Assignment 2: Finding Files

Most people, when using a computer system, often need to find files that they have accessed recently, possibly just read or perhaps modified. They may have forgotten where in the directory hierarchy the files were, or even forgotten the names of the files, but they know that it was in the last day or last week, or last hour perhaps. Most operating systems provide a command to help them search. These commands are often either slow and exhaustive, or complicated to learn.

In this assignment, I am asking you to write a simplified version of the UNIX `find` command. The `find` command is extremely versatile, very useful, and *very, very complex*. This assignment is simple by comparison, but it will require you to solve an interesting and useful problem. Your program must implement the command described by the man page below. You can write it in C or C++. You cannot use any commands to solve this problem; you must use the file system calls provided by the kernel or system libraries. If you can write this command, it is just another step beyond that to design a version of it that is more versatile and can run in a graphical user interface. Or perhaps design a program that recursively searches for other properties of interest.

NAME

`sfind - find files that have been accessed or modified within a given time period`

SYNOPSIS

`sfind [-a] [`*`TIME_UNITS`*`] `*`n`*` [`*`directory_name`*`] ...`

DESCRIPTION

`sfind` recursively searches through the directory hierarchies whose roots are the directories specified on the command line, for all non-directory files that have been *modified* within the specified time period, inclusively. If no directory is specified, the current working directory is assumed.

OPTIONS

There is only one option:

`-a` `sfind` displays files that have been accessed for reading or writing

TIME_UNITS

The time can be specified in exactly one of four different units of measurement:

`-s` The unit of time is seconds
`-m` The unit of time is minutes
`-h` The unit of time is hours
`-d` The unit of time is days

The number following the time unit designator is the number of time units. *If no time unit designator is supplied, days are assumed.* The number must be a positive integer, otherwise the command returns immediately displaying the message "Invalid time specification." If more than one unit is specified, it is an error and the `sfind` command returns immediately displaying the message "Invalid time specification."

EXAMPLES

`find -m 120 ..`	searches the parent directory for all files that have been modified within the past 120 minutes.
`find -a 1`	searches the current working directory for all files that have been accessed within the past day.
`find -h2 /tmp ~`	searches `/tmp` and the home directory for all files that have been modified within the past 2 hours.
`find -ad 7`	searches the current working directory for all files that have been accessed within the past 7 days.
`find -s 240 /tmp`	searches the `/tmp` directory for all files that have been modified within the past 240 seconds.
`find -s -5`	returns with an error message.

NOTES

The command will display the names of each file that has been accessed within the specified time period, one per line, *along with its time of last access or modification.* It does not need to display whether or was a read or a write, since it has no means of knowing, since both a read and a write are considered an access.

The granularity of this command is exact to the second. In other words, if 1 day is specified, and the current time is October 17, 13:00:01 (1:01 PM), then a file accessed on October 16 at 13:00:00 is not displayed, because it was accessed more than 1 day before the time that the command was initiated, albeit it one second more.

If the command does not have permission to read or execute a directory, it skips over it and writes its name to the `stderr` stream. If it cannot query a file because it does not have appropriate permission, it skips over it and writes its name to the `stderr` stream. The `sfind` command does not return an error value if it has to skip over files or directories.

`sfind` does not follow symbolic links. It ignores them. (Trying to follow symbolic links opens the possibility of infinite loops, which this program *abhors.*)

`sfind` displays nothing else. No messages, no greetings, no niceties. (Just stick to the facts, Ma'am.)

EXIT STATUS

`0` If it succeeded.

`1` If it failed.

Submitting the Assignment

You are to create a zip file containing all of your source code and put that zip file in the directory

```
/data/biocs/b/student.accounts/cs493.66/projects/project2
```

naming it `username_hwk2.zip`. Give it permissions 600 so that only you have access to it. Whether you have a single file or multiple files, you are to create a directory named `username_hwk2`, putting all files into it and using the command

```
zip -r username_hwk2.zip username_hwk2
```

to create the zip file. Make sure that you create a Makefile if you use multiple files, and include that Makefile in the directory. If you use a Makefile, make sure that it creates an executable named `sfind` when it is built. If it is a single file program, name the source file `sfind.X`, where `X` is the GNU extension for the language (.c for C, .cpp or .C for C++, etc.)

The program must be well-documented and must conform to my programming guidelines for full credit. It must be placed in the directory no later than midnight of the due date.