CSU, College of Engineering and Computer Science
Department of Computer Science
CSC 153 – Computer Forensics Principles and Practice

Catherine Nguyen
CSC 153
11/21/2020

Activity 5: Use Winhex to Examine NTFS Disks

Objectives:

- Become familiar with the WinHex forensics tool.
- Use WinHex to become familiar with different file types.
- Use WinHex to explore and become familiar with the MFT, including headers and attributes.

Part 1: Explore different file types.

The first part can be done on our local device, we use Microsoft word to create a document called, 'Mywordnew.doc', and this Word 97-2003 document will encompass the text, "This is a test. " Then we need to open WinHex and find the .doc we created and copy the file hexadecimal header D0 CF 11 E0 A1 B1 1A E1 to the next new document.

Header .doc

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	ÐÏ à¡± á

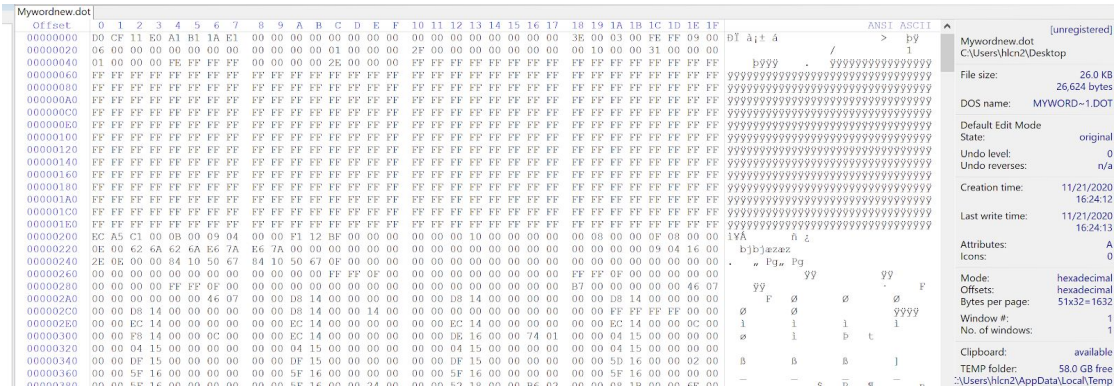

Figure 1: Header for Word 97-2003 .doc file.

These steps are then repeated for the following files types:

- .xls
- .docx
- .xlsx
- .jpg
- .png

Screenshots and text for each file type can be found below.

Header .xls

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	ÐÏ à¡± á

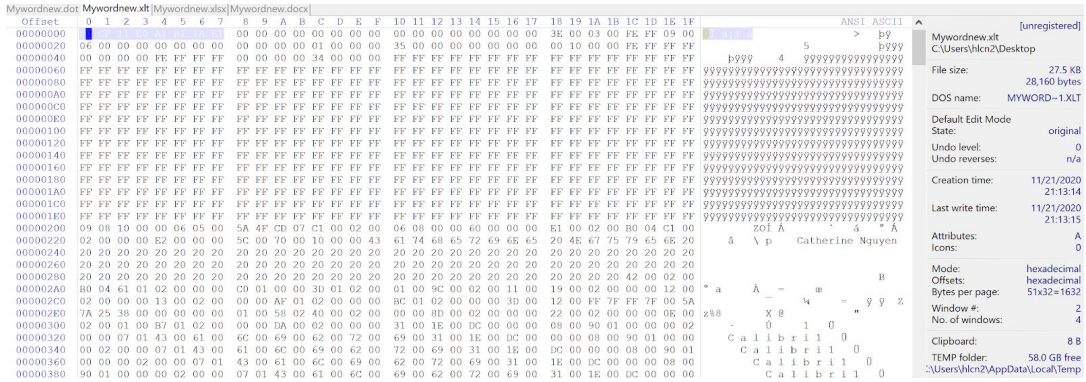

Figure 2. Header for Excel 97-2003 Workbook .xls.

Header .docx

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	DF	A4	PK ! ß¤

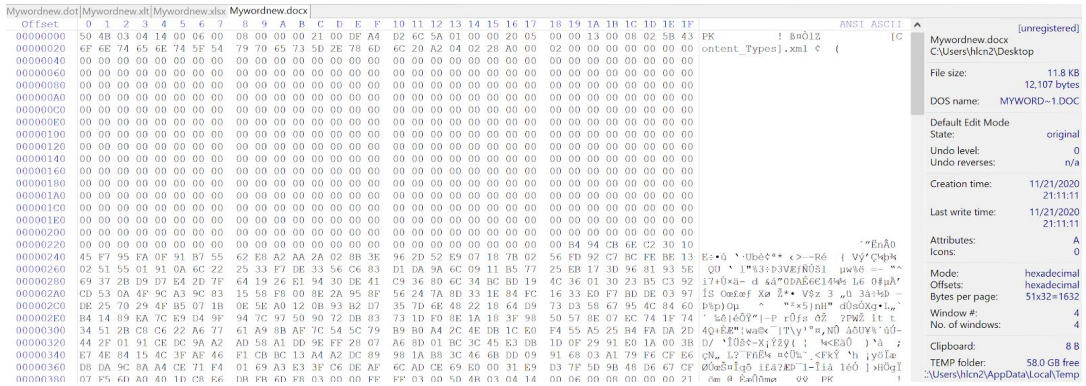

Figure 3. Header for Word 2007 Document .docx.

