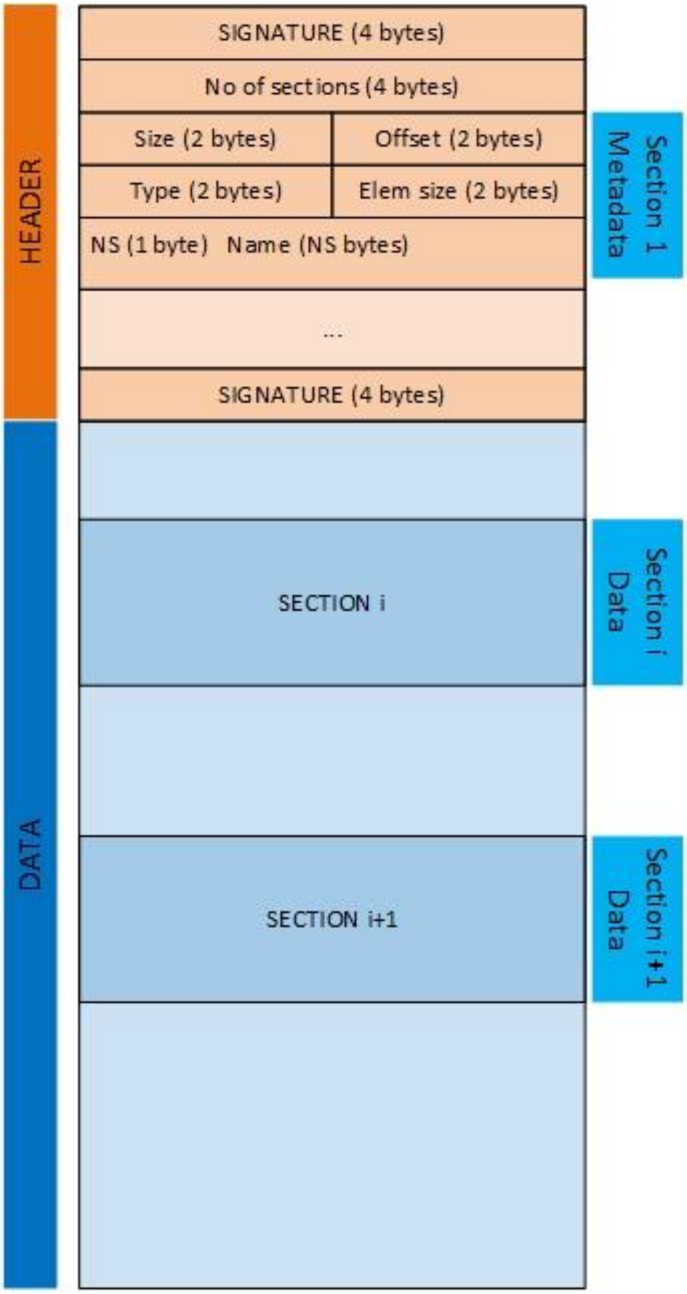
Lab ASSIGNMENT 1

You are given the following file format, which we will call from now on **SF** (i.e. "**section file**") **format**. A section file consists in two areas: a **header** and a **data area**. The header area describes the way the data area must be read. The header area also contains the file's signature, both at its beginning and end, like illustrated in the figure below. The signature of the section files is "0xB612B612".



The data area consists in more **sections**, not necessarily placed one after another, i.e. there could be holes between sections. A section is a contiguous area, consisting in bytes placed at consecutive positions in the file. A **hole** is also a contiguous area between two consecutive sections, but its contents is undefined and normally unused. Each section has a type. The codes associated to each section type and their interpretations are:

-
    - 0 (RESERVED),
    - 1 (TEXT),
    - 2 (BINARY).

Each section is considered to be a sequence of elements (records) of the same size. An element is a group of one (1) or more consecutive bytes. Elements in different sections could be of different sizes. The size of each section's elements is specified explicitly in the file's header. The name of a section (which is a string of ASCII characters) is of variable size, its actual size being specified by the "**NS**" **field**.

