# CSE 536 – Operating Systems
## Project Report On Journal File SYSTEM PART 1

**Project Report# 1**
**Assigned Date:** Mar 25, 2015
**Due Date:** Apr 10, 2015
**ASU ID :** 1207663258


| | |
|---|---|
| *Last Name* | REKALA |
| *First Name* | KRANTHI KUMAR |



**Abstract:**

The Project is aimed at implementing a File System called Journal File System Using the C programming language on Linux Platform. This part of the project takes into assumption that the Environment in which the program runs is an "Error – Free". This project is aimed at testing the concept of All-or-Nothing Atomicity in case of a single threaded operation.


**Background:**

As part of CSE536 – Advanced Operating Systems course many important and distinguished concepts are practically learnt and implemented. One of the most important concepts discussed are about atomicity and File Systems functioning. In order to combine these to vital concepts we have agreed upon designing a File System named Journal File System which helps us to gain a deeper insight into these two concepts. The present project is aimed at developing a JFS (Journal File System) in an error-free environment.

**Analytical Process:**

The Analytical process started with referring to the JFS in the Text book "Principles of Computer Design" and an understanding of the inherent methods of READ,WRITE,ALLOCATE,DEALLOCATE is gained. In order to simplify the process of developing this File System, I have implemented static allocation of memory to the arrays that hold the data and hence all the Test cases developed should be tested in one go. Also, single Array is used to hold different data_id's and their respective values. A different array is used in a similar way to hold the outcome records using global variables so that these are accessible by all the methods.

An interface which has the methods NEW_ACTION, READ_CURRENT_VALUE etc. has been implemented as per the requirement.

**Implementation Procedures:**

The Following are the procedures that are to be implemented as part of the project:

READ ()

WRITE ()

ALLOCATE ()

DEALLOCATE ()

The Procedures ALLOCATE and DEALLOCATE have been not implemented as I have implemented the memory and cell as static variables, so all the required memory is pre allocated. Also, a single array each has been defined to hold outcome records, data_records. The pointers to these arrays are declared as global variables in the header file read.h which is an implementation of READ (). write.h is the Implementation of WRITE().

The Following are the arrays that are used to visualize the implementation. The Committed values are updated in the outcome_record and data_record while temp values are stored in a temp array.

| data_record | | |
|---|---|---|
| data_id | transaction_id | Value |
| a | 1002 | 150 |
| b | 1003 | 200 |
| c | 1004 | 175 |
| d | 1005 | 200 |
| a | 1006 | 500 |

| outcome_record | |
|---|---|
| data_id | transaction_id |
| a | 1002 |
| b | 1003 |
| c | 1004 |
| d | 1005 |
| a | 1006 |

**Test Cases:**

The Following are the assumptions with respect to the project I implemented:

All the allocations of variables is static and hence the test cases are to be implemented in one go

The values in table above have been already updated and committed into the database (refer table above)

Let's start the Test Cases:

**ID: Test Case 1**

Purpose: Test all-or-nothing atomicity

Prerequisite:

1. Read, Write, Commit Procedures have been implemented.

Steps:

0. Read information at a certain location from disc.

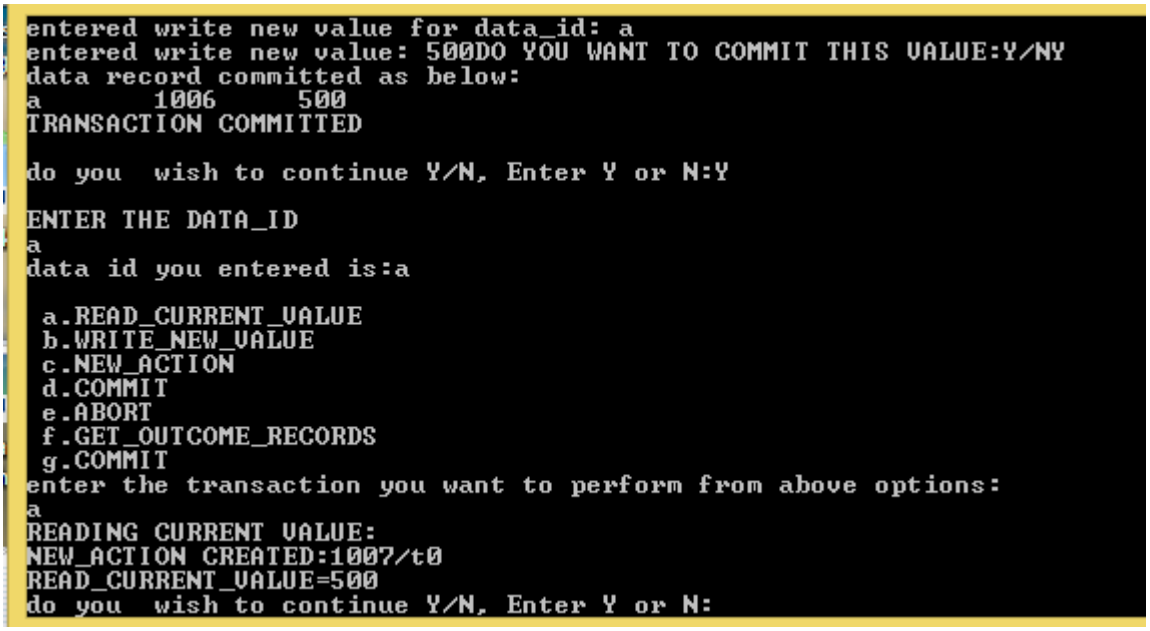
1. Write some information to that position in disc.

