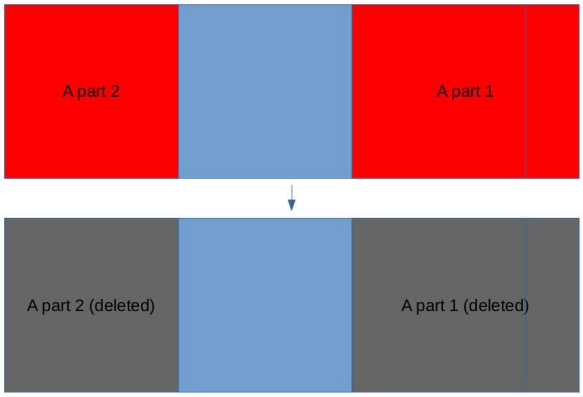
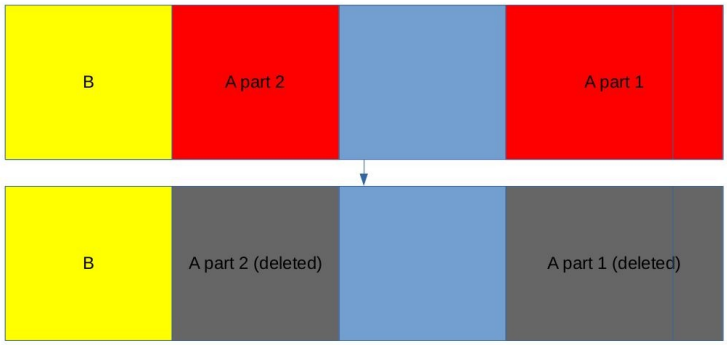
Andrew Meyer
DFR Project week 5

What I have done:

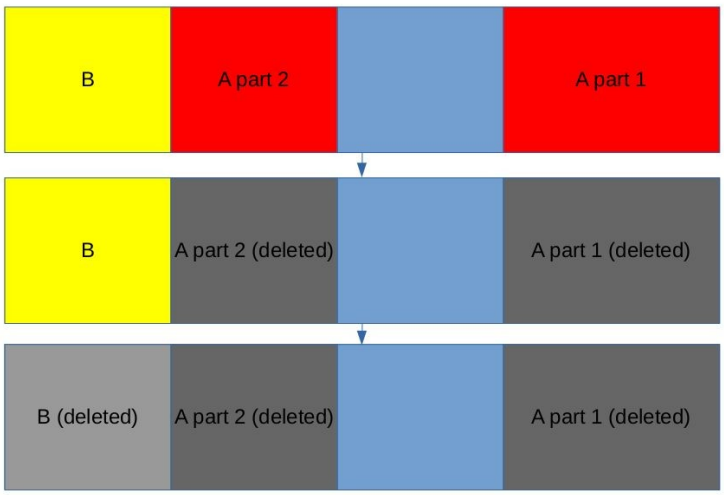
- Completed and documented 11 test images (1-3, 4i-4iv, 5i-5iv)
- Thought of 3 more test cases:
  - file is fragmented starting at the end of the file system and continuing at the beginning



  - file is fragmented starting at the end of the file system and continuing in the middle

