CS431 — Project 4

February 27, 2017

Due: Friday, March 10, 2017 (60 points)

Description

In this project you will be simulating a File Allocation Table based file system.

Your program should compile and run with just the following commands:

```
$ javac FileSystem.java
$ java FileSystem
```

File System

Your program should have the following three data structures:

1. An array of `int`s corresponding to the FAT. It will record the pointers for each block that is allocated to a file. A -1 should indicate the end of the linked list for a file.
2. A list of i-nodes where each i-node contains the file's name (no paths needed for this project) and starting block. You can also include the total number of blocks for the file if you desire.
3. A bitmap that tracks empty and allocated blocks in the file system where a 0 indicates a free block and a 1 indicates an allocated block.

Our simulated file system will only allow 64 blocks, i.e., the size of your FAT array should be 64. By doing this, the bitmap should be stored in a single `long`. Do not use something like a `boolean` array for the bitmap. You will need to use bitwise operators on a `long` variable to manage the bitmap.

Like the previous project, provide a command-line interface that allows a user to do the following:

1. `put name size`

   This command will attempt to put the file with the given name and size (in blocks) in to the file system. To do this, you must add an i-node and fill in the pointers for each block then update the bitmap. If there is already an i-node with this file name or if there are not enough available blocks, do not allow the operation. When filling in the blocks, search sequentially from the start of the file system and fill in as you find empty blocks.

2. `del name`

   This command should delete the file with the given name. To delete the file, remove the i-node and clear the bitmap for the appropriate blocks. It is not necessary to clear the FAT because the bitmap will indicate that the blocks are available and no i-nodes point to those blocks any more.

3. `bitmap`

   This command should print the bitmap as an 8×8 square of bits with each line labeled by the starting block number:

   ```
    0 00101000
    8 00000111
   16 00000000
   24 00000000
   32 01111000
   40 00000000
   48 00000110
   56 00000000
   ```

4. `inodes`

   This command should print all of the i-nodes and also the linked list of pointers from the FAT for each one. For example, if our i-node list has files `test1`, `test2`, and `test3`, the output might look like:

   ```
   test1: 13 -> 14 -> 15
   test2: 33 -> 34 -> 35 -> 36
   test3: 53 -> 54 -> 2 -> 4
   ```

   given the same allocations in the bitmap above where `test1` is using the sequential blocks in the second row, `test2` is using the sequential blocks in the fifth row, and `test3` is allocated in several places across the file system.

Your program should handle errors without crashing, e.g., no space to allocate, attempting to delete file that does not exist, or bad command given at command prompt.

Submission

1. On https://codebank.xyz, create a project named `CS431-P4`. Follow this naming convention precisely including case.
2. Create your own local repository by the following:
   (a) Navigate to your local directory for this project.
   (b) Run `git init` to initialize the repository.
3. Make sure you add the reference to the remote repository in your local repository with:

   ```
   $ git remote add origin https://codebank.xyz/username/CS431-P4.git
   ```

4. Your project should have a main class named `FileSystem` in a file named `FileSystem.java`. You can have other files or classes, but it should successfully compile and run by simplying using:

   ```
   $ javac FileSystem.java
   $ java FileSystem
   ```

You will lose points if your submission is not correct (e.g., incorrect repository name, file names, class names, or package declaration that causes the above commands to fail to run).