CPSC 457: Principles of Operating Systems

Assignment 5: FAT storage

Weight: 8%

Collaboration

Discussing the assignment requirements with others is a reasonable thing to do and an excellent way to learn. However, the work you hand in must ultimately be your work. This is essential for you to benefit from the learning experience and for the instructors and TAs to grade you fairly. Handing in work that is not your original work but is represented as such is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code you hand in that are not your original work. You can put the citations into comments in your program. For example, if you find and use code found on a website, include a comment that says, for example:

   # The following code is from https://www.quackit.com/python/tutorial/python_hello_world.cfm.

   Use the complete URL so that the marker can check the source.

2. A tool like chat-GPT can be used to improve small code blocks. For example, three lines of code. If you get help from code assistance like Chat-GPT, you should comment above the block of code you requested assistance on debugging or improving and cite the tool used to get that suggestion. Using a tool like chat-GPT to write the majority of your assignment requirements will be treated as plagiarism if found without citation, and with citation, it will be treated as 0 for the component the student did not complete. Code improvement of short length will get credit if commented/cited properly.
3. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. **However, you may still get a low grade if you submit code not primarily developed by yourself. Cited material should never be used to complete core assignment specifications unless clearly approved. Before submitting, you can and should verify any code you are concerned about with your instructor/TA.**
4. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code, it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's screen, this code is not yours.
5. **Collaborative coding is strictly prohibited. Your assignment submission must be strictly your code**. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing the code itself, or modelling code after another student's algorithm. **You can not use (even with citation) another student's code.**
6. Making your code available, even passively, for others to copy or potentially copy is also plagiarism.
7. We will look for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
8. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor for help rather than plagiarizing. A common penalty is an F on a plagiarized assignment.

Late Policy

All students will have 6 total days during the semester that can be used for none, one, some, or all the 6 assignments. Students can use these days all on one assignment, or some on each of the six.

Each 24-hour period late after an assignment deadline counts as one full day regardless of how many hours the assignment was late within that period. For example, deadlines are generally Fridays 11:59pm local time. That means an assignment submitted any time Saturday before 11:59pm local time will be considered as 1 day late and count against the students 5 total days.

As long as a student still has days left to use, their assignment will be graded without penalty. If a student has no more days left or their submission exceeds the days they have remaining, then they will receive a 0 grade for a late assignment. Students who use up late days on earlier assignments will not have them available for later assignments.

TAs will indicate the student's late day usage in grading feedback.

Due date is posted on D2L. Your D2L submission should include the files requested and a link to a Gitlab repository you used while completing the assignment with your TA added as a **Developer** role. Help with Gitlab Clone/Developer role access is available in D2L video.

Q1. Programming question [30 marks]

For this question you will implement a program (fat.cpp) that will check the consistency of a file allocation table with respect to the entries of a directory. Your program will read input from standard input and will output results to standard output.

Input

The input will contain a simplistic representation of a filesystem. It will contain the following, all separated by white space:

- block size – an integer in the range [1, 1024]
- number of entries in the directory – an integer in the range [0, 50]
- number of entries in FAT – an integer in the range [1, 200000]
- list of directory entries, one entry per line, each line containing:
  - filename – a string of up to 128 characters, any chars allowed except white space
  - pointer to the first block – an integer in range [-1, 200000], where '-1' denotes a NULL pointer, '0' denotes first block, etc.
  - file size in bytes – an integer in range [0, 2^{30}]
- FAT entries – a list of integers separated by white space
  - each entry represents a pointer to the next entry in the FAT

- each entry is an integer in a range [-1, 200000]
- -1 denotes a NULL pointer (end of chain)
- the entries in FAT are numbered starting from 0

Sample input file test.txt:

```
10 3 11
A.jpg 0 31
B.txt 6 23
C.zip -1 0
5 9 5 3 -1 1 8 0 6 -1 0
```

The above input describes a filesystem that has a block size of 10, contains FAT with 11 entries, and holds 3 files: A.jpg, B.txt and C.zip.

File A.jpg contains 31 bytes, and it is stored in blocks {0, 5, 1, 9}. It has the correct number of blocks, contains no cycles, and does not share blocks with any other file.

File B.txt has 23 bytes, and it is stored on blocks {6, 8}. The blocks belonging to file B.txt form a cycle, which is a problem that your program will need to detect. File B.txt also has an incorrect number of blocks for its size, which is another problem your program needs to report. File B.txt does not share blocks with any other file. If it did, you would need to detect and report that as well.