Header .xlsx

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	41	37	PK ! A7

Figure 4. Header for Excel 2007 Workbook .xlsx.

Header for .jpg

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	02	00	00	01	ÿØÿà JFIF

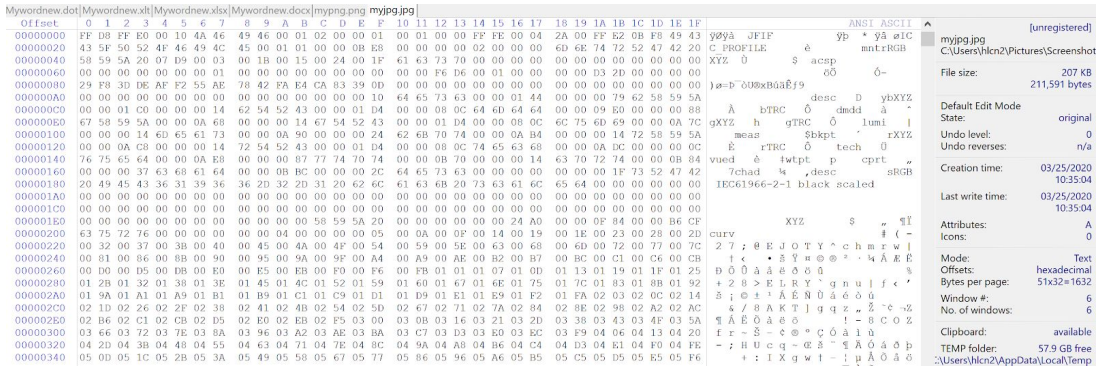

Figure 5. Header for .jpg file

Header for .png

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG IHDR

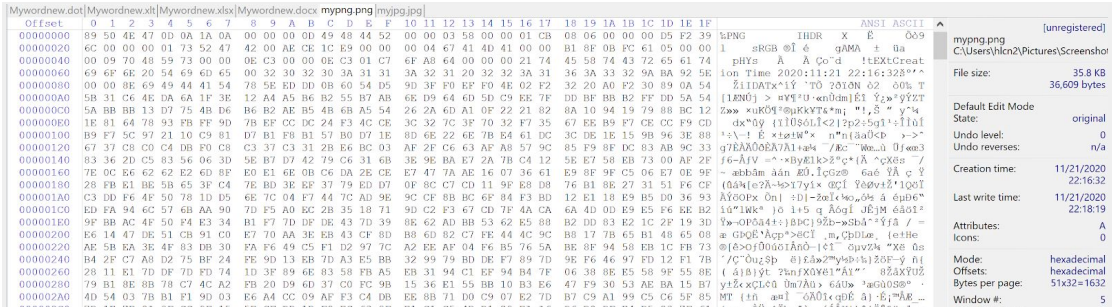

Figure 6. Header for .png file.

```
Word 97-2003 .doc
00000000   D0 CF 11 E0 A1 B1 1A E1  00 00 00 00 00 00 00 00 00                ÐÏ à¡± á
Excel 97-2003 Workbook .xls
00000000   D0 CF 11 E0 A1 B1 1A E1  00 00 00 00 00 00 00 00 00                ÐÏ à¡± á
.docx
00000000   50 4B 03 04 14 00 06 00  08 00 00 00 21 00 DF A4                  PK          ! ß¤
.xlsx
00000000   50 4B 03 04 14 00 06 00  08 00 00 00 21 00 41 37                  PK          ! A7
.jpg
00000000   FF D8 FF E0 00 10 4A 46  49 46 00 01 02 00 00 01                  ÿØÿà  JFIF
.png
00000000   89 50 4E 47 0D 0A 1A 0A  00 00 00 0D 49 48 44 52                  ‰PNG        IHDR
```

Figure 7. Text file with all the header info

Observations: As you can see, hexadecimal headers for .doc and .xls files match one another. Additionally, .docx and .xlsx also match one another. It seems the Microsoft Office Suite versions produce files with the same hex header, version 97-2003 match one another, and version 2007+ match one another. It doesn't appear that spreadsheet files differ from word documents, which is surprising. It's all about the version.

Part 2: Explore MFT.

The second part of this activity requires we create a file named lab1part2.txt, containing the following lines.

- A countryman between two layers is like a fish between two cats.
- A slip of the foot you may soon recover, but a slip of the tongue you may never get over.
- An investment in knowledge always pays the best interest.
- Drive thy business or it will drive there.