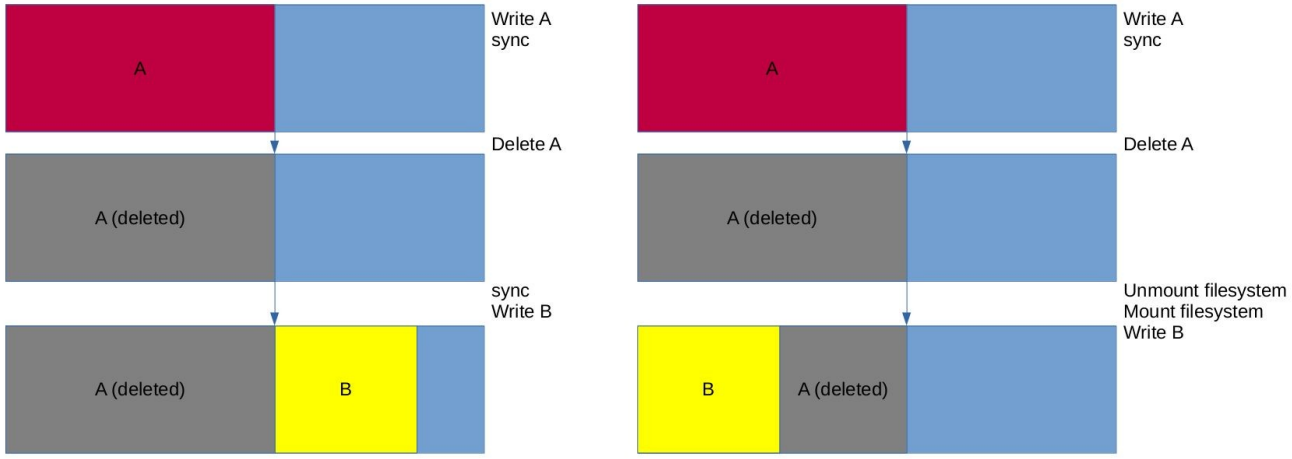


  - file is fragmented starting at the end of the file system and continuing in the middle with a deleted file at the beginning



What I have learned:

- Ubuntu appears to use the “next fit” aka “next available” file allocation algorithm for FAT. Knowing the allocation algorithm makes it easier to create test cases.
- Two different ways of forcing buffered writes to the disk:
  - Unmounting the file system: resets the “last written” location on the disk
  - Using the “sync” command: does not reset the “last written” location



This is very useful when creating cases that involve overwriting part of a deleted file.

- In Ubuntu, new directory entries may overwrite deleted ones, using what appears to be a “first fit” algorithm. This initially made cases 4-5 useless because A’s directory entry would be overwritten, making it completely unrecoverable within our scope.
  I was able to resolve this by strategically choosing whether to sync or remount. I gave an example (case 4i) on the next page.
- When a file doesn’t perfectly fill a sector, Ubuntu fills the remainder of that sector with 0.

Major Concerns:

- In cases of fragmentation, **do we expect the DFR tool to recover the entire file or only the first fragment** (the one addressed in the directory entry)?
  I think it depends on how we interpret the phrase “all non-allocated data blocks identified in a residual metadata entry” in DFR-CR-03.
  When a file is deleted, its data in the File Allocation Table is deleted. The directory entry only specifies where the file begins. So, in most cases the rest of a fragmented file can only be found by making an educated guess. Of course, guessing can sometimes result in recovering data that was not part of the file, violating DFR-CR-04.
  - During our meeting, we concluded that because there is no way to locate the fragments from essential metadata without guessing, only the first fragment is “identified in a residual metadata entry,” so as long as the tool at least recovers the first fragment, it fulfills DFR-CR-03.
  - “Each Recovered Object shall *include* all non-allocated data blocks identified in a residual metadata entry.” DFR-CR-03 does not limit the tool to *only* blocks indicated by metadata, so as long as all recovered blocks are part of the deleted block pool per DFR-CR-04, a tool could recover more or all of the fragmented file and still conform to the standards. Do we agree on this?

Work for the next week:

- Create and document my last 2 test images plus the 3 new ones (should not take as long as the previous ones now that I better understand the allocation algorithms)
- Create expected recovered files
- Run Autopsy on test images
- Start planning NTFS test cases

Directory entry problem example: Case 4i
When file B is written over the first cluster of A, it appears to overwrite A's directory entry as well.

Solved by:

1. Write files D (1M) and C (1M)
2. Sync
3. Delete D
4. Sync
5. Write E (1M)
6. Delete C
7. Remount
8. Write A (1M)

At this point A has the second directory entry while still occupying the first clusters of the data area. This way, when A is deleted and B is written, A's directory entry is preserved and E's directory entry is overwritten instead.

9. Delete E
10. Remount
11. Grow A to 3M
12. Sync
13. Delete A
14. Remount
15. Write B (1M)

Now only B and what remains of A exist in the data area, and only B and A's directory entries exist in the root directory.

Note: the process listed in case_4i.txt uses file dd1M twice. Here the second instance is named E for clarity. In practice both instances of D have their directory entries and data completely overwritten, their only purpose is to help position the other files.