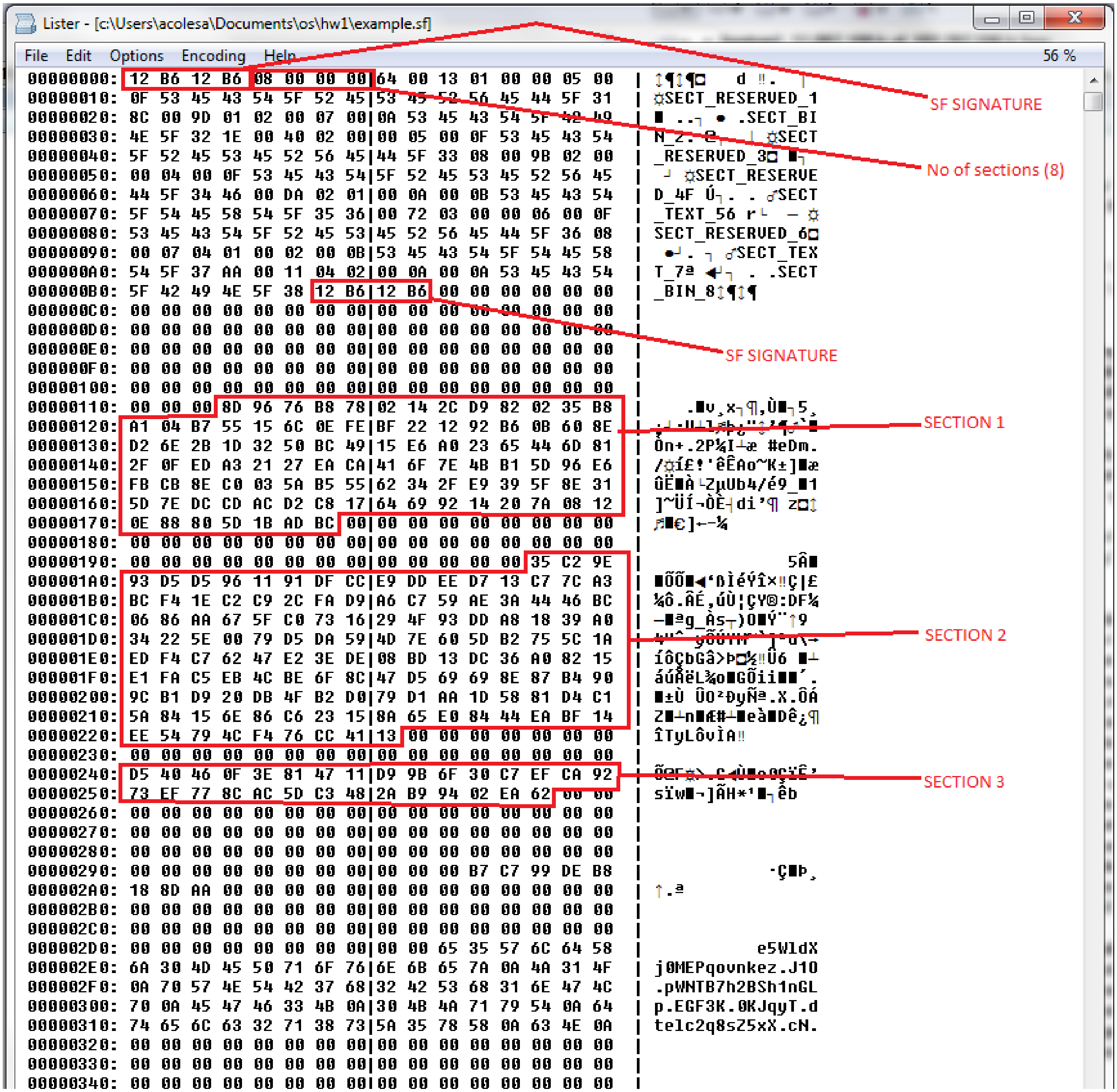
PART 1. Learning the SF format *(60 min)*

1. Download the following "example.sf" file, which complies perfectly the SF format.
2. We will use the following tools in order to analyze the example file. Both tools described below allow you to view the contents of a file as text (i.e. printable characters) or binary (i.e. hexadecimal codes).

Viewing a file's contents could be done by pressing F3. By default, a file contents is displayed as text, i.e. the bytes' values are interpreted as ASCII characters.

