

Raport 4: Systemy operacyjne – systemy plików

Wstęp

To laboratorium ma na celu zaznajomienie się z podstawowymi koncepcjami na temat systemów plików na przykładzie systemu plików ext4. W ramach trzech zadań wykonamy następujące operacje:

1. Stworzymy plik o wielkości 100 MB, który będzie „udawał” obraz partycji dysku.
2. W tym pliku stworzymy system plików ext4 z domyślnymi parametrami.
3. Przebadamy stworzony, „dziewiczy” system plików za pomocą komendy `dumpe2fs` oraz edytora szesnastkowego. Poznamy dzięki temu praktycznie strukturę superbloku.
4. Nauczymy się tworzyć tzw. loop device i montować systemy plików pod Linuxem.
5. Zmodyfikujemy system plików tworząc w nim jeden plik.
6. Spróbujemy w strukturach systemu pliku odnaleźć informacje o tym pliku

Zadanie 1

Wykonano komendę

```
$dd if=/dev/zero of=moj-fs.raw count=100 bs=1M
```

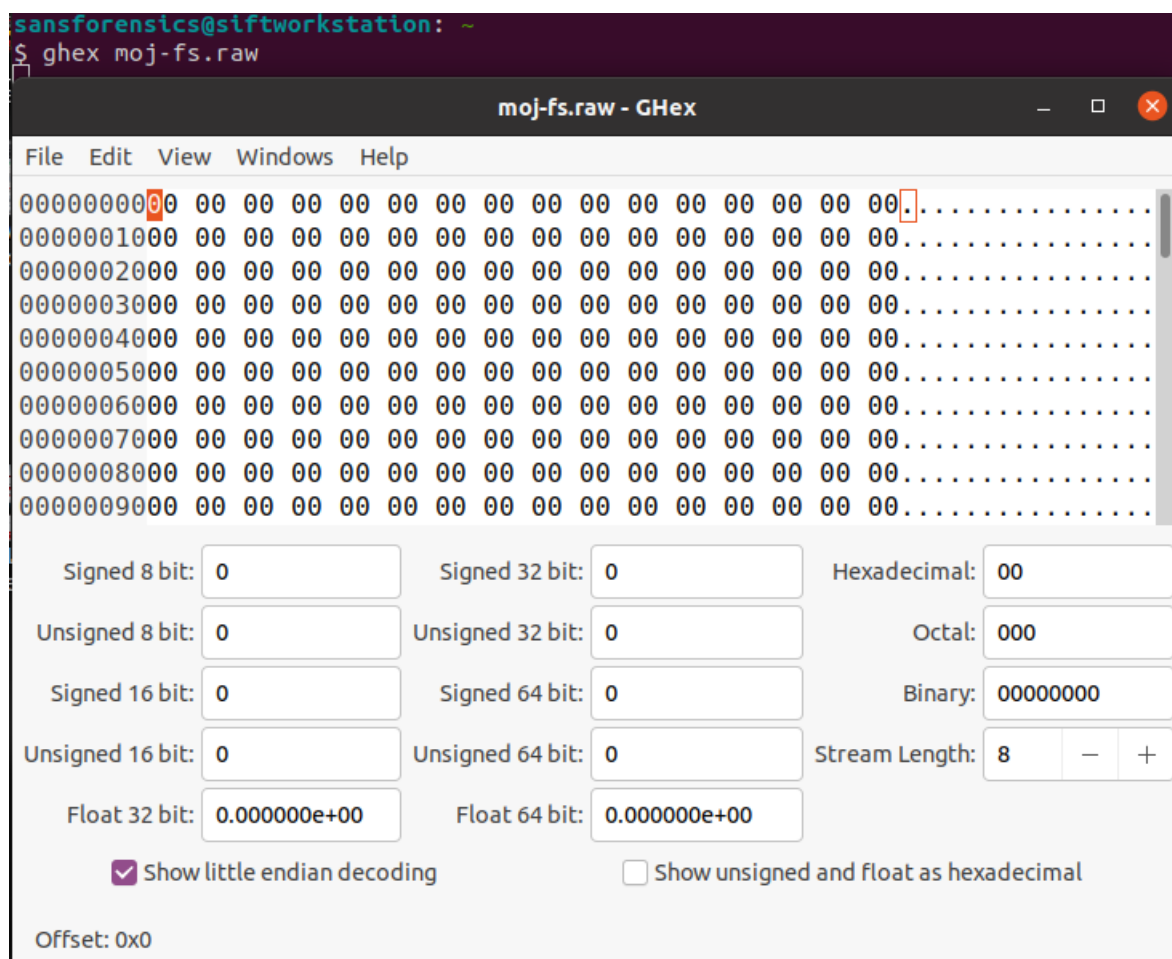
Co ona robi?

