Project 4: Recoverable Virtual Memory

Zheng Yong, Xuan Jiang

1. **Data Structure**

   Struct rvm_info contains:

   a. **map** which stores the mapping between memory address and disk file name.
   b. **busy** is another map which records the status of current segments, whether is busy(1) or not(0).
   c. **directory:** a variable stores the directory of disk files

   rvm: the pointer type of rvm_info struct

   transaction_struct:

   a. **numsegs:** a variable stores the number of segments specified in begin transaction
   b. **segbases:** a pointer points at the beginning segments array
   c. **rvm:** store the current rvm information during the same transaction
   d. **modified_segs:** a vector stores all the segments being modified, **offset:** a vector stores the starting points of corresponding segments, **size:** a vector stores the size modified of corresponding segments

   trans_t: the pointer type of transaction_struct

   verbose_enabled: a global variable which indicates enable verbose or not.

2. **Design**
   a. When trying to map a disk file to some memory, store the mapping into map in **rvm** struct. And when unmap, remove the mapping item in map.
   b. During begin transaction, set segments involved in the transaction to be busy. And set them back to be not busy during release.
   c. When committing transaction, open a log file and write information into the log file, and the format is as follows:

      segment_name offset size

content_been_modified

eg.

testseg 0 100
hello, world

d. When truncating log, shrink the log file

3. **Test Cases:**

test.cpp: test simple functions and test rvm_verbose()
crash.c: simulate the scenario when a process crashed before commit or abort
map-test-case.cpp: trying to detect:
1.destory before unmap test cases
2.map a segment already exits but with smaller size
map-twice.cpp: detect test case when trying to map the same segment twice