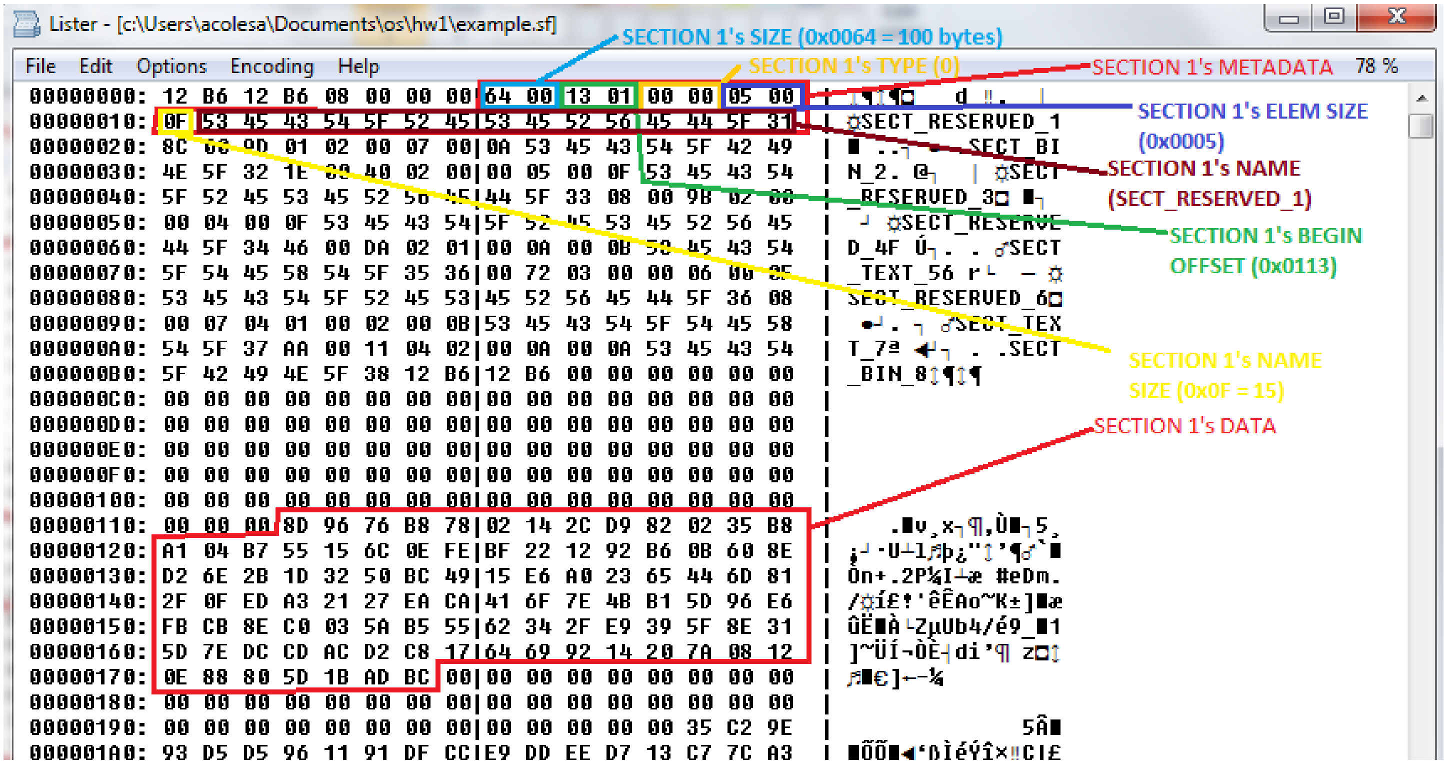
1. **on Linux**: *mc* utility (if not installed, install it by typing "`sudo apt-get install mc`" command in a terminal, or using "Ubunutu Software Center" application); *mc* should be run in a terminal; to switch to the hexadecimal view and back, F4 could be pressed;
2. **on Windows**: *Total Commander* (if not installed, download it from http://www.ghisler.com/ and install it); to switch to the hexadecimal view and back, "3" could be pressed (see the "*Options*" menu entry).

3. Open the "*example.sf*" file in a viewer and switch to the hexadecimal view. It should look like the picture below, where you can note the file's header and data area. We illustrated the file's signatures (at both the beginning and end of the header), number of sections (8) and few section's data components.