- `dd` to komenda do kopiowania plików – ale pozwala na kopiowanie dowolnej części ich zawartości
- `if=/dev/zero` mówi, że źródłowym plikiem jest File `/dev/zero` – jest to plik specjalny (urządzenie znakowe), które zwraca tylko zera
- `of=moj-fs.raw` mówi, że plikiem docelowym (wynikowym) jest File `moj-fs.raw`
- `count=100` określa, że kopiujemy 100 bloków, których rozmiar podany jest jako `bs`
- `bs=1M` określa, że operujemy na blokach o wielkości 1 MiB

```
sansforensics@siftworkstation: ~  
$ dd if=/dev/zero of=moj-fs.raw count=100 bs=1M  
100+0 records in  
100+0 records out  
104857600 bytes (105 MB, 100 MiB) copied, 0.0574834 s, 1.8 GB/s  
sansforensics@siftworkstation: ~  
$
```

Na tym etapie możesz zweryfikować, że plik składa się z samych zer za pomocą `xxd` albo `ghex` lub innego edytora szesnastkowego:

Wpisano komendę: `$ghex moj-fs.raw`



Stwórz system plików w tak stworzonym pliku:

```
$ mkfs.ext4 moj-fs.raw
```

```
sansforensics@siftworkstation: ~
$ mkfs.ext4 moj-fs.raw
mke2fs 1.45.5 (07-Jan-2020)
Discarding device blocks: done
Creating filesystem with 25600 4k blocks and 25600 inodes

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
```

Tak właśnie – komenda `mkfs.ext4` tworzy system plików ext4. Wykonanie jej na dużym pliku powoduje, że tworzy ona nowy system plików w tym pliku – zupełnie tak, jakbyśmy wykonali ją na fizycznej partycji na dysku. Na tym etapie będziemy chcieli ustalić parametry naszego systemu plików i zrobimy to na dwa sposoby. Pierwszy sposób – prosty. Uruchom komendę:

```
$ dumpe2fs moj-fs.raw
```

```
sansforensics@siftworkstation: ~
$ dumpe2fs moj-fs.raw > moj-fs.txt
dumpe2fs 1.45.5 (07-Jan-2020)
```

Dane uzyskane z komendy:

```

Filesystem volume name:  <none>
Last mounted on:         <not available>
Filesystem UUID:         f8812850-e085-48b8-a34b-bd99a2321b55
Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)
Filesystem features:     has_journal ext_attr resize_inode dir_index
filetype extent 64bit flex_bg sparse_super large_file huge_file
dir_nlink extra_isize metadata_csum
Filesystem flags:        signed_directory_hash
Default mount options:   user_xattr acl
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:             25600
Block count:             25600
Reserved block count:    1280
Free blocks:             23754
Free inodes:             25589
First block:             0
Block size:              4096
Fragment size:           4096
Group descriptor size:   64
Reserved GDT blocks:     12
Blocks per group:        32768
Fragments per group:     32768
Inodes per group:        25600
Inode blocks per group:  800
Flex block group size:   16
Filesystem created:      Mon Mar 28 11:36:30 2022
Last mount time:         n/a
Last write time:         Mon Mar 28 11:36:30 2022
Mount count:             0
Maximum mount count:     -1
Last checked:            Mon Mar 28 11:36:30 2022
Check interval:          0 (<none>)
Lifetime writes:         65 kB
Reserved blocks uid:     0 (user root)
Reserved blocks gid:     0 (group root)
First inode:             11
Inode size:              128
Journal inode:           8
Default directory hash:  half_md4
Directory Hash Seed:     c49b4947-92e5-4a0f-a5bb-2c3517d9439c
Journal backup:          inode blocks
Checksum type:           crc32c
Checksum:                0x965b528f
Journal features:        (none)
Journal size:            4096k
Journal length:          1024
Journal sequence:        0x00000001
Journal start:           0

```

```

Group 0: (Blocks 0-25599) csum 0xb7f5 [ITABLE_ZEROED]
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-13
  Block bitmap at 14 (+14), csum 0xedc12315
  Inode bitmap at 30 (+30), csum 0x0a40f45f

```

```

Inode table at 46-845 (+46)
23754 free blocks, 25589 free inodes, 2 directories, 25589 unused
inodes
Free blocks: 1846-25599
Free inodes: 12-25600

```

Tabela wartości, które zostały podane na wykładzie:

Offset	Rozmiar (B)	Opis
0x00	4	Liczba i-węzłów
0x04	4	Liczba bloków
0x08	4	Liczba bloków zarezerwowanych dla roota
0x0C	4	Liczba wolnych bloków
0x10	4	Liczba wolnych i-węzłów
0x14	4	Pierwszy blok z danymi
0x18	4	Rozmiar bloku
0x1C	4	Rozmiar klastra
0x20	4	Liczba bloków w grupie

Drugi sposób – trudny. Wybierz cztery parametry, które znajdują się w superbloku (mogą być nieomówione na wykładzie) i odnajdź je i ich wartości analizując zapis szesnastkowy w edytorze szesnastkowym (ghex ,xxd).

Parametr 1: Liczba i-węzłów, offset 0x00, wartość: 0 x 00 64 00 00, DEC: 25 600

000003A000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

000003B000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

000003C000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

000003D000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

000003E000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

000003F000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....

0000040000 00 64 00 00 00 64 00 00 00 05 00 00 CA 5C 00 00 .d...d.....\..

00000410F5 63 00 00 00 00 00 00 00 02 00 00 00 02 00 00 00 .c.....

0000042000 80 00 00 00 80 00 00 00 64 00 00 02 A6 41 62d....Ab

00000430A4 A9 41 62 01 00 FF FF 53 EF 01 00 01 00 00 00 ..Ab....S.....

Signed 8 bit: 0 Signed 32 bit: 6553600 Hexadecimal: 00

Unsigned 8 bit: 0 Unsigned 32 bit: 6553600 Octal: 000

Signed 16 bit: 0 Signed 64 bit: 6553600 Binary: 00000000

Unsigned 16 bit: 0 Unsigned 64 bit: 6553600 Stream Length: 8 - +

Float 32 bit: 9.183550e-39 Float 64 bit: 6.953356e-309

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x403; 0x4 bytes from 0x400 to 0x403 selected

Parametr 2: Liczba i-węzłów, offset 0x0C, wartość: 0 x 00 00 5C CA, DEC: 23 754

File	Edit	View	Windows	Help
000003D000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
000003E000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
000003F000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
0000040000	64	00	00	00 64 00 00 00 05 00 00 CA 5C 00 00 .d...d.....\..
00000410F5	63	00	00	00 00 00 02 00 00 00 02 00 00 00 00 .c.....
0000042000	80	00	00	00 80 00 00 00 64 00 00 02 A6 41 62d....Ab
00000430A4	A9	41	62	01 00 FF FF 53 EF 01 00 01 00 00 00 ..Ab....S.....
00000440BE	9D	41	62	00 00 00 00 00 00 00 01 00 00 00 ..Ab.....
0000045000	00	00	00	0B 00 00 00 80 00 00 00 3C 00 00 00<...
00000460C2	02	00	00	6B 04 00 00 F8 81 28 50 E0 85 48 B8k.....(P..H.

Signed 8 bit:	0	Signed 32 bit:	6550784	Hexadecimal:	00
Unsigned 8 bit:	0	Unsigned 32 bit:	6550784	Octal:	000
Signed 16 bit:	-2816	Signed 64 bit:	6550784	Binary:	00000000
Unsigned 16 bit:	62720	Unsigned 64 bit:	6550784	Stream Length:	8 - +
Float 32 bit:	9.179604e-39	Float 64 bit:	3.236517e-317		

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x40F; 0x4 bytes from 0x40C to 0x40F selected

Parametr 3: Pierwszy blok z danymi, offset = 0x14, wartość: 0 x 00 00 00 00, DEC: 0

