NYU Tandon – CS3224 Fall 2018
Operating Systems Prof. Sandoval

Homework 5

Academic Honesty

Aside from the narrow exception for collaboration on homework, all work submitted in this course must be your own. Cheating and plagiarism will not be tolerated. If you have any questions about a specific case, please ask me. We will be checking for this!

NYU Poly's Policy on Academic Misconduct:
http://engineering.nyu.edu/academics/code-of-conduct/academic-misconduct

General Notes:

- Read the assignment carefully, including what files to include.
- Don't assume limitations unless they are explicitly stated.
- Treat provided examples as just that, not exhaustive list of cases that should work.
- When in doubt regarding what needs to be done, ask. Another option is test it in the real UNIX operating system. Does it behave the same way?
- **TEST** your solutions, make sure they work. It's obvious when you didn't test the code.

Adding Timestamps to the XV6 FileSystem

In this assignment you will add support for tracking when a file was created to the xv6 filesystem. We will also add support to the `ls` and `mkfs` utilities so that they display and create timestamps, respectively.

Getting the code from Github

As usual, we'll be working off of a slightly modified version of xv6. The only difference is that the **`date.h`** header, which you will need in order to get the current time, has been modified so that it can be included multiple times without causing a compile error.

If you still have your xv6 directory from last time, remove or rename it first. Then get the base xv6 code for this assignment:

```
$ git clone https://github.com/moyix/xv6-public.git
```

```
Cloning into 'xv6-public'...
remote: Counting objects: 4517, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 4517 (delta 1), reused 0 (delta 0), pack-reused 4512
Receiving objects: 100% (4517/4517), 11.67 MiB | 6.31 MiB/s, done.
Resolving deltas: 100% (1825/1825), done.
Checking connectivity... done.
$ cd xv6-public/
$ git checkout hw7
Branch hw7 set up to track remote branch hw7 from origin.
Switched to a new branch 'hw7'
```

Part 1: Tracking Creation Time

In xv6, each file and directory is associate with an ***i-node***, you might remember the i-node data structure we saw in class, which keeps track of details such as the file's type (file, directory, or device), reference count, and list of disk blocks. i-nodes are stored both on disk (in a `` `struct dinode` ``) and in memory (`` `struct inode` ``), and xv6 copies data between the two structures as needed.

Modify the filesystem code so that on-disk and in-memory i-nodes both contain a field that keeps track of the time the i-node was created. Make sure that you modify both the on-disk and in-memory i-node structures, and that when xv6 copies data between them you also copy the creation time.

You can verify that you did this part correctly by using `gdb` to examine a newly created i-node and making sure that it contains the correct timestamp (or just do part 2, which modifies the `ls` command to print out timestamps as well).

Hints

* All files are created using the `create()` function in `` `sysfile.c` ``.
* You can use the `` `cmostime()` `` function to get the current date and time from **inside** of xv6.
* Because they need to fit neatly into one disk block, the size of the `struct dinode` must divide the block size evenly. So when you add in the timestamp field, you will have to pad out the structure **somehow** (perhaps by increasing the number of direct block pointers).

Part 2: Displaying Timestamps

Modify the **`ls`** command so that it prints out the creation time of each

file or directory.

```
$ ls
.              1 1 512 0/0/0 00:00:00
..             1 1 512 0/0/0 00:00:00
README         2 2 1972 0/0/0 00:00:00
cat            2 3 13420 0/0/0 00:00:00
echo           2 4 12525 0/0/0 00:00:00
forktest       2 5 8257 0/0/0 00:00:00
grep           2 6 15088 0/0/0 00:00:00
init           2 7 13190 0/0/0 00:00:00
kill           2 8 12677 0/0/0 00:00:00
ln             2 9 12531 0/0/0 00:00:00
ls             2 10 15259 0/0/0 00:00:00
mkdir          2 11 12678 0/0/0 00:00:00
rm             2 12 12659 0/0/0 00:00:00
sh             2 13 23639 0/0/0 00:00:00
stressfs       2 14 13401 0/0/0 00:00:00
usertests      2 15 58688 0/0/0 00:00:00
wc             2 16 13938 0/0/0 00:00:00
zombie         2 17 12295 0/0/0 00:00:00
console        3 18 0 11/20/2015 23:08:00
$ echo hello > file.txt
$ ls file.txt
file.txt       2 19 3 11/21/2015 23:14:02
```