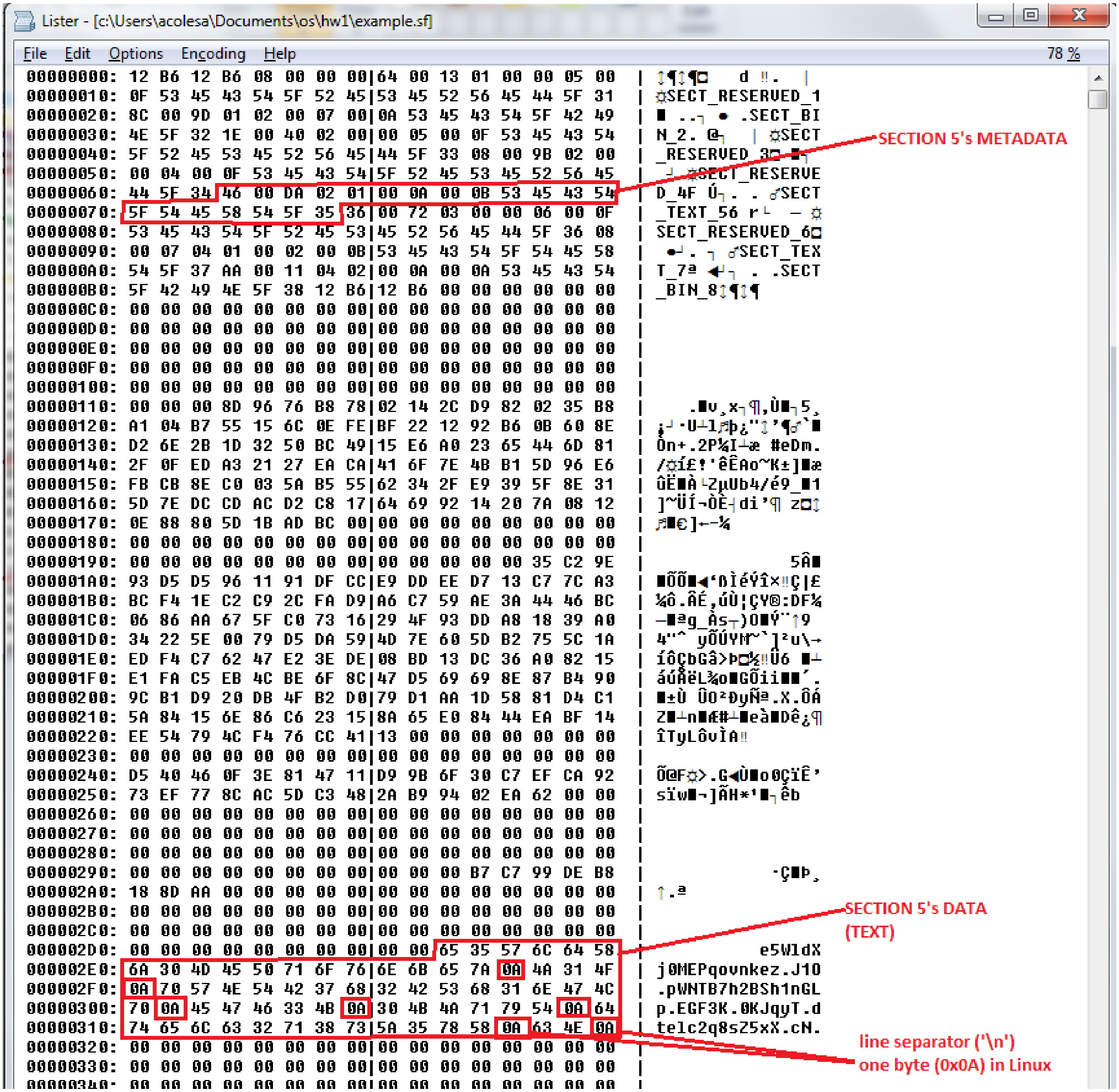
```
00000000: 12 B6 12 B6 08 00 00 00|64 00 13 01 00 00 05 00  | ↕¶↕¶◘    d ‼.  ♣
00000010: 0F 53 45 43 54 5F 52 45|53 45 52 56 45 44 5F 31  | ☼SECT_RESERVED_1
00000020: 8C 00 9D 01 02 00 07 00|0A 53 45 43 54 5F 42 49  | Œ ..┐ • .SECT_BI
00000030: 4E 5F 32 1E 00 40 02 00|00 05 00 0F 53 45 43 54  | N_2▲ @☻  ♣ ☼SECT
00000040: 5F 52 45 53 45 52 56 45|44 5F 33 08 00 9B 02 00  | _RESERVED_3◘ ›☻
00000050: 00 04 00 0F 53 45 43 54|5F 52 45 53 45 52 56 45  |  ┘ ☼SECT_RESERVE
00000060: 44 5F 34 46 00 DA 02 01|00 0A 00 0B 53 45 43 54  | D_4F Ú┐. . ♂SECT
00000070: 5F 54 45 58 54 5F 35 36|00 72 03 00 00 06 00 0F  | _TEXT_56 r└  ─ ☼
00000080: 53 45 43 54 5F 52 45 53|45 52 56 45 44 5F 36 08  | SECT_RESERVED_6◘
00000090: 00 07 04 01 00 02 00 0B|53 45 43 54 5F 54 45 58  |  •┘. ┐ ♂SECT_TEX
000000A0: 54 5F 37 AA 00 11 04 02|00 0A 00 0A 53 45 43 54  | T_7ª ◄┘┐ . .SECT
000000B0: 5F 42 49 4E 5F 38 12 B6|12 B6 00 00 00 00 00 00  | _BIN_8↕¶↕¶
000000C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000D0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000E0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000F0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000100: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000110: 00 00 00 8D 96 76 B8 78|02 14 2C D9 82 02 35 B8  |    .■v¸x┐¶,Ù■┐5¸
00000120: A1 04 B7 55 15 6C 0E FE|BF 22 12 92 B6 0B 60 8E  |
00000130: D2 6E 2B 1D 32 50 BC 49|15 E6 A0 23 65 44 6D 81  | Òn+.2P¼I┴æ #eDm.
00000140: 2F 0F ED A3 21 27 EA CA|41 6F 7E 4B B1 5D 96 E6  | /☼í£!'êÊAo~K±]■æ
00000150: FB CB 8E C0 03 5A B5 55|62 34 2F E9 39 5F 8E 31  | ûË■À└ZµUb4/é9_■1
00000160: 5D 7E DC CD AC D2 C8 17|64 69 92 14 20 7A 08 12  | ]~ÜÍ¬ÒÈ┤di'¶ z◘↕
00000170: 0E 88 80 5D 1B AD BC 00|00 00 00 00 00 00 00 00  | ♫■€]←-¼
00000180: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000190: 00 00 00 00 00 00 00 00|00 00 00 00 00 35 C2 9E  |              5Â■
000001A0: 93 D5 D5 96 11 91 DF CC|E9 DD EE D7 13 C7 7C A3  | ■ÕÕ■◄'ßÌéÝî×‼Ç|£
000001B0: BC F4 1E C2 C9 2C FA D9|A6 C7 59 AE 3A 44 46 BC  | ¼ô.ÂÉ,úÙ¦ÇY®:DF¼
000001C0: 06 86 AA 67 5F C0 73 16|29 4F 93 DD A8 18 39 A0  | ─■ªg_Às┬)O■Ý¨↑9
000001D0: 34 22 5E 00 79 D5 DA 59|4D 7E 60 5D B2 75 5C 1A  |
000001E0: ED F4 C7 62 47 E2 3E DE|08 BD 13 DC 36 A0 82 15  | íôÇbGâ>Þ◘½‼Ü6 ■┴
000001F0: E1 FA C5 EB 4C BE 6F 8C|47 D5 69 69 8E 87 B4 90  | áúÅëL¾o■GÕii■■´.
00000200: 9C B1 D9 20 DB 4F B2 D0|79 D1 AA 1D 58 81 D4 C1  | ■±Ù ÛO²ÐyѪ.X.ÔÁ
00000210: 5A 84 15 6E 86 C6 23 15|8A 65 E0 84 44 EA BF 14  | Z■┴n■Æ#┴■eà■Dê¿¶
00000220: EE 54 79 4C F4 76 CC 41|13 00 00 00 00 00 00 00  | îTyLôvÌA‼
00000230: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000240: D5 40 46 0F 3E 81 47 11|D9 9B 6F 30 C7 EF CA 92  |
00000250: 73 EF 77 8C AC 5D C3 48|2A B9 94 02 EA 62 00 00  | sïw■¬]ÃH*¹■┐êb
00000260: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000270: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000280: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000290: 00 00 00 00 00 00 00 00|00 00 00 B7 C7 99 DE B8  |            ·Ç■Þ¸
000002A0: 18 8D AA 00 00 00 00 00|00 00 00 00 00 00 00 00  | ↑.ª
000002B0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000002C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000002D0: 00 00 00 00 00 00 00 00|00 00 65 35 57 6C 64 58  |           e5WldX
000002E0: 6A 30 4D 45 50 71 6F 76|6E 6B 65 7A 0A 4A 31 4F  | j0MEPqovnkez.J1O
000002F0: 0A 70 57 4E 54 42 37 68|32 42 53 68 31 6E 47 4C  | .pWNTB7h2BSh1nGL
00000300: 70 0A 45 47 46 33 4B 0A|30 4B 4A 71 79 54 0A 64  | p.EGF3K.0KJqyT.d
00000310: 74 65 6C 63 32 71 38 73|5A 35 78 58 0A 63 4E 0A  | telc2q8sZ5xX.cN.
00000320: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000330: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000340: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
```