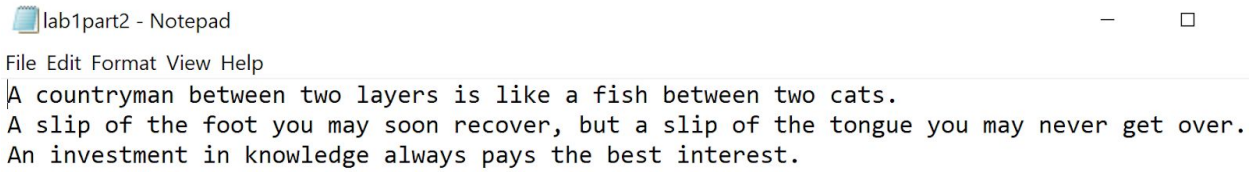

Figure 8. Creating lab1part2.txt file.

We then need to open WinHexx as administrator and navigate the options to change it to read-only mode.

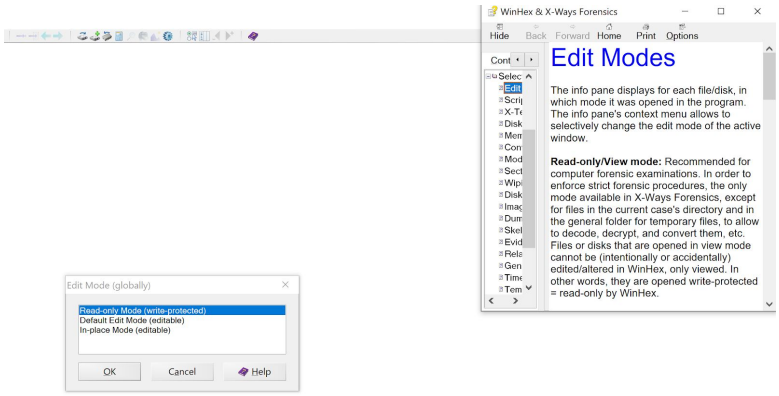

Figure 9. Set Winhex to Read-Only mode.

After that, we need to select the drive that has our lab1part2.txt file for us to examine. To do this we need to go into tools then open disk. Selecting C: drive.

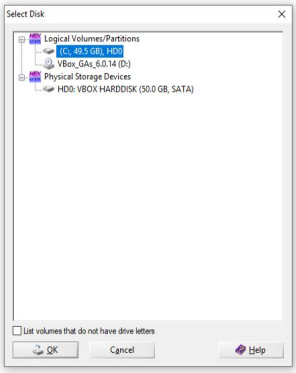

Figure 10. Choosing the C: drive as our disk to open.

Now before we can begin analyzing our created .txt file we first need to change the data interpreter settings. So we need to open that up and **check the Win32 FILETIME (64 bit)** before we can progress.

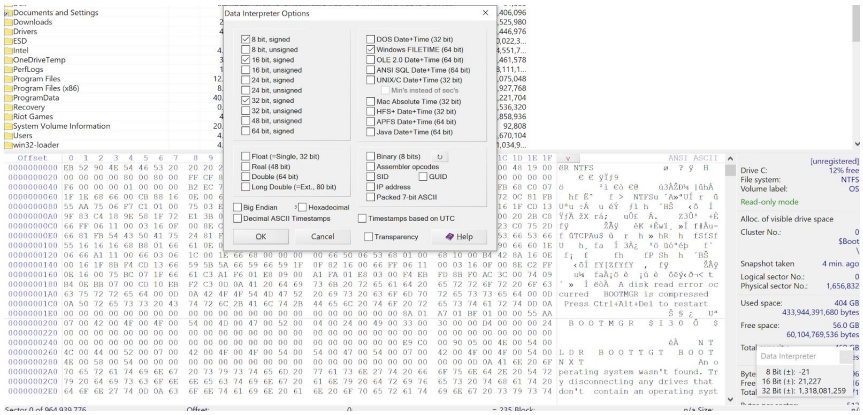

Figure 11. Setting Data interpreter to Win32 FILETIME (64 bit).

Once that is selected and changed we can now examine our lab1part2.txt file and find the beginning.

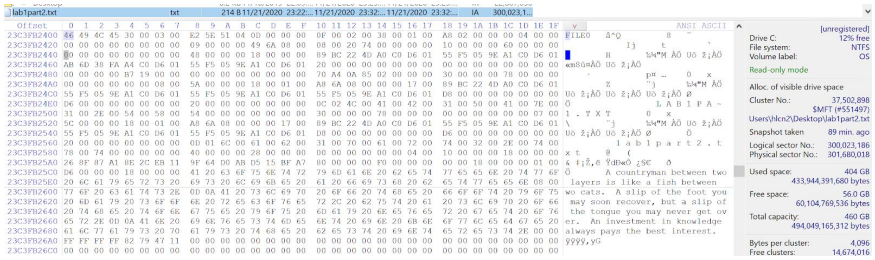

Figure 12. Beginning of the record for lab1part2.txt

Now in order to perform the 0x10 attribute, we click the beginning of the MFT record and drag until the offset counter is 0x38.