File C.zip is empty. No blocks are allocated to this file. There are no problems with this file.

Finally, the total number of unused blocks on the filesystem is 5. This is a number you will need to calculate and report.

Output

After reading the input, your program will check the consistency of the filesystem. For every file you need to determine whether there are any issues with that file. You need to check for the following issues:

- Does the file contain the right number of blocks, or are there too many or too few?
- Do the blocks allocated to the file contain a cycle?
- Does the file share its blocks with any other file?

You then need to report all issues you found for every file to standard output.

You will also determine how many of the blocks are unused in the filesystem. Unused blocks are the ones not allocated to any file.

Starter code

Start by downloading the following starter code:

```
$ git clone https://csgit.ucalgary.ca/jwhudson/cpsc457w24.git

$ cd cpsc457w24/fat

$ make
```

You need to implement **checkConsistency()** function by modifying the **fat.cpp** file. **Only modify** file **fat.cpp,** and **do not** modify any other files.

The included driver (**main.cpp**) will read edge descriptions from standard input. It parses them into input, calls your **checkConsistency()** and finally prints out the results. Here is how you run it on file **test.txt**:

```
$ ./fat < test.txt
Issues with files:
  A.jpg:
  B.txt: not enough blocks, contains cycle
  C.zip:
Number of free blocks: 5
```

Submission

Submit 1 file for this assignment to D2L:

- Your solution to Q2 in a file called **fat.cpp.**

Please note – you need to **submit all files every time** you make a submission, as the previous submission will be overwritten.

Submit this as a separate file. **Do not** submit an archive, such as ZIP or TAR. If you submit an archive, you will receive a penalty.

Submit the web address of your Gitlab repository you used for your assignment (I recommend one repository for all 6 of your assignments that you can re-use). Your TA must be added as a Developer. There are penalties for submissions without the ability to access the corresponding students Gitlab by the TA.

While the starter code contains many different files, the only file you are allowed to modify is **fat.cpp.** Do not modify any other files. All code you write must go in the **fat.cpp** files, and that should be the only file you will submit for grading. We will test your code by supplying our own **main()** function, which will be different from the **main()** function in the starter code. It is therefore vital that you maintain the same function signature as declared in **fat.h.** Before you submit **fat.cpp** to D2L, make sure it works with unmodified files from the starter code!

General information about all assignments:

All assignments are due on the date listed on D2L. Late submissions without remaining late days banked will not be marked.

1. Extensions beyond the late day policy can be discussed more than 5 business days in advance and are granted only by the course instructor.
2. After you submit your work to D2L, verify your submission by re-downloading it.
3. You can submit many times before the due date. D2L will simply overwrite previous submissions with newer ones. It is better to submit incomplete work for a chance of getting partial marks, than not to submit anything. Please bear in mind that you cannot re-submit a single file if you have already submitted other files. Your new submission would delete the previous files you submitted. So please keep a copy of all files you intend to submit and resubmit all of them every time.
4. Assignments are likely going to be marked by your TAs. If you have questions about assignment marking, contact your TA first. If you still have questions after you have talked to your TA, then you can contact your instructor.
5. All programs you submit must run on **linux lab** or **cslinux.ucalgary.ca.** If your TA is unable to run your code on these, you will receive 0 marks.
6. Unless specified otherwise, you must submit code that can finish on any valid input under 10s on **linux lab** or **cslinux.ucalgary.ca (will be slower)**, when compiled with **-O2** optimization. Any code that runs longer than this may receive a deduction, and code that runs for too long (about 30s) will receive 0 marks.
7. **Assignments must reflect individual work.** Here are some examples of what you are not allowed to do for individual assignments: you are not allowed to copy code or written answers (in part, or in whole) from anyone else; you are not allowed to collaborate with anyone; you are not allowed to share your solutions (code or pseudocode) with anyone; you are not allowed to sell or purchase a solution; you are not allowed to make your code available publicly (e.g. via public git repositories). This list is not exclusive. For further information on plagiarism, cheating and other academic misconduct, check the information at this link: http://www.ucalgary.ca/pubs/calendar/current/k-5.html .
8. We will use automated similarity detection software to check for plagiarism. Your submission will be compared to other students (current and previous), as well as to any known online sources. Any cases of detected plagiarism or any other academic misconduct will be investigated and reported.