4. In the figure below we marked the header details of the first section. Please identify them on you own example file.


Lister - [c:\Users\acolesa\Documents\os\hw1\example.sf]
File
Edit
Options
Encoding
Help
78 %
SECTION 1's SIZE (0x0064 = 100 bytes)
SECTION 1's TYPE (0)
SECTION 1's METADATA
SECTION 1's ELEM SIZE (0x0005)
SECTION 1's NAME (SECT_RESERVED_1)
SECTION 1's BEGIN OFFSET (0x0113)
SECTION 1's NAME SIZE (0x0F = 15)
SECTION 1's DATA

```
00000000: 12 B6 12 B6 08 00 00 00|64 00 13 01 00 00 05 00  | ↕¶↕¶◘   d ‼.  |
00000010: 0F 53 45 43 54 5F 52 45|53 45 52 56 45 44 5F 31  | ☼SECT_RESERVED_1
00000020: 8C 00 0D 01 02 00 07 00|0A 53 45 43 54 5F 42 49  | █ ..┐ •  SECT_BI
00000030: 4E 5F 32 1E 00 40 02 00|00 05 00 0F 53 45 43 54  | N_2. @┐   | ☼SECT
00000040: 5F 52 45 53 45 52 56 45|44 5F 33 08 00 9B 02 00  | _RESERVED_3◘ █┐
00000050: 00 04 00 0F 53 45 43 54|5F 52 45 53 45 52 56 45  |  ┘ ☼SECT_RESERVE
00000060: 44 5F 34 46 00 DA 02 01|00 0A 00 0B 53 45 43 54  | D_4F Ú┐. . ♂SECT
00000070: 5F 54 45 58 54 5F 35 36|00 72 03 00 00 06 00 0F  | _TEXT_56 r└  ─ ☼
00000080: 53 45 43 54 5F 52 45 53|45 52 56 45 44 5F 36 08  | SECT_RESERVED_6◘
00000090: 00 07 04 01 00 02 00 0B|53 45 43 54 5F 54 45 58  |  •┘. ┐ ♂SECT_TEX
000000A0: 54 5F 37 AA 00 11 04 02|00 0A 00 0A 53 45 43 54  | T_7ª ◄┘┐ . .SECT
000000B0: 5F 42 49 4E 5F 38 12 B6|12 B6 00 00 00 00 00 00  | _BIN_8↕¶↕¶
000000C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000D0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000E0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000F0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000100: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000110: 00 00 00 8D 96 76 B8 78|02 14 2C D9 82 02 35 B8  |    .█v¸x┐¶,Ù█┐5¸
00000120: A1 04 B7 55 15 6C 0E FE|BF 22 12 92 B6 0B 60 8E  | ¡┘·U┴l♫þ¿"↕'¶♂`█
00000130: D2 6E 2B 1D 32 50 BC 49|15 E6 A0 23 65 44 6D 81  | Òn+.2P¼I┴æ #eDm.
00000140: 2F 0F ED A3 21 27 EA CA|41 6F 7E 4B B1 5D 96 E6  | /☼í£!'êÊAo~K±]█æ
00000150: FB CB 8E C0 03 5A B5 55|62 34 2F E9 39 5F 8E 31  | ÛË█À└ZµUb4/é9_█1
00000160: 5D 7E DC CD AC D2 C8 17|64 69 92 14 20 7A 08 12  | ]~ÜÍ¬ÒÈ┤di'¶ z◘↕
00000170: 0E 88 80 5D 1B AD BC 00|00 00 00 00 00 00 00 00  | ♫█€]←-¼
00000180: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000190: 00 00 00 00 00 00 00 00|00 00 00 00 00 35 C2 9E  |              5Â█
000001A0: 93 D5 D5 96 11 91 DF CC|E9 DD EE D7 13 C7 7C A3  | █ÕÕ█◄'ßÌéÝî×‼Ç|£
```