Hints:

* `ls` uses the `fstat` call to retrieve information about the file or directory; you will need to modify the implementation of `fstat` so that it returns information about the creation time as well.
* Don't bother making sure that leading zeroes in the time are printed correctly. You can just use the "%d" format character.

Part 3: timestamps in mkfs

You may have noticed that many of the files on the filesystem have a creation time of 0/0/0 00:00:00. This is because the initial filesystem image (`fs.img`) is not created from within xv6, but rather by the `mkfs` program, which is run outside of xv6 by the Makefile.

To finish up the implementation, modify `mkfs.c` so that it writes the current timestamp into the on-disk i-node when the initial filesystem image is created. Verify that when you boot xv6, you now get a directory listing that looks like:

```
$ ls
.              1 1 512 11/22/2015 00:00:29
..             1 1 512 11/22/2015 00:00:29
README         2 2 1972 11/22/2015 00:00:29
cat            2 3 13420 11/22/2015 00:00:29
echo           2 4 12525 11/22/2015 00:00:29
forktest       2 5 8257 11/22/2015 00:00:29
grep           2 6 15088 11/22/2015 00:00:29
init           2 7 13190 11/22/2015 00:00:29
kill           2 8 12677 11/22/2015 00:00:29
ln             2 9 12531 11/22/2015 00:00:29
ls             2 10 15631 11/22/2015 00:00:29
mkdir          2 11 12678 11/22/2015 00:00:29
rm             2 12 12659 11/22/2015 00:00:29
sh             2 13 23639 11/22/2015 00:00:29
stressfs       2 14 13401 11/22/2015 00:00:29
usertests      2 15 58688 11/22/2015 00:00:29
wc             2 16 13938 11/22/2015 00:00:29
zombie         2 17 12295 11/22/2015 00:00:29
console        3 18 0 11/22/2015 00:00:32
```

Hints:

* You will likely find the `gmtime()` function helpful.
* You may need to run `make clean` in order to remove the old `fs.img` and re-run `make` before the timestamps show up correctly (or just run `rm fs.img`).

Submitting

As with some previous assignments there are 3 steps to submit your homework.

1. use git to **commit your changes**,
2. **create a patch** and finally
3. **submit** the patch to NYU Classes.

Here are steps in detail

1. In order to **commit your changes** you type the following (the message can be different):

```
    $ git commit --all --message="Add filesystem timestamps"
    [hw7 b778e43] Add filesystem timestamps
     7 files changed, 31 insertions(+), 3 deletions(-)

(Note: if you added any new files, you will also have to use `git add
```

```
<filename>` before you run `git commit`.)
```

You can can type git status to make sure nothing is uncommited.

2. In order to **create a patch file**, you need to type:

```
$ git format-patch hw7.unmodified
0001-Add-filesystem-timestamps.patch
```

This command will create a file, `0001-Add-filesystem-timestamps.patch`, containing the changes you've made.

3. **Submit** this file on ****NYU Classes****

Submission Notes

Don't try to edit the patch file after creating it. Doing so will most likely corrupt the patch file and make it impossible to apply. Instead, change the original file, commit your changes, and run `` `git format-patch` `` again:

```
$ git commit --all --message="Description of your change here"
[hw7 7cc4977] Description of your change here
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git format-patch hw7.unmodified
0001-Add-filesystem-timestamps.patch
0002-Description-of-your-change-here.patch
```

Then submit *both* patch files.