ECEC 353: Systems Programming
Programming Assignment: Filesystem Logging

Instructor: Michael Lui
ECE Department, Drexel University

August 17, 2016

This assignment is due by September 2, 2015, 11:59 pm via BBLearn. You may work on this assignment in a team of up to two people.

The FUSE library allows users to implement and use filesystems that are not natively supported by the kernel. The Linux Virtual Filesystem (**VFS**) makes this possible by abstracting filesystem actions. This allows for features such as NTFS support on Linux.

For this project, you will be modifying a simple filesystem implementation in FUSE. We will be using the `bbfs` filesystem created by Dr. Joseph Pfeiffer of NMSU, available at:
`http://www.cs.nmsu.edu/~pfeiffer/fuse-tutorial/`

The example code can be downloaded via:
```
$ curl http://www.cs.nmsu.edu/~pfeiffer/fuse-tutorial.tgz | tar xz
```

Although you are not required to know specifics about FUSE, it would be useful to familiarize yourself with the implementation by looking at:
`src/bbfs.c`
`src/log.c`
`src/params.h`

The application `bbfs` is built with:
```
$ ./autogen.sh
$ ./configure
$ ./make
```

The binary is created as `src/bbfs`.

You can now use this binary to mount any accessible directory onto another directory.
```
$ src/bbfs my/root/directory my/mounted/directory
```

You can check that the directory is mounted by typing:
```
$ mount | grep bbfs
```

Unmount the bbfs with: `$ fusermount -u my/mounted/directory`

You'll see that the same contents exist in both the root directory and the mounted directory. Supported VFS operations will be logged in a local file called `bbfs.log`.

Project Deliverables: For this assignment, modify the source code of bbfs to

- **(5 points)** enable truncation of the log file while the bbfs is mounted
  - i.e.
    `$ echo > bbfs.log`
    should cause all following writes to start from an empty file
- **(5 points)** ensure multiple filesystems can be mounted at the same time with your executable
- **(5 points)** centralize all logging to one file, for all filesystems mounted with the same executable
- **(15 points)** ensure logging is atomic and asyncronous; i.e. multiple logs do not get incorrectly interleaved and do not wait for writes to flush

In addition, please include a description of the software architecture of the logging implementation as well as the key data structures used. The discussion should also include the design challenges faced, and how you addressed them.

Submit your files as a single zip file to BBLearn.