5. As an exercise, identify the header fields for another section, i.e. the fifth one, which has a TEXT type in our example. See the picture below



```
00000000: 12 B6 12 B6 08 00 00 00|64 00 13 01 00 00 05 00  | ↕¶↕¶◘   d ‼.  |
00000010: 0F 53 45 43 54 5F 52 45|53 45 52 56 45 44 5F 31  | ☼SECT_RESERVED_1
00000020: 8C 00 9D 01 02 00 07 00|0A 53 45 43 54 5F 42 49  | █ ..┐ • .SECT_BI
00000030: 4E 5F 32 1E 00 40 02 00|00 05 00 0F 53 45 43 54  | N_2. @┐  | ☼SECT
00000040: 5F 52 45 53 45 52 56 45|44 5F 33 08 00 9B 02 00  | _RESERVED_3◘ █┐
00000050: 00 04 00 0F 53 45 43 54|5F 52 45 53 45 52 56 45  |  ┘ ☼SECT_RESERVE
00000060: 44 5F 34 46 00 DA 02 01|00 0A 00 0B 53 45 43 54  | D_4F Ú┐. . ♂SECT
00000070: 5F 54 45 58 54 5F 35 36|00 72 03 00 00 06 00 0F  | _TEXT_56 r└  ─ ☼
00000080: 53 45 43 54 5F 52 45 53|45 52 56 45 44 5F 36 08  | SECT_RESERVED_6◘
00000090: 00 07 04 01 00 02 00 0B|53 45 43 54 5F 54 45 58  |  •┘. ┐ ♂SECT_TEX
000000A0: 54 5F 37 AA 00 11 04 02|00 0A 00 0A 53 45 43 54  | T_7ª ◄┘┐ . .SECT
000000B0: 5F 42 49 4E 5F 38 12 B6|12 B6 00 00 00 00 00 00  | _BIN_8↕¶↕¶
000000C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000D0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000E0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000000F0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000100: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000110: 00 00 00 8D 96 76 B8 78|02 14 2C D9 82 02 35 B8  |    .█v¸x┐¶,Ù█┐5¸
00000120: A1 04 B7 55 15 6C 0E FE|BF 22 12 92 B6 0B 60 8E  | ¡┘·U┴l♫þ¿"↕'¶♂`█
00000130: D2 6E 2B 1D 32 50 BC 49|15 E6 A0 23 65 44 6D 81  | Òn+.2P¼I┴æ #eDm.
00000140: 2F 0F ED A3 21 27 EA CA|41 6F 7E 4B B1 5D 96 E6  | /☼í£!'êÊAo~K±]█æ
00000150: FB CB 8E C0 03 5A B5 55|62 34 2F E9 39 5F 8E 31  | ûË█À└ZµUb4/é9_█1
00000160: 5D 7E DC CD AC D2 C8 17|64 69 92 14 20 7A 08 12  | ]~ÜÍ¬ÒÈ┤di'¶ z◘↕
00000170: 0E 88 80 5D 1B AD BC 00|00 00 00 00 00 00 00 00  | ♫█€]←-¼
00000180: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000190: 00 00 00 00 00 00 00 00|00 00 00 00 00 35 C2 9E  |              5Â█
000001A0: 93 D5 D5 96 11 91 DF CC|E9 DD EE D7 13 C7 7C A3  | █ÕÕ█◄'ßÌéÝî×‼Ç|£
000001B0: BC F4 1E C2 C9 2C FA D9|A6 C7 59 AE 3A 44 46 BC  | ¼ô.ÂÉ,úÙ¦ÇY®:DF¼
000001C0: 06 86 AA 67 5F C0 73 16|29 4F 93 DD A8 18 39 A0  | ─█ªg_Às┬)O█Ý¨↑9
000001D0: 34 22 5E 00 79 D5 DA 59|4D 7E 60 5D B2 75 5C 1A  | 4"^ yÕÚYM~`]²u\→
000001E0: ED F4 C7 62 47 E2 3E DE|08 BD 13 DC 36 A0 82 15  | íôÇbGâ>Þ◘½‼Ü6 █┴
000001F0: E1 FA C5 EB 4C BE 6F 8C|47 D5 69 69 8E 87 B4 90  | áúÅëL¾o█GÕii██´.
00000200: 9C B1 D9 20 DB 4F B2 D0|79 D1 AA 1D 58 81 D4 C1  | █±Ù ÛO²ÐyÑª.X.ÔÁ
00000210: 5A 84 15 6E 86 C6 23 15|8A 65 E0 84 44 EA BF 14  | Z█┴n█Æ#┴█eà█Dê¿¶
00000220: EE 54 79 4C F4 76 CC 41|13 00 00 00 00 00 00 00  | îTyLôvÌA‼
00000230: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000240: D5 40 46 0F 3E 81 47 11|D9 9B 6F 30 C7 EF CA 92  | Õ@F☼>.G◄Ù█o0ÇïÊ'
00000250: 73 EF 77 8C AC 5D C3 48|2A B9 94 02 EA 62 00 00  | sïw█¬]ÃH*¹█┐êb
00000260: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000270: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000280: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000290: 00 00 00 00 00 00 00 00|00 00 00 B7 C7 99 DE B8  |            ·Ç█Þ¸
000002A0: 18 8D AA 00 00 00 00 00|00 00 00 00 00 00 00 00  | ↑.ª
000002B0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000002C0: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
000002D0: 00 00 00 00 00 00 00 00|00 00 65 35 57 6C 64 58  |           e5WldX
000002E0: 6A 30 4D 45 50 71 6F 76|6E 6B 65 7A 0A 4A 31 4F  | j0MEPqovnkez.J1O
000002F0: 0A 70 57 4E 54 42 37 68|32 42 53 68 31 6E 47 4C  | .pWNTB7h2BSh1nGL
00000300: 70 0A 45 47 46 33 4B 0A|30 4B 4A 71 79 54 0A 64  | p.EGF3K.0KJqyT.d
00000310: 74 65 6C 63 32 71 38 73|5A 35 78 58 0A 63 4E 0A  | telc2q8sZ5xX.cN.
00000320: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000330: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00000340: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
```