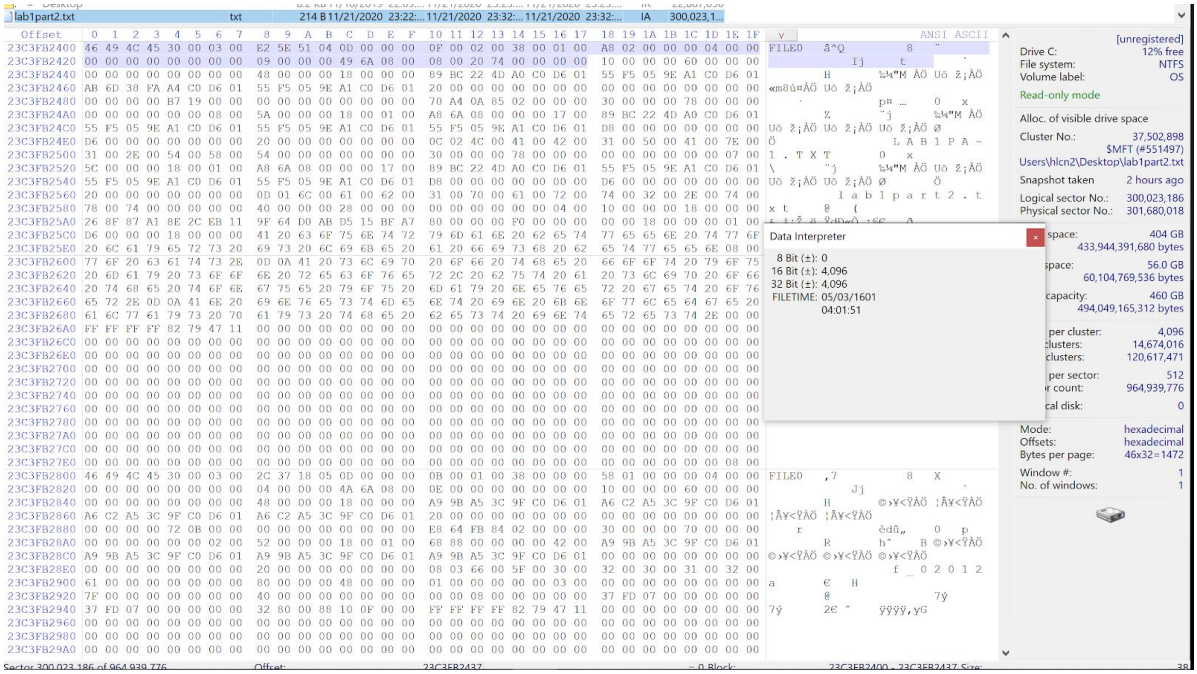

Figure 13. The start of attribute 0x10 at offset 0x38

The files created date and time can be found at offset 0x10 to 0x1f from the beginning of attribute 0x10.

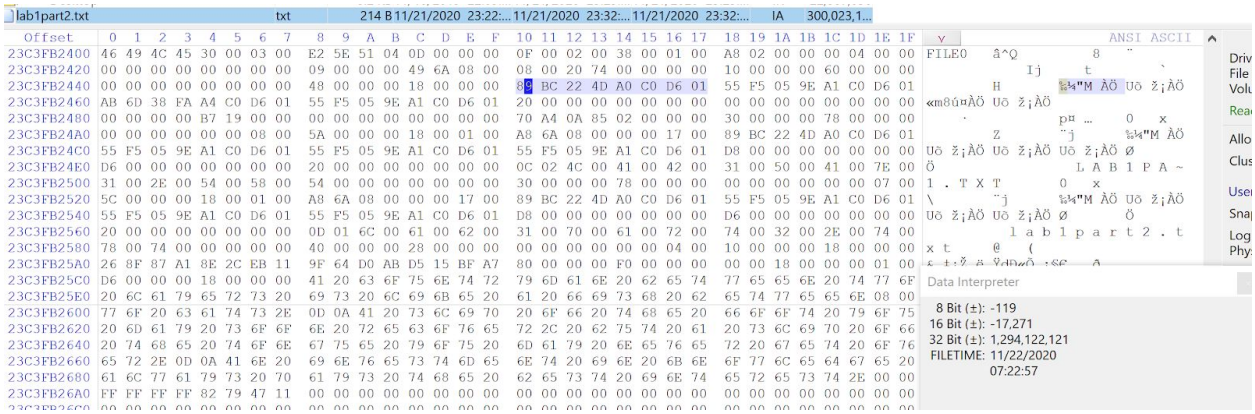

Figure 14. File created date and time.

