CSC 501 Program: RAM Filesystem

Assignment:

Implement an in-memory filesystem (ie, RAMDISK) using FUSE.

Due:

See home page.

Description

In this program students will design, implement, and evaluate an *in-memory filesystem*.

FUSE

Modern operating systems support multiple filesystems. The operating system directs each filesystem operation to the appropriate implementation of the routine. It multiplexes the filesystem calls across many distinct implementations. For example, on a read system call the operation system uses NFS code if it is an NFS file but uses ext3 code if it is an ext3 file.

FUSE (Filesystem in Userspace) is an interface that exports filesystem operations to user-space. Thus filesystem routines are executed in user-space. Continuing the example, if the file is a FUSE file then operating system upcalls into user space in order to invoke the code associated with read.

RAMDISK

Students will create an *in-memory filesystem*. Instead of reading and writing disk blocks, the RAMDISK filesystem will use main memory for storage. (Note: When the main memory is over allocated, the operating system will page some memory out to disk. For this assignment, we still consider that in-memory.)

The RAMDISK filesystem will support the basic POSIX/Unix commands listed below. Externally, the RAMDISK appears as a standard Unix FS. Notably, it is hierarchtical (has directories) and must support standard accesses, such as read, write, and append. However, the filesystem is not persistent. The data and metadata are lost when the process terminates, which is also when the process frees the memory it has allocated.

The internal design of the filesystem is left to the student. For example, the Unix inode structure does not have to be mimicked; in fact, you need not have inodes at all. When RAMDISK is started it will setup the filesystem, including all metadata. In other words, there is no need to run *mkfs(8)*.

RAMDISK should not write any data to disk.

Basics for RAMDISK

Be able to run *postmark* in **unbuffered** mode. This requires supporting at least the following system calls.

- open, close
- read, write
- creat [sic], mkdir
- unlink, rmdir
- opendir, readdir

Limitations for RAMDISK

RAMDISK is **not** expected to support the following:

- Access control,
- Links, and
- Symbolic links.

Turn In:

Be sure to turnin **all** the files needed. Include a `Makefile` that creates the appropriate files. A appropriate penalty will be assessed if this is not so. Your Makefile should create an executable program named **ramdisk**, and this program can accept two parameters: 1) the directory to mount and 2) the size (MB) of your filesystem. To be specific, this following command will be used to mount a ramdisk:

```
ramdisk /path/to/dir 512
```

Also, denote any resources used--including other students--in a file named **REFERENCES**.

Resources

- FUSE
  - Home page
  - Wiki
  - FUSE on VCL
- Relevant man pages
  - close(2), ioctl(2), lseek(2), mmap(2), open(2), pread(2), readdir(4), readlink(2), write(2), fread(3)
- Useful Tools
  - strace (help you understand the FUSE operations required to implement)

Evaluation

The basic evaluation will use *postmark* and some other programs.

Notes:

- The code will be tested on the VCL image named here.
- You can save your postmark parameters into a configuration. For example,

./postmark benchmark.conf

- To support **vim**, you need to implement more FUSE operations; you are engouraged to give it a try if you would like to do so.

Extra Credit:

Make the filesystem persistent for 10% extra credit. Students that want their program to be graded for extra credit will submitted a file named “extra-credit-implemented”. (Please use this exact name so that your assignment does not require special handling.) Programs written for extra credit will accept an additional, optional argument that names a file.

ramdisk <mount_point> <size> [<filename>]

If the named file exists, then the image is read and loaded into the RAM filesystem at startup. If it does not exist, then start up a fresh (empty) filesystem. In either case, write the image to the named file when the RAM filesystem is unmounted. If students do not support extra credit, then their program should complain about too many arguments.

Grading:

The weighting of this assignment is given in policies.
