CS 558: Computer Systems Lab Assignment
Deadline: 22 April 2021

Instructions:

- This assignment is to be done in groups. Continue with the same group partner you had for the last assignment.
- The programs can be written in C/C++/Java.
- No extensions in submission are allowed. Delay in submission will lead to penalty in marks.
- Assignments submitted before the deadline will only be considered for evaluation.
- Please do not email your assignments separately to the TAs, it will not be considered for evaluation.
- Your code will be checked for plagiarism. Any kind of academic dishonesty, plagiarism, etc. will lead to penalties.
- No sharing of code between students, submission of downloaded code (from the Internet, Campus LAN, or anywhere else) is allowed.
- The first instance of code copying will result in ZERO marks for the assignment.
- The second instance of code copying will result in a `F' grade. Students may also be reported to the Students Disciplinary Committee, which can impose additional penalties.

1. Consider a scenario where you have N number of boxes and M number of slots (M<N). Each box is having their identification number (more than one box can have the same identification number). Each slot is having two parts, one is for accommodating the box and another is for reference bits (this is for referring to the boxes).

You need to accommodate the boxes into the slots exactly once.

Condition:

- Initially all slots will be empty and the reference bit field will be 0.
- if reference bit is 0, replace
- if 1, give box a 2nd chance and move onto next FIFO box; reset reference bit to 0 and reset arrival time to current time

- a box given a second chance will not be replaced until all other boxes have been replaced (FIFO) or given second chances
- if a box is used often enough to keep its reference bit set, it will never be replaced.

Implementation: the clock algorithm using a circular queue

  - a pointer (clock) indicates which box is to be replaced next
  - when a box is needed, the pointer advances until it finds a box with a 0 reference bit
  - as it advances, it clears the reference bits
  - once a victim box is found, the box is replaced, and the new box is inserted in the circular queue in that position
  - degenerates to FIFO replacement if all bits are set

Input:

- Number of blocks
- Number of slots

Output:

- Number of replacements.
- Vary number of slots and show the number of replacements
- List of blocks, which got replaced

2. This assignment is for you to simulate page-replacement algorithms and compare their performances. Your simulations should be able to take in the required parameters from a data file in the following format:

   - The first line of the file contains two integers in this order: the number of pages in the reference string, and the number of page frames.

   - The rest of the file contains page reference string as a sequence of integers that are the page numbers (in virtual address space) accessed by the process in sequence.

The simulator will simulate the behavior of the following page-replacement algorithms on the reference string, and report the number of page faults generated for each algorithm:

FIFO
LFU
LRU

The simulator should take the following command-line arguments (in this sequence): the name of the data file, a sequence of strings (maximum three) from the following sets: FF, LF, LR. The simulator simulates only the algorithms specified in the command line. If no algorithm is specified, all three are simulated.

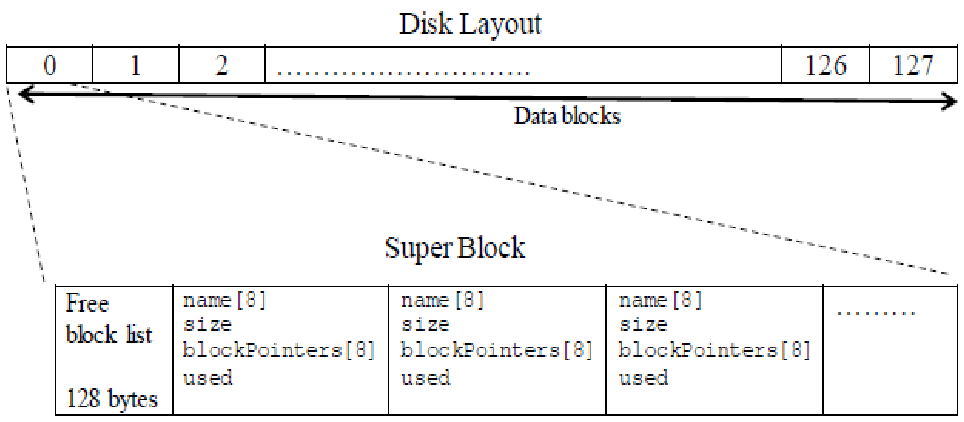
3. This assignment is to design a simple File System which makes the following assumptions:
   - The File system resides on a disk that is 128 KB in size.
   - There is only one root directory. No subdirectories are allowed.
   - The File system supports at most 16 files.
   - The maximum size of a file is 8 blocks where each block is 1KB in size.
   - Each File has a unique name.

   The layout of 128 KB disk is as follows:

   - The first 1KB block is called the super block. It stores the free block list and index nodes (inode) for each file.
   - The remaining 127 1KB blocks store data blocks of the files on your file system.
   - The exact structure of the super block is as follows:
   - The first 128 bytes stores the free block list. Each entry in this list indicates whether the corresponding block is free or in use (if the i-th byte is 0, it indicates that the block is free, else it is in use). Initially all blocks except the super block are free.
   - Immediately following the free block list are the 16 index nodes, one for each file that is allowed in the file system. Initially, all inodes are free. Each inode stores the following information:

     ```
     char name[8]; //file name
     int size;     // file size (in number of blocks)
     int blockPointers[8]; // direct block pointers
     int used;             // 0 => inode is free; 1 => in use
     ```

Note that each inode is 56 bytes in size; Since you have 16 of these, the total size of occupied by the inodes is 896 bytes. The free/used block information (mentioned above) is 128 byes. So the total space used in the super block is 1024 bytes.



You need to implement the following operations for your file system.

- ***create(char name[8], int size)***: create a new file with this name and with these many blocks. (You can assume that the file size is specified at file creation time and the file does not grow or shrink from this point on)
- ***delete(char name[8])***: delete the file with this name
- ***read(char name[8], int blockNum, char buf[1024])***: read the specified block from this file into the specified buffer; blockNum can range from 0 to 7.
- ***write(char name[8], int blockNum, char buf[1024])***: write the data in the buffer to the specified block in this file.
- ***ls(void)***: list the names of all files in the file system and their sizes.