Questions:

1. **According to the data interpreter, what are the file creation date and time for the file lab1part.txt?**
    a. According to the data interpreter, the create date and time is **11/22/2020 07:22:57.**

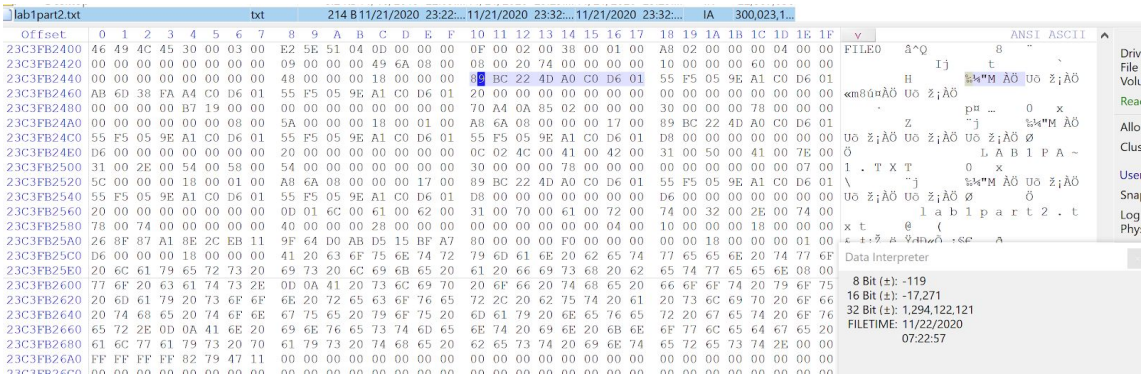

Figure 15. Data Interpreter's file created date and time.

2. **Using File Explorer and go to the folder where the lab1part2.txt located, right-click on the arrow near "Size" or "Name" and select the "Date created". Now the "Date created" time is also displayed.**
    a. The file explore says that the txt file's created date and time is **11/01/2020 at 11:22 PM.**

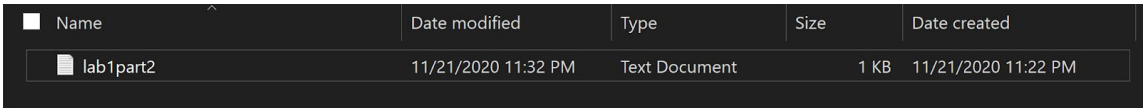

Figure 16. File Explorer's date and time creation.

3. **Compare this time and the time you got from the data interpreter. Are they the same? If not, why?**
    a. The time for the data interpreter is ahead of the time the file explorer said it to be. We can find out that the reason the times differ is due to the data interpreter's timestamp. We just need to change the time to UTC. To do so go to **options then press Data Interpreter** and check the **Timestamps based on the UTC** box. After doing this, the time and data

become identical. The reason behind the time difference is that the file explorer's default time creation timestamp follows UTC time and the Data interpreter does not.

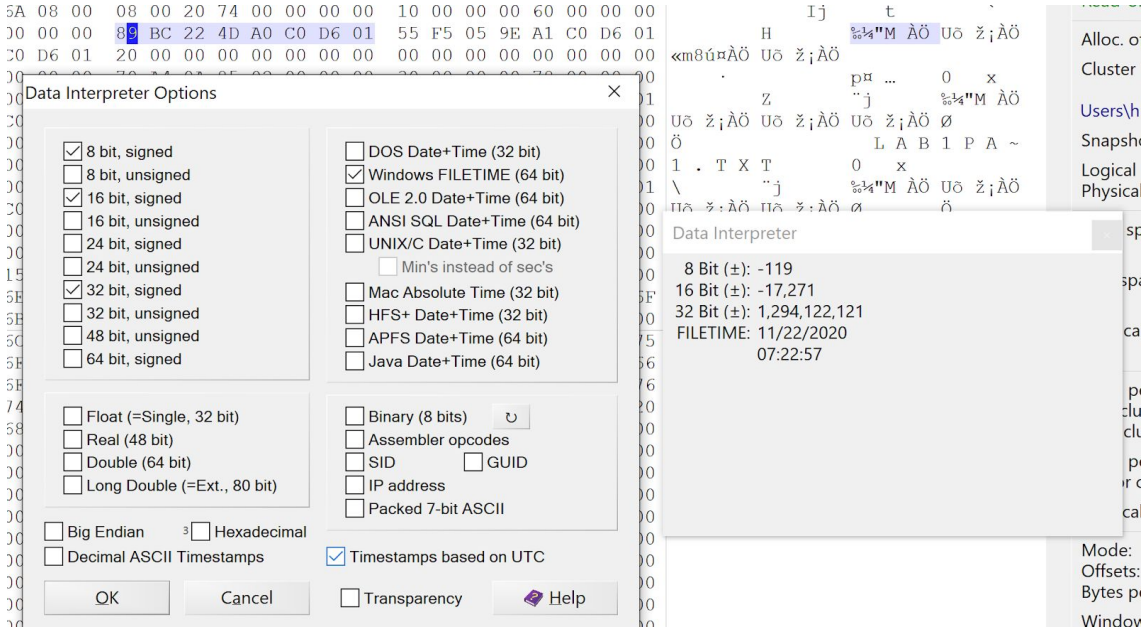

Figure 17. Checking 'Timestamps based on UTC'.