The fields values should be:

1. Size: 0x0046
2. Beginning offset: 0x02DA
3. Type: 0x0001 (i.e. TEXT)
4. Elements' size: 0x000A
5. Name size (NS): 0x0B
6. Name: SECT_TEXT_5

PART 2. Assignment's Requirements
You are required to write a C program named "*coordinator.c*" that perform the following steps:

1.
    1. *(5 min)* Checks if it is given one command line parameter and validate that it corresponds to an accessible file. If not, displays an error message and terminates. We will refer from now on to that file as "CONFIG_FILE".
    2. (*30 min*) **Reads** one by one **all the text lines** from the CONFIG_FILE. Each line must consist in two strings of characters separated by one or more spaces. The first string corresponds to a directory path and the second one to a username. We will refer from now on to the first parameter (i.e. the directory path) as USER_DIR and to the second as USER_NAME. For each read line the program must:
        1. **validate** the line format**;**
        2. **validate** the USER_DIR by checking that it corresponds to an **existing**, **accessible** directory;
        3. **display the line**, prepended by the "VALID: " or "INVALID: " string, depending if the validation checks passed or not, respectively.
    3. (*10 min*) Calls a function named "`authenticate()`", which reads from STDIN (i.e. keyboard) a line representing a username and **checks if that username exists** in the CONFIG_FILE. If not, terminates displaying on the screen the following error message: "`ERROR: invalid user`". If the username exists, but its associated USER_DIR in CONFIG_FILE is invalid, then the program terminates with the error message "`ERROR: Invalid user directory`". Your program must restrict in the following an authenticated user's accesses only to the files located in the directory associated to that user and only if that directory is a valid one.
    4. (*20 min*) Calls repeatedly a function called "`get_and_execute_command_line()`", which reads from STDIN (i.e. keyboard) a line of characters, which correspond to a command line your program must execute. Each command line consists of a command name followed by its arguments, separated by one or more spaces, like the format illustrated below:
       ```
       CMD_NAME ARG1 ARG2 ARG3 ...
       ```

       The supported commands and their syntax are the following:
       ```
       INFO file_path
       SEARCH file_name [-R] [section_name section_size]
       SECT_DISPLAY file_path section_name
       SECT_HASH file_path section_name
       EXIT
       ```

