Programming Assignment 2: RAID-5 Simulator
CSC 4103: Operating Systems, Fall 2021

Due: November 22, Monday (by 11:59 PM)
Total Points: 10

Instructions:

(1) Compile and test-run your code on the classes server.

(2) Include a README file with your code.

(3) Submit your work and verify your submission on the classes server.

(4) Submit a tar ball or a Zip file of your code on Moodle for backup.

Late submissions will be penalized at the rate of 10% per day late and no more than 3 days late.

Objective

To implement a simulator of RAID-5 storage.

Background

RAID is a widely used storage technology for enhancing performance and reliability. RAID-5 uses block-level striping and block-interleaved distributed parity blocks. By evenly distributing data blocks and parity blocks across multiple disks, the overall performance and reliability can be improved.

Programming Language

C/C++ or Java.

Programming Task

Write a program that simulates the structure of RAID-5. Assuming an RAID-5 storage consisting of *N* disk drives, each stripe (a row) has *N* blocks (stripe unit), including *N*-1 data blocks and 1 parity block. The parity block can be calculated by using the XOR operation. For example, if we have 5 disk drives, and a stripe has 4 data blocks, A, B, C, D, and the parity block P. P can be calculated by XORing the four data blocks ($P=A \oplus B \oplus C \oplus D$). If any disk is lost, the blocks on the lost disk can be recovered by using the other disks. For example, if Disk #0 is lost, we can reconstruct block A by XORing the other 4 blocks ($A=B \oplus C \oplus D \oplus P$). In this program, you need to use files to simulate the disks. Given an input byte stream, use the RAID-5 structure to decide how to split and store the byte stream over the "disks". If one disk fails, your program should be able to reconstruct the lost "disk".

Detailed Requirements

The program should accept the following input parameters to specify:

a. The number of member disks.
b. The block size (in bytes).
c. The command (write, read, or rebuild).
d. The file name (input, output, or disk).

(1) Write data into the RAID-5 storage.

Example: `$./raid 4 16 write ./input`

This example command means that the RAID-5 storage has 4 member disks, the block size is 16 bytes, and write the byte stream stored in file ./input into the RAID.

Your program should generate 4 "disk" files (disk.0, disk.1, disk.2, and disk.3) to simulate the 4 member disks of the RAID.

Note: If the input cannot fully occupy a stripe (the last one), you may pad the stripe with 0s. Using the example above, each stripe can store 3 data blocks, which is 48 bytes in total. Assuming the input is 64 bytes, it will use two stripes: The first stripe is completely full, storing the first 48 bytes, and the second stripe is partially full, storing 16 bytes. The remaining 32 bytes in the second stripe are unused. Since calculating the parity block needs three data blocks, you may fill the unused 32 bytes with 0s.

(2) Read data from the RAID-5 storage.

Example: `$./raid 4 16 read ./output`

This example command means that the RAID-5 storage has 4 member disks, the block size is 16 bytes, and read the stored byte stream from the RAID into file ./output.

(3) Reconstruct data in the RAID-5 storage.

Example: `$./raid 4 16 rebuild ./disk.0`

This example command means that the RAID-5 storage has 4 member disks, the block size is 16 bytes, and rebuild the lost "disk" (Disk #0) in file ./disk.0.

NOTE:

(1) Compile and test-run your code on the classes.csc.lsu.edu server. **Your code should be compilable and runnable in the Linux environment of the classes server.** Windows code will NOT be accepted.

(2) Include an README file to clearly explain how to compile and run your code, such as the command, argument, expected input and output, etc. The README file should also include your full name, LSU ID, email address, and the classes server login ID.

(3) Submit your work by the deadline on both classes server and Moodle. Verify your submission, which will display your submission date/time. Note that if you make multiple submissions, the prior submission will be overwritten. The final submission date/time will be the latest one.

(4) Late submissions will be penalized by 10% per day late and no more than 3 days late.

Submitting Your Work

You need to submit a complete package on **both classes server and Moodle.** The grader will grade your code on the classes server. Your submission on Moodle will be used as a backup. If you made multiple submissions, the final submission date/time will be the latest one.

All files you submit must have a **header** with the following (enclosed in comment lines):

Name:	Your Name (Last, First)
Email:	Your LSU email
Project:	PA-2 (RAID)

Instructor:	Feng Chen
Class:	cs4103-au21
Login ID:	cs4103xx

(1) Submission to the classes server

You need to use the server "classes.csc.lsu.edu" to work on the assignment. You can login to your account in the server using SSH. Create a directory **prog2** (by typing **mkdir prog2)** in your home directory in which you create your program or source code.

Make sure that you are in the **prog2** directory while submitting your program. It is suggested to pack all the files in a tar tall (or a Zip file) before submission. Submit your assignment to the grader by typing the following command:

$ p_copy 2

This command copies everything in your **prog2** directory to the grader's account.

Verify that all the files have been submitted successfully:

$ verify 2

(2) Submission on Moodle

You should create a tar ball (or a Zip file) of your complete code and submit it on Moodle.

Note: The submission on Moodle will be only used as a backup. You still need to submit on the classes server as instructed above.