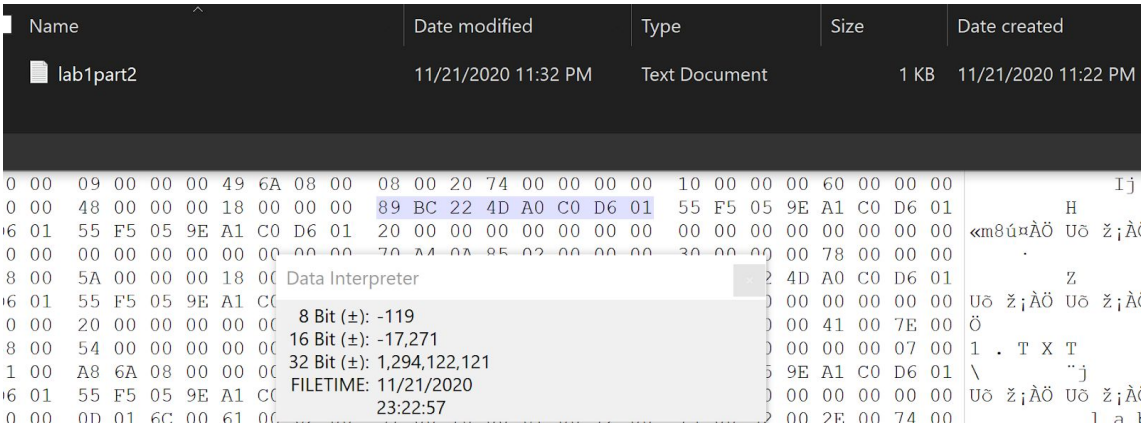

Figure 18. Data Interpreter time now matches File explorer time.

4. **What is the size of the MFT record?**
   a. The size of the MTF record is, in big-endian, offset 23C3FB2400 **00 04 00 00**. We can find that at offset **0x1C to 0x1F.**

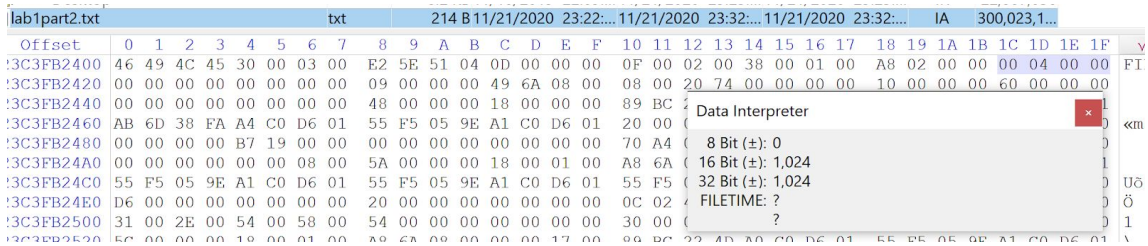

Figure 19. The size of the MTF Record.

5. **What is the length of the header for the MFT record?**
    a. The header length for the MTF record is **0x38**. Found at offset **0x00 to 0x14.**

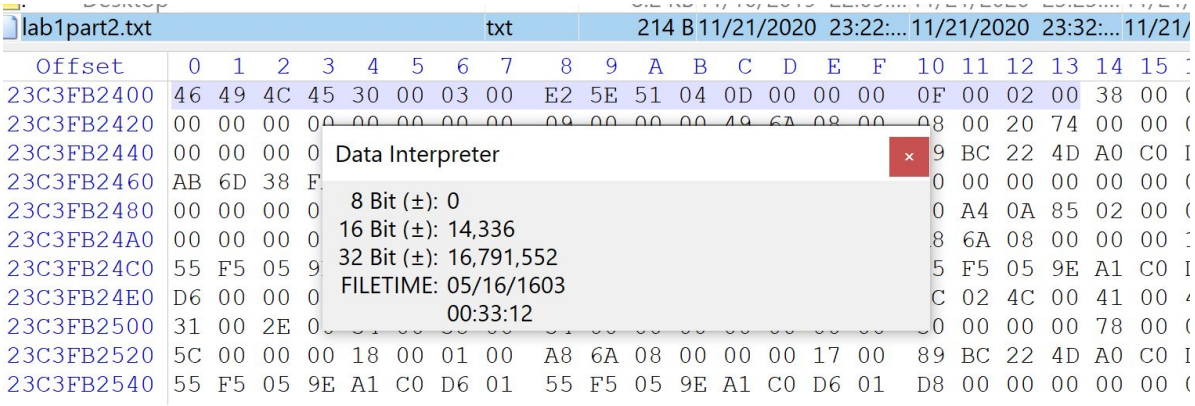

Figure 20. The length of the MTF record's header is 0x38.

