

Cpr E 308:Operating Systems, Principles and Practice, Spring 2006

Project 3 (due April 24)

1 Goal

The goal of this project is to gain experience in using system calls and library functions relevant to the file system by designing and implementing a file synchronization program.

2 Description

In this project you are required to design and implement a file system synchronization program. The name of the program is `synch` and its syntax is as follows.

```
synch src dst
```

`src` and `dst` are the names of two directories. The program should examine all files and subdirectories in both `src` and `dst`. If any file or subdirectory under `src` does not appear in `dst`, or if the file appeared in `dst`, but the version under `src` was modified more recently than the version under `dst`, then the file should be copied from `src` to `dst`. Any redundant files or subdirectories in `dst` should be left there. For example, if `src=/home/newton/work` and `dst=/home/newton/backup`, and the file `/home/newton/work/math/a` does not appear in `/home/newton/backup/math` or if file `/home/newton/work/math/a` was modified more recently than `/home/newton/backup/math/a`, then `/home/newton/work/math/a` should be copied into `/home/newton/backup/math/a`. Note that the directory names (`src` and `dst`) may be either absolute pathnames (starting from the root “/”) or relative pathnames (starting from the current directory).

Here are two examples:

1. Suppose `src=/home/newton/work` and `dst=/home/newton/backup`, and the contents of the two directories were as follows.

```
/home/newton/work/a.txt: file, time of latest modification = 2006, April 1, 3:07:02 am.  
/home/newton/work/b.txt: file, time of latest modification = 2006, April 1, 5:07:02 am.  
/home/newton/work/c.txt: file, time of latest modification = 2006, April 1, 2:07:02 am.  
/home/newton/work/d: directory, time of latest modification = 2006, April 1, 9:07:02 am.
```

```
/home/newton/backup/a.txt: file, time of latest modification = 2006, April 1, 3:07:02 am.  
/home/newton/backup/b.txt: file, time of latest modification = 2006, April 1, 4:07:02 am.
```

In this case, when `synch src dst` is called,

- (a) `/home/newton/work/b.txt` should be copied into `/home/newton/backup/b.txt`, since `/home/newton/work/b.txt` is more recently modified.

(b) `/home/newton/work/c.txt` and `/home/newton/work/d` should be copied into `/home/newton/backup/c.txt` and `/home/newton/backup/d` respectively, since they do not appear in `/home/newton/backup`. Note that since `d` is a directory, all files and subdirectories in `/home/newton/work/d` should be recursively copied to `/home/newton/backup/d`.

2. `src=/home/newton/work` and `dst=/home/newton/backup`. Here is the content of these two directories.

```
/home/newton/work/a.txt: file, time of latest modification = 2006, April 1, 3:07:02 am.  
/home/newton/work/b.txt: file, time of latest modification = 2006, April 1, 5:07:02 am.  
/home/newton/work/c.txt: file, time of latest modification = 2006, April 1, 2:07:02 am.  
/home/newton/work/d: directory, time of latest modification = 2006, April 1, 9:07:02 am.  
/home/newton/work/d/e.txt: file, time of latest modification = 2006, April 1, 1:07:02 am.  
/home/newton/work/d/f.txt: file, time of latest modification = 2006, April 1, 7:07:02 am.  
/home/newton/work/d/g.txt: file, time of latest modification = 2006, April 1, 9:07:02 am.
```

```
/home/newton/backup/a.txt: file, time of latest modification = 2006, April 1, 3:07:02 am.  
/home/newton/backup/b.txt: file, time of latest modification = 2006, April 1, 4:07:02 am.  
/home/newton/backup/d: directory, time of latest modification = 2006, April 1, 5:07:02 am.  
/home/newton/backup/d/e.txt: file, time of latest modification = 2006, April 1, 1:07:02 am.  
/home/newton/backup/d/f.txt: file, time of latest modification = 2006, April 1, 5:07:02 am.
```

In this case, when `synch src dst` is called,

- (a) `/home/newton/work/b.txt` and `/home/newton/work/c.txt` should be copied into `/home/newton/backup/b.txt` and `/home/newton/backup/c.txt` respectively.
- (b) Since both `/home/newton/work/d` and `/home/newton/backup/d` exist, the program should treat `/home/newton/work/d` as a new source directory, and `/home/newton/backup/d` as a new destination, and synchronize them recursively. Due to this recursive processing, `/home/newton/work/d/f.txt` and `/home/newton/work/d/g.txt` will be copied to `/home/newton/backup/d/f.txt` and `/home/newton/backup/d/g.txt`. Note that the program should ignore the `.` and `..` directories, which are created automatically by the system, otherwise the program will run forever.

