Intro. to Operating Systems Programming Assignment

File Deduplication

Prof. Li-Pin Chang
CS@NYCU

Objectives

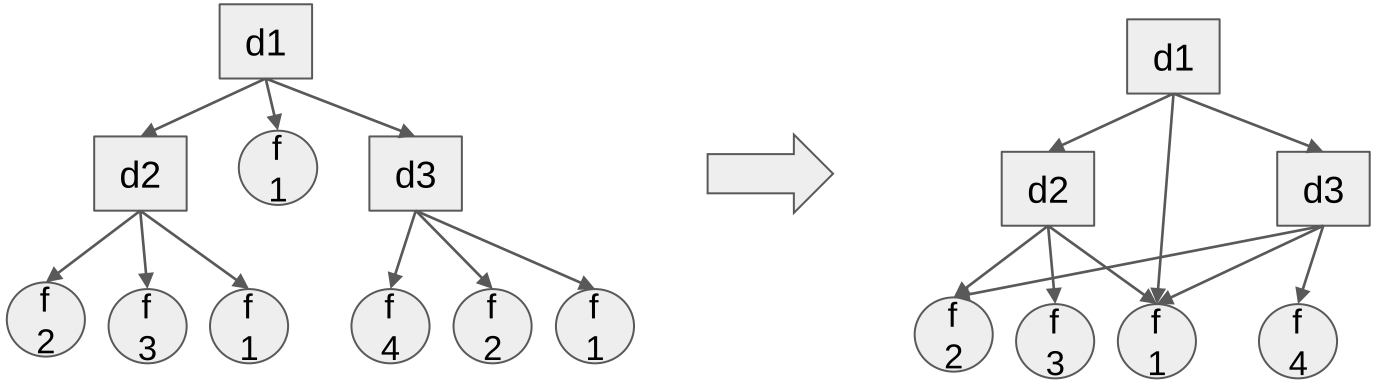
- Design and Implementation of file deduplication
- Skills to be learned from this assignment
    - Directory traverse APIs
    - File link APIs (hard link)
    - File duplication detection method

File Deduplication

- A critical technique for storage space saving; applications include
  - Differential backup
  - Virtual machine image management
  - Git repository actually involves file deduplication as well

Program Input and Output

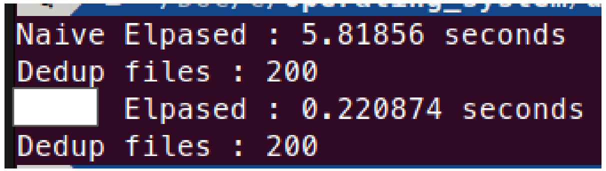
- Input: a directory of (many) files
  - TA will provide an archive file; extract this file to generate the source directory
- Output: the same directory with duplicate files replaced with hard links
  - Perform in-place deduplication, i.e., dedup in the source directory



Unused duplicated files must be deleted!

File Duplication Detection

- Two duplicate files have exactly the same contents, but may have different names, timestamps and they may appear in different directories
- You are not allowed to do verbatim comparison among files as it is extremely slow and scales poorly (see →)
  - The execution time will reveal this if you do…



```
Naive Elpased : 5.81856 seconds
Dedup files : 200
      Elpased : 0.220874 seconds
Dedup files : 200
```

- Think about it: How to detect a duplicate accurately and efficiently?
- Unique files having the same size/name will be used for testing

Hints

- *You may need SHA1 APIs*
- Install necessary files with command “sudo apt install libssl-dev”
- #include <openssl/sha.h>
    - SHA1_Init(), SHA1_Update(), and SHA1_Final()
- MD5 is okay, but SHA1 is more robust

- To create a hard file link, use link()
- To delete unnecessary duplicated files, use unlink()

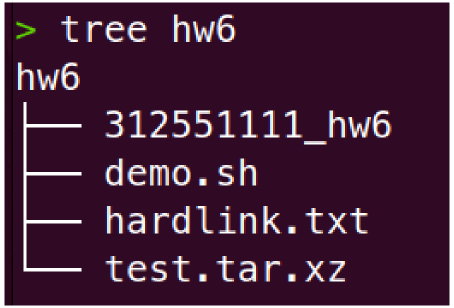
Run Your Program

1. Extract test.tar.xz to some directory.
```
$ tar -xf ./test.tar.xz --directory=$DIRECTORY
```
2. Execute your program with the directory name
```
$ ./$EXECUTABLE $DIRECTORY
```
3. For example
```
 $ mkdir -p test
 $ tar -xf ./test.tar.xz --directory=test
 $ ./312551111_hw6 ./test
```
- Your program must perform file de-duplication correctly for all files in the directory tree “test”
- But for grading, we will use a shell script (see the next slice)

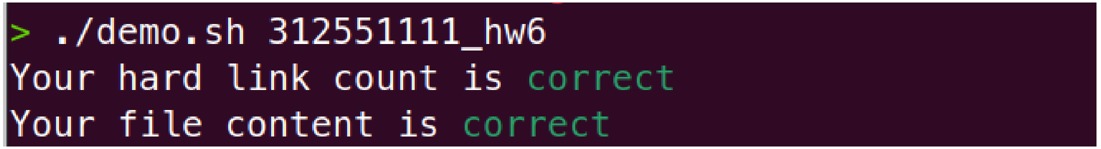
Test Script (for Grading)

You will receive **hw6.zip** from us. Steps for testing:

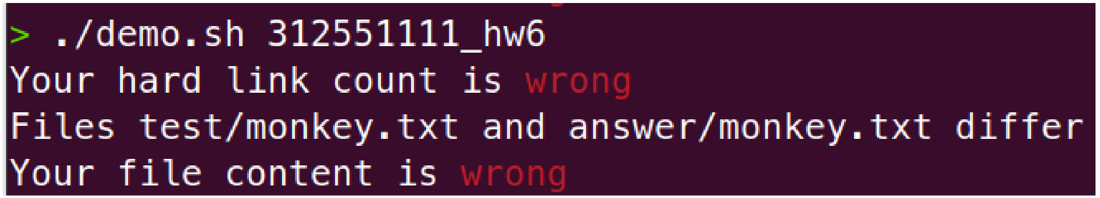
1. Download and extract **hw6.zip**
2. Put **your executable** and **demo.sh** in the same directory
3. Change demo.sh permission
   ```
   $ chmod u+x demo.sh
   ```
4. Run the script
   ```
   $ ./demo.sh $EXECUTABLE
   ```



```
> tree hw6
hw6
├── 312551111_hw6
├── demo.sh
├── hardlink.txt
└── test.tar.xz
```



```
> ./demo.sh 312551111_hw6
Your hard link count is correct
Your file content is correct
```



```
> ./demo.sh 312551111_hw6
Your hard link count is wrong
Files test/monkey.txt and answer/monkey.txt differ
Your file content is wrong
```

This script will check file hard link counts and compare contents of all files

Grading Policy

- The script demo.sh must output “correct”
  - Both hardlink count and file contents
- Do not use verbatim file content comparison
  - Will be detected easily by your total execution time
  - You will receive a score penalty if you do so

Header of your .c or .cpp

```
/*
Student No.: <your student id>
Student Name: <your name>
Email: <your email>
SE tag: xnxcxtxuxoxsx
Statement: I am fully aware that this program is not supposed to be posted to a
public server, such as a public GitHub repository or a public web page.
*/
```

Testing OS Environment

- Ubuntu 18.04
- Install as a VM or on a physical machine

Credit

- 魏翌丞 helpd design this project
- Direct your questions to the TAs