CSE130 Winter 2020 : Assignment 5

In this assignment you will use system calls to read and write files and explore directory structures.

This lab is worth 5% of your final grade.

Submissions are due NO LATER than 23:59, Monday March 9 (1 week)

Setup

SSH in to one of the two CSE130 teaching servers using your CruzID Blue password:

```
$ ssh <cruzid>@noggin.soe.ucsc.edu
```
(use Putty http://www.putty.org/ if on Windows)

Or
```
$ ssh <cruzid>@nogbad.soe.ucsc.edu
```

Authenticate with Kerberos: ***(do this every time you log in)***

```
$ kinit
```
(you will be prompted for your Blue CruzID password)

Authenticate with AFS: ***(do this every time you log in)***

```
$ aklog
```

Create a suitable place to work: ***(only do this the <u>first</u> time you log in)***

```
$ mkdir -p CSE130/Assignment5
$ cd CSE130/Assignment5
```

Install the lab environment: ***(only do this once)***

```
$ tar xvf /var/classes/CSE130/Winter20/Assignment5.tar.gz
```

Build the starter code:

```
$ cd ~/CSE130/Assignment5
```
(always work in this directory)
```
$ make
```

Then try:

```
$ make grade
```
(runs the required functional tests - see below)
(also tells you what grade you will get - see below)

Run the incomplete fileman executable:

```
$ ./fileman -b
```
(basic tests)
```
$ ./fileman -s
```
(advanced tests)
```
$ ./fileman -s
```
(stretch tests)

Additional Information

In this assignment you can make use of the Linux versions of the Pintos system calls you implemented in the second half of Lab 3. However, you can use any standard Linux utility you find useful.

Do NOT use the `system()` or `exec()` family of system calls!

Requirements

Basic: Read and write ASCII text files

Read Missing	Return appropriate error code when asked to read a non-existent file.
Read Exists	Read requested number of bytes from an existing file starting at the first byte of the file and populating from the first byte of the supplied buffer.
Read File Offset	Read requested number of bytes from an existing file starting at a given byte of the file and populating from the first byte of the supplied buffer.
Read Buffer Offset	Read requested number of bytes from an existing file starting at the first byte of the file and populating from a given byte of the supplied buffer.
Read Both Offset	Read requested number of bytes from an existing file starting at a given byte of the file and populating from a given byte of the supplied buffer
Write Missing	Create a file and write the requested number of bytes from the supplied buffer starting at the first byte of the new file and the first byte of the supplied buffer.
Write Exists	Return appropriate error code when asked to write to an existing file.
Write File Offset	Create a file and write the requested number of bytes from the supplied buffer starting at the requested byte of the new file and the first byte of the supplied buffer.
Write Buffer Offset	Create a file and write the requested number of bytes from the supplied buffer starting at the first byte of the new file and the requested byte of the supplied buffer.
Write Both Offset	Create a file and write the requested number of bytes from the supplied buffer starting at the requested byte of the new file and the requested byte of the supplied buffer.

Advanced: Copy and append data to ASCII text files

Append Missing	Return appropriate error code when asked to append to a non-existent file.
Append Exists	Append the contents of a supplied buffer to an existing file.
Copy Missing	Return appropriate error code when asked to copy from a non-existent file.
Copy Exists	Copy an the contents of an existing file to a new one.

Stretch: List a directory tree like so:

```
home
└── vkhqmgwsgd
    ├── agmugje
    │   ├── fwbv
    │   │   ├── trx
    │   └── surxeb
    │       ├── dyjxfseur
    │       └── wy
    └── tcx
        └── jbjfwbv
            └── rnixyjz
```

Note all characters are ASCII so you can simply copy them from the example in the `fileman.h` header file.

What steps should I take to tackle this?

This assignment is all about research. There are many ways to pass the tests, so find one that works. Using the system calls you became familiar with in Lab 3 will be an easy way to get started.

How much code will I need to write?

A model solution that satisfies all requirements has approximately 200 lines of executable code.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

    a. Basic Requirements (30%)
    b. Advanced Requirements (30%)
    c. Stretch Requirements (30%)
    d. Your implementation is free of compiler warnings (10%)

2. (-100%) **Did you use the** `system()` **or** `exec()` **family of system calls?**

    Do not use these system calls. Use system calls like `open()`, `read()`, and `write()`.

3. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%). You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy.

    b. Your submission is determined to be a copy of a past or present student's submission (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 33% copied code No deduction
        - 33% to 66% copied code (-50%)
        - > 66% copied code (-100%)

What to submit

In a command prompt:

```
$ cd ~/CSE130/Assignment5
$ make submit
```

This creates a gzipped tar archive named `CSE130-Assignment5.tar.gz` in your home directory.

****** UPLOAD THIS FILE TO THE APPROPRIATE CANVAS ASSIGNMENT ******