       Your program must separate the read command line into elements, i.e. command name and its arguments, then based on the command name executes the following operations:
        1. (3*0 min*) for the INFO command, calls a function named "`info(char* file_path)`", which checks if the given "`file_path`" corresponds to an existing and valid SF file (the path is supposed to be given relative to the USER_DIR directory) and if so, displays on the screen on separate lines the following information: (1) the path of the file, (2) its number of sections and for each section, all on the same line and separated by spaces, (3) the section's name, (4) section's type, (5) section element's size, (6) section's number of elements and (7) section's size in bytes. If the path does not correspond to a SF file, the program must display the message "`ERROR: not a SF file`".
        2. (*30 min*) for the SEARCH command, calls a function named "`search_files_and_sections(char* file_name, char*`

`section_name, unsigned int section_size, unsigned char recursive)`", which searches in the authenticated user's USER_DIR for all the SF files containing in their name the given "`file_name`" string, having a section whose name contains the given "`section_name`" string and a size of minimum "`section_size`" bytes. The arguments "`section_name`" and "`section_size`" could be missing, in which case any section name or section size, respectively, is accepted. If the "-R" option is specified, the search must be done in the entire file tree starting from the authenticated user's USER_DIR. The function must display on the screen for the matching files on different lines: (1) their path relative to the USER_DIR directory, (2) the matching section's name and (3) size in bytes.

3. (30 *min*) for the SECT_DISPLAY command, calls a function named "`sect_display(char* file_path, char* sect_name)`", which checks if the file corresponding to the given "`file_path`" corresponds to a SF file and the given "`sect_name`" to a TEXT section and if so, displays on the screen the contents of that section. If the checks do not succeed, the function displays on the screen the message "`ERROR: not a SF file or not a TEXT section`".
4. (*30 min*) for the SECT_HASH command, calls a function named "`sect_hash(char* file_path, char* sect_name)`", which checks if the file corresponding to the given "`file_path`" corresponds to a SF file and the given "`sect_name`" to a BINARY section and if so, calculate the XOR value of all its elements (taking into account the element size). Then, displays on the screen as three strings separated by a space the element size, the number of elements and the resulted XOR value as a hexadecimal value. For instance, for a binary section containing three elements of two-bytes size like "`000345A5116B`", the output would be "`2 3 54CD`". If the checks do not succeed, the function displays on the screen the message "`ERROR: not a SF file or not a BINARY section`".
5. (*3 min*) for the EXIT command, calls a function named "exit`()`", which terminates the program.
6. (*2 min*) for an unrecognized command, displays the "`ERROR: Unrecognized command`" message.

IMPORTANT NOTES

1. **YOU MUST UPLOAD ONLY YOUR "coordinator.c" file!**
2. **You program must successfully compile with a command line like "`gcc -Wall -Werror coordinator.c -o coordinator`" to be further evaluated.** The "`-Wall`" option says the compiler to report anything that could be wrong from its point of view, while the "`-Werror`" says it to report any warning as an error and do not accept it. As a consequence, your program must compile without any error and warning at all in order to be admitted for evaluation.
3. You are required and **restricted to use only the OS system calls**, i.e. low-level functions, not higher-level one, in your entire solution, in all lab assignments. For instance, regarding the file accesses, you **MUST use system calls** like `open()`, `read()`, `write()` etc., but NOT higher-level functions like `fopen()`, `fgets()`, `fscanf()`, `fprintf()` etc. The only accepted exceptions from this requirement are the functions to read from STDIN or display to STDOUT / STDERR, like `scanf()`, `printf()`, `perror()` and functions for string manipulation and conversion like `sscanf()`, `snprintf()`.
4. For testing purposes you can download the following archive, containing a directory with few testing SF files. Please note that there could be both valid and invalid SF format files in the given archive. Start by reading the README file in the given archive.

5. A **SF format validation must check for both file signatures and the header structure integrity**, i.e. checks if the number of meta-data areas corresponds to the reported number of sections.
6. It could be supposed that any SF file contains only sections with distinct name, having no two sections with the same name. Though, there could be different sections whose name contain a common sub-string (like, for instance, "SECT_1", "SECT_2", "MY_SECT").
7. The text sections were generated using the Linux convention, so contain just one byte (value 10) for the new-line ('\n') separator. Take care that the last line could end without a '\n', i.e. the last character in the file could be different by '\n'.
8. For string tokenization (i.e. separate a string into elements based on specific separators, like spaces) we recommend you using the `strtok()` or `strtok_r()` functions.
9. You can automate your program runs by redirecting its STDIN to a text file prepared to contain the commands normally given from the keyboard. This way you can easily and efficiently run the same test more times for debugging purposes. This is actually the way we will test your solutions.
10. Extra credit (1 point) will be given to you if you use regular expressions to specify string patterns for file and section names for the SEARCH command (see manual page REGEX(3) for details).