## CMPS 181, Spring 2015

## Course Project Environment Info for Project 1 (and for later Projects too)

In project 1, you will implement a very simple paged file (PF) manager. It builds up the basic file system required for continuing with projects 2, 3 and 4. The PF component provides classes and methods for managing files and pages in files. In addition, you will implement the first few operations for a record-based file manager (which you will continue working on in part 2 of the project) on top of the basic paged file system. This document aims at providing you with the necessary information required to start project 1.

## Goal

The goal of project 1 is threefold:

* Getting familiar with a C/C++ development environment
* Implement a simple paged file system.
* Implement a few operations of a record-oriented (also known as tuple-oriented) file system.

The detailed description of the project is in the file: CMPS181\_Project1\_Introduction.docx

## Overview of Steps

1. Development environment
2. Download and deploy the codebase of Project 1
3. Finish the development of Project 1

## Detailed Instructions

1. **Development environment**

* You may develop your code on any system you like using the steps below, but please test that it works on **unix.ucsc.edu**, which is where it will be tested. It may simplify things for you if you develop on that machine.

**2. Download and deploy the codebase of Project 1**

* **Download the codebase of Project 1**

Please download the codebase.zip file [​](https://grape.ics.uci.edu/wiki/asterix/raw-attachment/wiki/cs222-2014-fall-project1/codebase.zip) onto your own computer. Unzip the file.

* **Deploy the codebase**

Read the readme.txt under ./codebase/.

Go to the codebase, and modify the CODEROOT in makefile.inc properly.

Go to folder "rbf", and type in:

make clean

make

./rbftest

You will be able to see the output.

**3. Finish the development of Project 1**

We have seen the results of running the code in codebase. But since the implementation of methods is empty, you cannot manage any file yet. Please finish the implementation in pfm.cc and the following methods in rbfm.cc (besides the constructor and destructor): 1) insertTuple. 2) readTuple. 3) printTuple. The remaining methods are not required for part 1 of the project; instead you will implement them as part of part 2 of the project. Please write your own test cases in rbftest.cc.

You may find these functions useful: [​http://www.cplusplus.com/reference/clibrary/cstdio/](http://www.cplusplus.com/reference/clibrary/cstdio/)

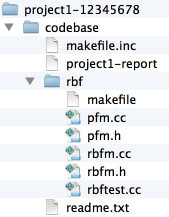
## Submission Instructions

The following are requirements on your submission. **Points may be deducted if they are not followed.**

* Write a report to briefly describe the implementation of your paged file and record-based file systems.
* You need to submit the source code under the "rbf" folder. Make sure you do a "make clean" first, and do NOT include any useless files (such as binary files and data files). You should make sure your makefile runs properly.
* Please organize your project in the following directory hierarchy:

project1-studentID / codebase / {rbf, makefile.inc, readme.txt, project1-report} where rbf folder include your source code and the makefile.

(e.g. project1-12345678 / codebase / {rbf, makefile.inc, readme.txt, project1-report})

Your folder structure should look like this picture:   
[](https://grape.ics.uci.edu/wiki/asterix/attachment/wiki/cs222-2014-fall-project1/final_folder_structure_example.jpg)

* Compress project1-studentID into a SINGLE zip file, with the name "project1-studentID.zip" (e.g. project1-12345678.zip).
* Put test.sh and the zip file under the same directory. Run it to check whether your project can be properly unzipped and tested (use your own makefile.inc and the rbftest.cc when you are testing the script). If the script doesn't work correctly, it's most likely that your folder organization doesn't meet the requirement. Our grading will be automatically done by running script. The usage of the script is:

./test.sh ''project1-studentID''

* **IMPORTANT: make sure the script works on a Linux machine(or virtual machine), because readers will use a Linux machine to grade the assignment.**
* Upload the zip file "project1-studentID.zip" to ECommons.

## Testing

Please use the code tests rbftest.cc, rbftest11a.cc and rbftest11b.cc to test your code. Note that other test cases may be used as well to grade your project. This is by no means an exhaustive test suite. Please feel free to add more cases to this, and test your code thoroughly.

## Grading Rubrics

Please see the grading rubric  <<grading rubric to be added>>.

## Q & A

* **Q1**: In a case where Page 1 and 3 of the file are written in completely and Page 2 is partially filled and the user wants to append data on page 2. Now if the size of this data that he wants to write is more than the available space on Page 2, then what is the expected action to be taken. Do we just fit in whatever data we can and truncate the rest OR completely disallow the user to make such a write?   
  **A**: AppendPage() always happens to the end of the file. The number of file bytes affected by each page operation is always PAGE\_SIZE.
* **Q2**: Is it fine if I do the file handling in C++ using the binary mode of read/write?   
  **A**: You should use the binary mode.
* **Q3**: Why is the access specifier of the constructor and destructor of the class PagedFileManager set to be "protected"?   
  **A**: The PagedFileManager is a singleton class, which means only one instance of PagedFileManager is allowed. You cannot instantiate the class by calling its constructor. Instead you should get an instance of the class by calling the Instance() function of PagedFileManager. The Instance() function has been implemented for you in pfm.cc. The same applies to the RecordBasedFileManager.
* **Q4**: As for pages, if I understand correctly, the Read/Write/AppendPage functions are operating on these files and if you want to write 3rd page (page number: 2) of a file, you'd seek 8K bytes into the file and start writing the data. Is this correct or am I misunderstanding the concept of pages?  
  **A**:
  + Read reads a page that has to exist
  + Append adds a page
  + Write overwrites a page that has to exist  
    To write to 3rd page of a file, the file should already have at least 2 pages (page number: 0,1) that contain valid data. Then you can either append the data to 3rd page if it doesn't exist, or overwrite the 3rd page if it already exists. Please do not leave "holes" in files by writing past EOF. We won't allow the case of appending garbage pages.
* **Q5**: Since I need to change the path of codebase in makefile.inc to test the project, do I need to change it back when I submit the zip file?   
  **A**: No, you don't need to change back, but you need to make sure the path is **relative**, so that the test.sh script can also work on another machine.
* **Q6**: Are we supposed to work on heap files? In particular, for inserting a tuple, do we only have to consider about insertion the new tuple at the end of the last page? Or instead, we have to be able to support insertion in wherever free spaces among all the pages?  
  **A**: You are supposed to insert the record on the first page with sufficient space available.
* **Q7**: What's the data format for insertTuple?   
  **A**: The format to insertTuple is as follows: Suppose you want to insert the following record: ("Tom", 25, "UCIrvine", 3.1415) Then the format of the record should be: [4 bytes for the length 3] [3 bytes for the string "Tom"] [4 bytes for the integer 25] [4 bytes for the length 8] [8 bytes for the string "UCIrvine"] [4 bytes for the float 3.1415] That is, for Int and Real type, the data length is NOT needed. Please follow this format.

## Platform Q & A

* **Q1**: Do I have to use a specific platform?   
  **A**: You can implement your code on any operating system where g++ works such as Windows, or Linux. However, you can not use any platform specific APIs. Please also note that readers will use a Linux machine to grade your code. Therefore, please make sure that your code works on Linux.