6. **What are the file's last modified date and time?**
    a. The last modified time and date are 1**1/21/2020 11:32 PM or 11/21/2020 23:32:22** according to the data interpreter. We can find the modified date and time at offset **0x20 to 0x27.**

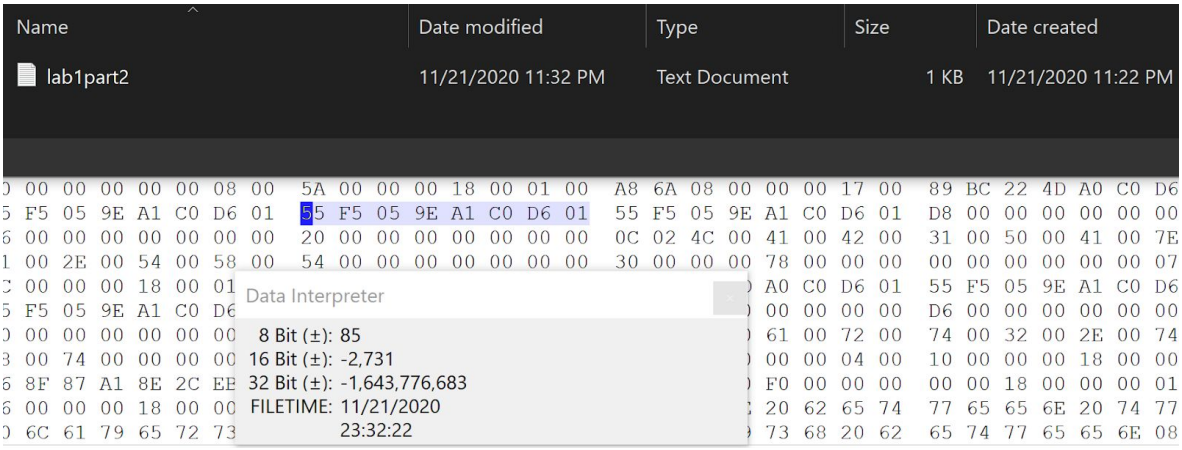

Figure 21. File last modified data based on UTC time.

7. **What is the file name? In which attribute and at what position can you find it?**
    a. From lecture 5's handout, if the file name is longer than eight characters there are two attributes 0x30. Since our file name is longer than 8 characters we fall into this case. That means we have a short file name and a long file name.
    b. Short file names are found at offset 0x5A from the first 0x30 attribute. Our short file name is LAB1PA~1.TXT

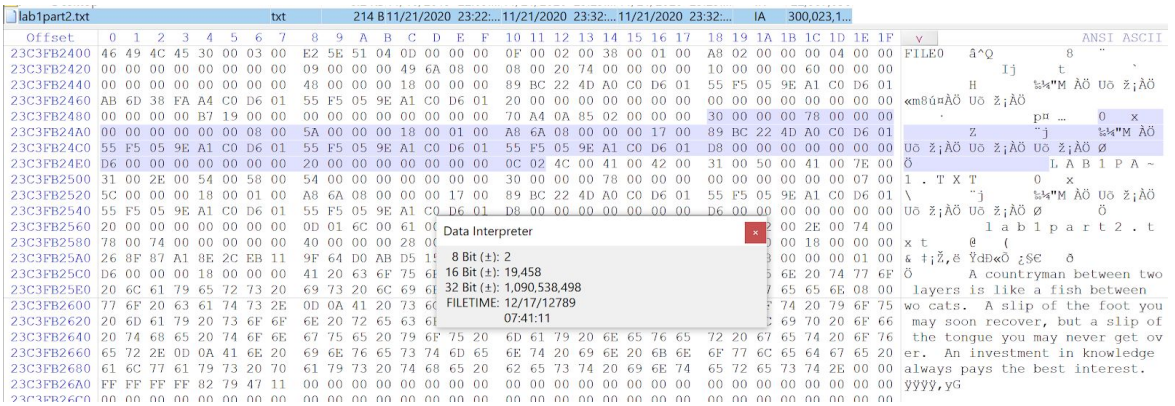

Figure 22. Short name 0x5A from the first 0x30 attribute.

c. Long file names are found at offset 0x5A from the second 0x30 attribute. Our short file name is lab1part2.txt.

