Team members:
Shruti Padamata (GT Id: spadamata3)
Archana Venkatesh (GT Id: avenkatesh8)

Overview:
rvm_init creates a directory with the given name, which is used to store all the persistent segment files and redo log files during the execution of the program. This data persists even after program termination and can be seen during a subsequent invocation on using the same directory name in rvm_init. The segment files are created during rvm_map if not already existing. Transactions are allowed only on mapped segments. Calls to rvm_begin_trans / rvm_commit_trans / rvm_abort_trans / rvm_unmap would fail if the segment is not mapped and the program will terminate with an assertion failure. Similarly, rvm_destroy will fail if the segment being destroyed was not unmapped before. All the other error conditions like malloc failures or file operation failures also result in termination of the program with an assertion failure. If it is required to disable the asserts for any reason, please add the line '#define NO_ASSERT' at the beginning of the file rvm.cpp.

How you use logfiles to accomplish persistency plus transaction semantics:

There are two kinds of logs that we maintain, similar to those described in LRVM paper:
- Undo logs that are only in memory
- Redo logs that are only maintained on disk

At the beginning of a transaction (rvm_beign_trans), a copy of each segment involved in the transaction is stored in per segment undo logs. These logs are maintained only for the duration of the transaction i.e., till the call to either rvm_commit_trans or rvm_abort_trans. The log memory will be freed on a commit and on an abort, the contents of the log will be copied into the data part of in-memory segment data structure, before freeing. The changes done in the transaction happen only on the in-memory segments. Thus we guarantee that an abort of the transaction would completely revert the changes to the segments. The redo logs explained below are used for ensuring persistency.

What goes in them? How do the files get cleaned up, so that they do not expand indefinitely?

The redo logs are also maintained per segment. Each log has multiple records, each corresponding to one call to rvm_about_to_modify. Each record contains the offset into the segment, the size of the modified part of segment and the new data to be written to that part of segment. We maintain in-memory data about the offset and size of modifications that happen during a transaction and write the log records in rvm_commit_trans using them.

The rvm_truncate_log function reads the redo log and writes the updated segment data into the external segment file. This function has to be called explicitly to make sure that the file gets cleaned up. There is also a rvm_truncate_segment function to truncate a given segment. This function is called during the map primitive if the segment already has a corresponding redo log file, so as to ensure that it will get the latest updated data of the segment.