CS4210: Project 4

Chris Gordon
Jonathan Egbert

For project 4, we implemented recoverable virtual memory in the spirit of the Lightweight Recoverable Virtual Memory paper. We implemented an API that uses transactions as a system to facilitate aborting memory writes, and recovering commited data after a crash. Calling about_to_modify will save existing data in a region to memory, so that upon a transaction abort, the data can be rolled back to the state it was in before the transaction began. Committing a transaction results in the segments involved being committed to a log, to be later written to a backing store. Calling truncate or mapping a dirty segment with an existing backing store results in the relevant logs being applied to their backing store.

Our implementation keeps a list of active rvm directories, and each rvm_node contains a linked list of all of the mapped segments. Each segment node contains a stack of the regions that have been saved at each about_to_modify call. Each segment can only be associated with one transaction at a time and begin_transaction will fail if any of the segments is already involved in another transaction. Upon starting a transaction, our library will iterate through transaction_ids until it finds the first available one and assigns it to the relevant segments. Each segment that has been recently committed in a transaction has its own log file, which contains a size (the size of the segment) and then the contents of of the segment at the time of the commit.

To run the code, just run make in the project directory. From there, just run the test file executables from the same directory. For example, running the basic test files would require the following:

```
make
./basic
```