Think about any error handling that the program may need to perform. In particular, think about the situation when `src` is a prefix of `dst`, i.e., `dst` is a subdirectory of `src`, or vice versa. Explain what errors could occur in these scenarios. In your implementation, the program should detect these two situations and exit with an error message if either occurs.

3 Output

The program should verify if it has appropriate permissions to do the required operations. If the program does not have sufficient permissions, then it should print an error message for the respective files/directories, such as `no executable permission of dir:/home/newton/cpre308` and should continue synchronizing the other files and directories.

Files and directories that are synchronized should be printed out on the screen with information such as:

```
/home/newton/cpre308/proj3 --> /home/newton/cpre308.bak/proj3
/home/newton/coms311/a.c --> /home/newton/coms311.bak/a.c
/home/newton/coms311/p1/s.c --> /home/newton/coms311.bak/p1/s.c
```

4 Useful Tools

The following system calls and library functions maybe useful in programming. You may obtain more information about them using the Linux man pages. Of course, it does not necessarily mean that you have to use all these tools.

- File Manipulation: `open`, `close`, `creat`, `read`, `write`.

These are the very basic system calls relevant to the file system operations in Linux system. They are used to implement other file system relevant library functions, which make it easier for the user to manipulate the file system in programming. If you directly use these system calls, you must understand the low level directory structure in the Linux file system.

- Directory Manipulation: `opendir`, `closedir`, `readdir`, `rewinddir`, `scandir`, `seekdir`

These functions are specially implemented in Linux for directory manipulation. Note that in Linux, a directory is a special type of file whose data is a list of files and subdirectories contained within the directory.

The above functions allow the user to read/manipulate the directory. For example, `opendir/closedir` can be used to open/close the directory, `readdir` can be used to read an item in the directory. Note that the user cannot directly modify a directory; a directory is changed by the system whenever files are added/deleted from it. The above functions calls allow more convenient access to the directory.

- Getting information about a file: `stat`, `lstat`, `fstat`, `access`

The functions `stat/lstat/fstat` return information about an existing file. The function `access` can be used to test the existence of a file.

- Execute a shell command: `system`

This function helps you to execute a shell command in programming, like the `cp` command that we might need to execute for our file system synchronization. For example, calling `system("cp /home/newton/work/main.c /home/newton/bak/main.c")` will execute the shell command: `cp /home/newton/work/main.c /home/newton/bak/main.c` in `/bin/sh`. You can obtain more information by "`man 3 system`" in Linux.

5 Report

To hand in the project, do the following.

1. Limit the source code into one `.c` file accompanied by a `readme` file, which explains how to compile the source file. Compress these two files into one zip file and email it to your TA. The name of the zip file should be “`firstname_lastname_sectionnumber_proj3.zip`”. For example, if Isaac Newton in section 3 were to email his program, the zip file would be called “`Isaac_Newton_3_proj3.zip`”.
2. Include the following in your lab report.
 - Your name and lab section.
 - A brief description of what you learned from the project (no more than 2 paragraphs).
 - A description of the key algorithm used in synchronizing the two file systems.
 - An explanation of what problems may occur in `synch src dst` if `dst` is a subdirectory of `src` or vice versa.

6 Grading

70% for the correctness of the program, 20% for the formatted code and the coding style, 10% for the writeup.

7 Due Date

The project report is due April 21, 2006 at the beginning of class. The source code should be emailed to the TA before 5pm on the April 21.

Late Policy: You will lose 10% per day for one week. After one week you will no longer get any credit.