**Project 4: Recoverable Virtual Memory**

**Zheng Yong, Xuan Jiang**

1. **Data Structure**

**Struct rvm\_info** contains:

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

**rvm\_t**: the pointer type of rvm\_info struct

**transaction\_struct:**

1. **numsegs:** a variable stores the number of segments specified in begin transaction
2. **segbases:** a pointer points at the beginning segments array
3. **rvm:** store the current rvm information during the same transaction
4. **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.

rvm\_t

unordered\_map<string, void\*>\* map;

unordered\_map<void\*, int>\* busy;

string directory;

trans\_t

int numsegs;

void\*\* segbases;

rvm\_t rvm;

vector<void\*>\* modified\_segs;

vector<int>\* offset;

vector<int>\* size;

rvm\_t

trans\_t3

trans\_t2

trans\_t1

1. **Implementation**
   1. When trying to map a disk file to some memory, store the mapping into map in **rvm** struct. If there is a corresponding file we call rvm\_truncate\_log to update the content of the file and then load the file into memory. If there is not, we create a new empty file that matches the size. If the file exists and the size does not match we extend the file size to the given parameter. When unmap, remove the corresponding item in map. And when destroy is called, delete the file on disk.
   2. During begin transaction, set segments involved in the transaction to be busy to indicate that the memory is being used. When begin is called the busy variable is checked first for all the segments in the parameter. If any of them is busy, we do not begin the transaction (return -1).
   3. When about\_to\_modify is called, first we check if the segment is in the given trans\_t. If it is not, an error occurs. Then we initialize the modified\_segs variable along with size and offset variables.
   4. 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

* 1. If abort is called, the current transaction is aborted. Since the memory is not destroyed, we should restore it from the file. And before doing this we should check the log to see if there is any updates. So rvm\_truncate\_log is called and then the content is updated from the backup file.
  2. When truncating log, shrink the log file to update all the backup files.

1. **Additional 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