2. Abort the write request (i.e. do not commit)

3. Read from the write location

Expected:

1. The information that is read at Step 3. Should be the same as information in step 0.



```
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
a
READING CURRENT VALUE:
NEW_ACTION CREATED:10030
READ_CURRENT_VALUE=500
do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
b
ENTER THE VALUE TO WRITE:125
WRITING NEW VALUE
NEW_ACTION CREATED:10030
caller_id is 1003entered write value for id: 1003
entered write new value for data_id: a
entered write new value: 125DO YOU WANT TO COMMIT THIS VALUE:Y/NN

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
e
Enter the transaction id you want to ABORT1003
INVALID TRANSACTION IDTRANSACTION ID:1003 IS ABORTED
TRANSACTION ID:1003 IS ABORTED
INVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRAN
CTION IDINVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRANSACTION ID
do you  wish to continue Y/N, Enter Y or N:
```

```
b
ENTER THE VALUE TO WRITE:125
WRITING NEW VALUE
NEW_ACTION CREATED:10030
caller_id is 1003entered write value for id: 1003
entered write new value for data_id: a
entered write new value: 125DO YOU WANT TO COMMIT THIS VALUE:Y/NN

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
e
Enter the transaction id you want to ABORT1003
INVALID TRANSACTION IDTRANSACTION ID:1003 IS ABORTED
TRANSACTION ID:1003 IS ABORTED
INVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRANSA
CTION IDINVALID TRANSACTION IDINVALID TRANSACTION IDINVALID TRANSACTION ID
do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
a
READING CURRENT VALUE:
NEW_ACTION CREATED:10040
READ_CURRENT_VALUE=500
do you  wish to continue Y/N, Enter Y or N:
```

Outcome:

PASS/FAIL.

PASS

ID: Test Case 2

Purpose: General Read-Write

Prerequisite:

1. Read, Write, Commit Procedures have been implemented.

Steps:

0. Read information at a certain location from disc.

1. Write some information to that position in disc.

2. Commit the write request.

3. Read from the write location.

Expected:

1. The information that is read at Step 3. Should be the same as information that was written.

```
enter the transaction you want to perform from above options:
b
ENTER THE VALUE TO WRITE:500
WRITING NEW VALUE
NEW_ACTION CREATED:10020
caller_id is 1002entered write value for id: 1002
entered write new value for data_id: a
entered write new value: 500DO YOU WANT TO COMMIT THIS VALUE:Y/NY
data record committed as below:
a       1002     500
TRANSACTION COMMITTED

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
b
ENTER THE VALUE TO WRITE:125
WRITING NEW VALUE
NEW_ACTION CREATED:10030
caller_id is 1003entered write value for id: 1003
entered write new value for data_id: a
entered write new value: 125DO YOU WANT TO COMMIT THIS VALUE:Y/NY
data record committed as below:
a       1003     125
TRANSACTION COMMITTED

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
a
READING CURRENT VALUE:
NEW_ACTION CREATED:10040
READ_CURRENT_VALUE=125
do you  wish to continue Y/N, Enter Y or N:
```

Outcome:

PASS/FAIL.

PASS

ID : Test Case 3

Purpose: General Read Write

Prerequisite:

1. Read, Write, Commit Procedures have been implemented.

Steps:

1. Write some information to a position in disc.

2. Commit the write

3. Write another information in the same position (same sector and address if implemented as such)

3. Read from the write location.

Expected:

1. The information that is read at Step 4. Should be the result of the most recent write.

```
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
b
ENTER THE VALUE TO WRITE:500
WRITING NEW VALUE
NEW_ACTION CREATED:10020
caller_id is 1002entered write value for id: 1002
entered write new value for data_id: a
entered write new value: 500DO YOU WANT TO COMMIT THIS VALUE:Y/NY
data record committed as below:
a       1002     500
TRANSACTION COMMITTED

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
b
ENTER THE VALUE TO WRITE:125
WRITING NEW VALUE
NEW_ACTION CREATED:10030
caller_id is 1003entered write value for id: 1003
entered write new value for data_id: a
entered write new value: 125DO YOU WANT TO COMMIT THIS VALUE:Y/NN

do you  wish to continue Y/N, Enter Y or N:Y

ENTER THE DATA_ID
a
data id you entered is:a

 a.READ_CURRENT_VALUE
 b.WRITE_NEW_VALUE
 c.NEW_ACTION
 d.COMMIT
 e.ABORT
 f.GET_OUTCOME_RECORDS
 g.COMMIT
enter the transaction you want to perform from above options:
a
READING CURRENT VALUE:
NEW_ACTION CREATED:10040
READ_CURRENT_VALUE=500
do you  wish to continue Y/N, Enter Y or N:
```

Outcome:

PASS/FAIL.

PASS

Output Screenshots:

Output Screenshots are provided in above test cases.

**Summary:**

- An in depth understanding of JFS File System has been understood
- The Procedures for READ and WRITE procedures have been implemented
- A clear understanding of COMMIT and ABORT procedures has been gained
- Cell Storage Working has been understood

**Conclusion & lessons learned:**

The same File System implementation is to be implemented using multiple threads, Hence an understanding of the JFS in the multiple thread and error prone systems is to be understood as a future enhancement.

The Working of multiple WRITE and READ operations in a File System is understood

The procedure to compile multiple header files using gcc is learnt.

**Deliverables:**

Code, Report, Output