HARD LINK TO TARGET
SEARCHING DIRECTORY ../Test_Folder/2

../Test_Folder/2/target_duplicate_hardlink.txt   DUPLICATE OF TARGET (nlink=2)

SEARCHING DIRECTORY ../Test_Folder/2/3

../Test_Folder/2/3/target_duplicate_2.txt        DUPLICATE OF TARGET (nlink=1)

SEARCHING DIRECTORY ../Test_Folder/2/2

../Test_Folder/2/2/target_duplicate_symlink.txt         SYMLINK (/Users/fakharyarkhan/Desktop/Test_Folder/target_duplicate.txt) RESOLVES TO DUPLICATE
```

```
Test_Folder/
├── target_duplicate.txt
├── file1
├── 2/
│   ├── 2/
│   │   └── target_duplicate_symlink.txt
│   ├── 3/
│   │   ├── file2
│   │   └── target_duplicate_2.txt
│   └── target_duplicate_hardlink.txt
├── 3/
│   └── target_hardlink.txt
└── 4/
    └── target_symlink.txt
```

As can be seen, our program begins by going through the files directly underneath Test_Folder. The files called file[num] are files that are completely unrelated to target.txt. As a result, we receive no output from the program when it evaluates file1 as it wasn't a match. The file target_duplicate is a copy of target.txt and also since there is a hardlink to target_duplicate in Test_Folder/2, the file has an nlink of 2.

The program then looks through directory 4 (we tried to see what order readdir gives out the contents of the folder and it looks like it depends on the filesystem implementation) and correctly identifies target_symlink.txt as a symlink to target.txt. Likewise, in directory 3 it finds a hardlink to target.txt and identifies it as such. Now the program looks through directory 2 where it finds the hardlink to target_duplicate.txt. Then it looks through subdirectory 3 and finds another duplicate of target.txt. Finally, it searches through subdirectory 2 and finds a symlink to target_duplicate.txt and because it points to a duplicate of the target, the output gives the file path to the duplicate as well.