Open-source Software Lab

# Project 2. Keep: Directory Backup Tool

11 May 2023

Shin Hong

# Motivation

- 프로젝트 결과 보고서_1126
- 프로젝트 결과 보고서_1126_수정
- 프로젝트 결과 보고서_1126_수정2
- 프로젝트 결과 보고서_최종
- 프로젝트 결과 보고서_최종_보고용
- 프로젝트 결과 보고서_최종_보고용_1127 수정
- 프로젝트 결과 보고서_최종_보고용_1127 최종
- 프로젝트 결과 보고서_최종_보고용_1127 최종_진짜진짜진짜 최종
- 프로젝트 결과 보고서_최종_보고용_1127 최종_진짜진짜최종

# Overview

- Construct a directory backup tool `keep` in C programming language
  - with keep, a user can save a series of the snapshot of a target directory
  - also, a user can bring back a stored status of the target directory using keep
- Use file-I/O functions appropriately, and handle various error cases properly
- Submit your resulting artifact and a demo video by 9 PM, Mon 29 May

# Keep Commands

- A user can create a backup space for a target directory, and then copy the files in the directory to the backup space using keep

- User commands
  - `keep init`: construct a backup space for the current directory, and initialize it
  - `keep track <file or directory>`: start to track a specified file or all files under the specified directory as backup targets
  - `keep untrack <file or directory>`: cancel tracking the specified file or all fines under the specified directory
  - `keep store “<note>”`: store the current status of the current directory
  - `keep restore <version>`: bring back the status of the specified version
  - `keep versions`: show the list of versions

# Directory Structure

- The backup space of a target directory is a hidden directory `.keep` at the target directory
- For each store command, the copy of the target directory is given with a unique version number
  - a version number starts from one, and then increases by one at a time
- The `.keep` directory has the following files and directories:
  - `tracking-files` : list up the information of all tracking files
  - `latest-version` : store the latest version number, or 0 at the beginning
  - <version> : store all information and files of the specified version (e.g., 1, 2)
    - `tracking-files`
    - `note`
    - `target` : the directory for holding the backup files

# Example (1/2)

```
$ ls -R
README.md
ex1/hello.c
ex1/main.c
ex1/data/d001
ex1/data/d002
ex2/list1.c
ex2/list2.c
ex2/list3.c
$ keep init
$ ls .keep
tracking-files
latest-version
```

```
$ keep track README.md
$ keep track ex1
$ keep store “First version”
stored as version 1
$ ls -R .keep
tracking-files
latest-version
1/tracking-files
1/note
1/target/README.md
1/target/ex1/hello.c
1/target/ex1/main.c
1/target/ex1/data/d001
1/target/ex1/data/d002
```

```
$ keep versions
1  First version
$ vim README.md
$ keep untrack ex1/main.c
$ keep store “Second one.”
stored as version 2
$ keep versions
1  First version
2  Second one.
$ ls -R .keep/2
2/tracking-files
2/note
2/target/README.md
1/target/ex1/hello.c
1/target/ex1/data/d001
1/target/ex1/data/d002
```

# Example (2/2)

```
$ rm -rf README.md
$ keep store “Remove README”
stored as version 3
$ ls .keep
tracking-files
latest-version
1
2
3
```

```
$ keep restore 2
restored as version 2
$ ls -R
README.md
ex1/hello.c
ex1/data/d001
ex1/data/d002
$ vim ex1/hello.c
$ keep store “update hello.c”
stored as version 4
$ cat .keep/latest-version
4
```

```
$ ls -R .keep/4
4/tracking-files
4/note
4/target/README.md
4/target/ex1/hello.c
4/target/ex1/data/d001
4/target/ex1/data/d002
$ keep store “backup”
nothing to store
$
```

# Functionalities (1/4)

- `keep init`
  - make `.keep` at the current directory, and create `tracking-files` as an empty file and `latest-version` to have `0`
  - print an error message if `.keep` already exists
- `keep track <file or directory>`
  - add the entry of the specified file, or the entries of all files under the specified directory to `tracking-files`
    - a file entry consists of the file path, and its last modification time
    - when a file is newly added, define the last modification time as 0

# Functionalities (2/4)

- `keep untrack <file or directory>`
  - remove the entry of the specified file, or the entries of all files under the specified directory to `tracking-files`

- `keep versions`
  - display the version number and the note of all existing versions

# Functionalities (3/4)

- `keep store “note”`
  - identify which tracking files has been modified
    - for each tracking file, compare the actual last modification time (`st_mtime` of `struct stat`) and the one recorded in `tracking-flies`
    - Show “nothing to update” and reject the command if no file was updated
  - set up the version directory under .keep
    - make the version directory and the `target` directory
    - create `tracking-files` and
  - copy each tracking file to `target`
    - (optional)
      - if the file was modified after the latest version, copy the actual file
      - if unmodified, create a hard link to the file of the latest version
  - increment the latest version number

# Functionalities (4/4)

- `keep restore <version>`
  - check if any tracking file was modified after the latest store or restore, and reject the command if a modified file exists
    - for each tracking file, compare the actual last modification time and the one in `tracking-flies`
  - copy each tracking file from the version directory to the target directory
  - erase non-tracking files
  - update `./keep/tracking-files`

# Other Requirements

- Assume that no hard or symbolic links exist under the target directory
- Assume that a user does not use keep for a directory in parallel
- Show proper error messages to different exceptional cases, and handle them properly
- Write a build script as Makefile

# Instructions

- Open chat for Q&A: https://open.kakao.com/o/gII0Vjbf
- Demo video
  - No more than 5 minutes.
  - Each member must take a part in the video.
  - Upload your demo video to YouTube, and write down the URL on the submission message
  - You must use English in recording a video demo
  - Your demo video may be shared in the class, especially if it is to be recognized.
- Submission
  - HDLMS
  - One team, one submission
  - No late submission will be accepted

# Evaluation

- Your result will be evaluated according to the following criteria:
  - all functionalities are correctly implemented
  - the source code is clean and comprehensible
  - the video clearly evidences that the program works correctly, and handle various error cases properly
- You will get extra points if you add new interesting instructions to Keep
- Good demo videos will be recognized