Project Description

CSE 536 Advanced Operating Systems

March 25, 2015

1

The project is to implement a filesystem using the C programming language and to test this filesystem for the following properties :

1. Error Free environment
2. Error Free environment with multi-threading(before or after atomicity)
3. Error-prone system without multi-threading
4. Error-prone system with multi-threading.

Head over to the book to chapter 9. Read the chapter thoroughly, paying special attention to sections surrounding the diagram (9.11)

Your job is to implement the entire "Journal Storage System". The "Cell Storage System" can be abstracted as a file with multiple sectors (Research on how large this can be and how many sectors are allowed specific to the underlying OS you are using). Then Implement the "Read", "Write", "Allocate" and "Deallocate" procedures in the C programming language. (Create separate files for each of these procedures and compile them together.) The "Journal Storage manager" will be another application (program) that you will build which will utilize these procedure calls to modify the "Cell Storage System" and provide outward facing interface for the NEW_ACTION, READ_CURRENT_VALUE etc. that can be seen in the diagram. The "Journal Storage Manager" should have all the logic to run all the scenarios mentioned above and have support for multi-threading.

Continue reading the text to section 9.4 and further to gain more understanding of "Before or After Atomicity" and implement a multi-threaded environment (using p-threads for example) to test the scenarios 2 and 4.

The details of what it means to have an "Error-free system" for multi-threaded and single threaded is up to you to determine according to what is discussed in the book and what we discussed in class, but please do state all your assumptions. Your system will be tested using automatic software to account for most common errors.

Figure 9.11 illustrates the overall structure of such a journal storage system,

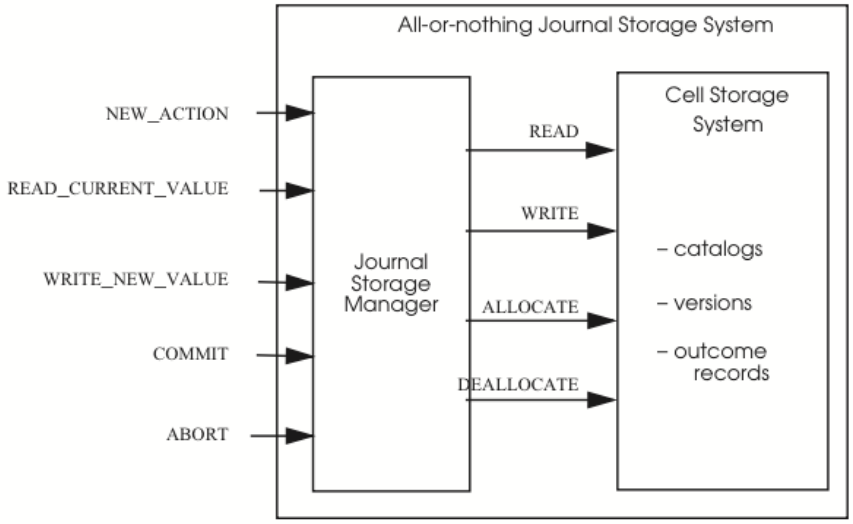

Figure 9.11: Interface to and internal organization of an all-or-nothing storage system based on version histories and journal storage.

Figure 1: Journal Storage System

For number 3, 4 you will demonstrate how your system will perform when there are failures, for example when the system crashes. You can simulate this by intentionally introducing errors in the procedures. As discussed in the chapter and in class (and in the previous assignment) your job here is to showcase the system's recovery mechanism.

The actual details of the implementation are open except for the following:

1. I will be using Ubuntu Distribution 14.04.2 LTS for testing so it is advisable to use the same to code. Ubuntu Install page
2. All programs are to be written in C.
3. All procedures that the "Journal Storage Manager" utilizes should be separate programs that are compiled together.
4. The programs you use to introduce faults should be named with a "Faulty" prefix to the name.

Please state all your assumptions.

2 Deliverables

1. An archive file (.zip) containing all source code : .ZIP
2. The output of the console while running the code. : .txt / .pdf
3. A project report detailing on the implementation procedures, Test cases, results, discussion and output screenshots, summary, conclusion and lessons learned. : .PDF (I will provide some common test cases early next week)

3 Useful Information

1. There will be a blackboard discussion board on this. All questions regarding the assignments should be posted there. While open discussion on ideas and issues is encouraged, sharing of any code is strictly prohibited.
2. A good refresher on the very powerful 'C' programming language can be found at GNU-C Manual. I would recommend using a Makefile or other build tool specially for ease in swapping in and out the "Faulty" programs
3. While the book is a good reference for this project, please don't hesitate to venture out of the book for ideas and possible tweaks. However be sure to point out any and all resources utilized in the references section in the project report.