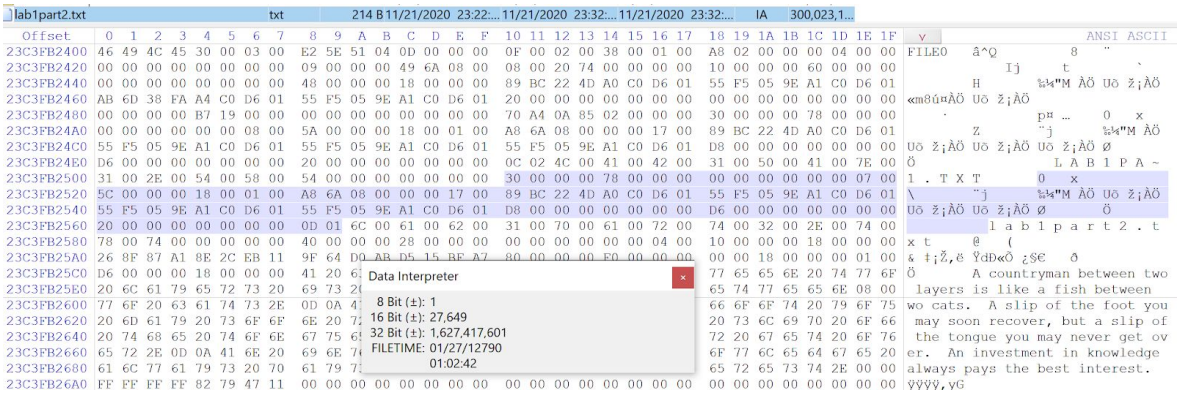

Figure 23. Long name 0x5A from the second 0x30 attribute.

8. Is this file a resident file or a nonresident file? Where can you find the evidence?

a. The resident/nonresident flags exist at offset 0x08 from attribute 0x80. In this case, it is a resident file.

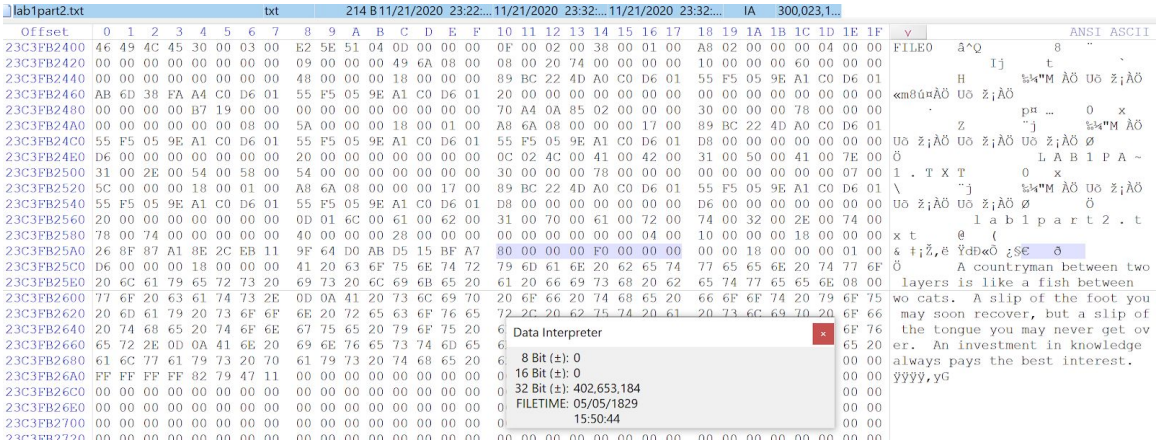

Figure 24. Resident flag set to 0x00, resident file.

9. In which attribute can you find the data run? Where is the start of the data run?

a. The start of the data run for a resident exists at offset 0x18 from attribute 0x80.

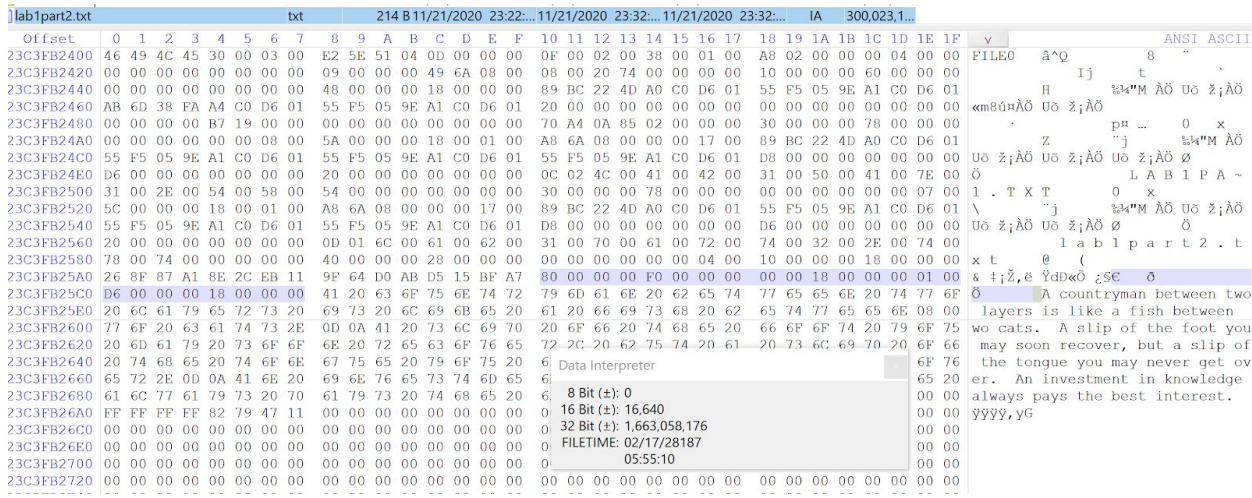

Figure 25. Start of data run for the resident file. Offset 0x18 from 0x80.