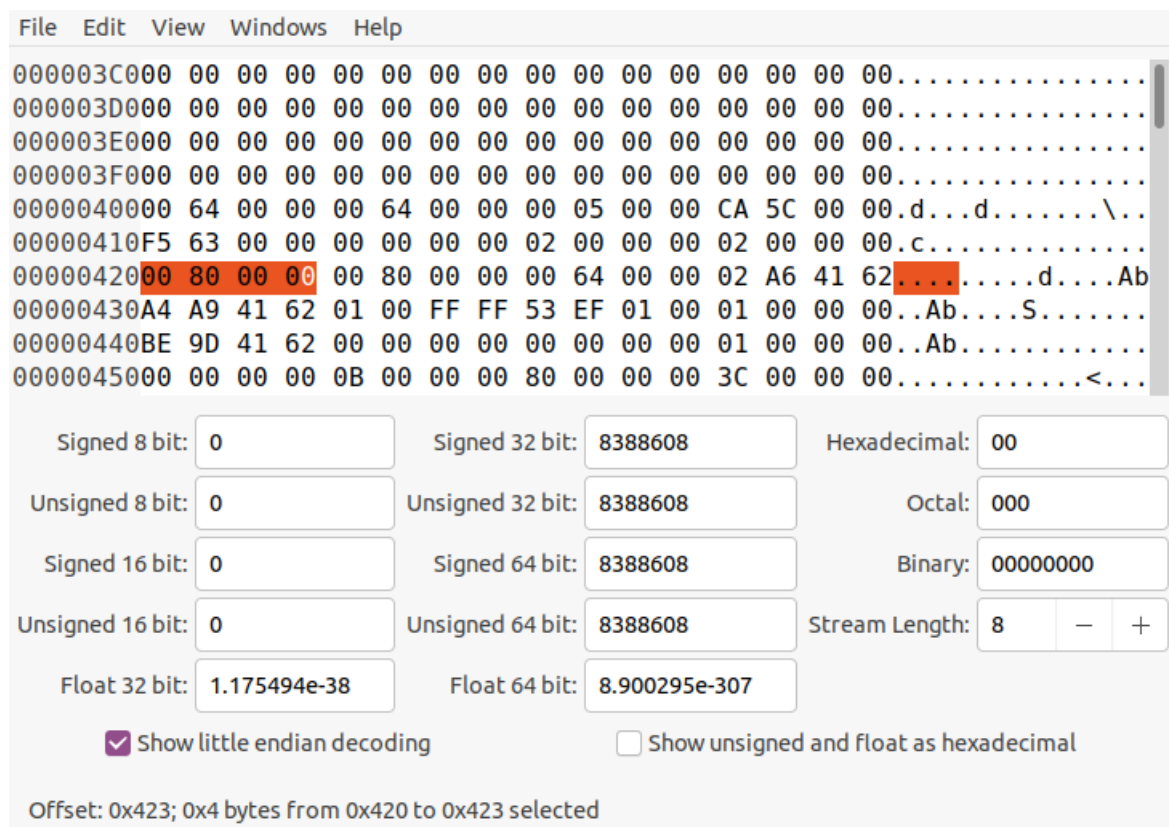
File	Edit	View	Windows	Help
000003C000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
000003D000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
000003E000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
000003F000	00	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.....
0000040000	64	00	00	00 64 00 00 00 05 00 00 CA 5C 00 00 .d...d.....\..
00000410F5	63	00	00	00 00 00 00 02 00 00 00 02 00 00 00 .c.....
0000042000	80	00	00	00 80 00 00 00 64 00 00 02 A6 41 62d....Ab
00000430A4	A9	41	62	01 00 FF FF 53 EF 01 00 01 00 00 00 ..Ab....S.....
00000440BE	9D	41	62	00 00 00 00 00 00 00 01 00 00 00 ..Ab.....
0000045000	00	00	00	0B 00 00 00 80 00 00 00 3C 00 00 00<...

Signed 8 bit:	0	Signed 32 bit:	512	Hexadecimal:	00
Unsigned 8 bit:	0	Unsigned 32 bit:	512	Octal:	000
Signed 16 bit:	512	Signed 64 bit:	512	Binary:	00000000
Unsigned 16 bit:	512	Unsigned 64 bit:	512	Stream Length:	8 - +
Float 32 bit:	7.174648e-43	Float 64 bit:	1.086462e-311		

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x417; 0x4 bytes from 0x414 to 0x417 selected

Parametr 4: Liczba bloków w grupie, offset = 0x20, wartość: 0x 00 00 80 00, DEC: 32 768



Zadanie 2

Teraz zamontujemy nasz stworzony system plików. Najpierw stwórzmy urządzenie blokowe wskazujące na plik z systemem plików:

```
$ sudo losetup --find --show moj-fs.raw
```

```
sansforensics@siftworkstation: ~  
$ sudo losetup --find --show moj-fs.raw  
/dev/loop7
```

Wypisaną przez system wartość należy zapamiętać i użyć w kolejnej komendzie. Teraz wykonujemy właściwe montowanie:

```
$ sudo mount /dev/loop18 /mnt/usb
```

```
sansforensics@siftworkstation: ~  
$ sudo mount /dev/loop7 /mnt/usb  
sansforensics@siftworkstation: ~
```

Poprawne wykonanie powyższych komend możemy z grubsza zweryfikować następująco:

```
$ ls -li /mnt/usb/
```

```
sansforensics@siftworkstation: ~  
$ ls -li /mnt/usb/  
total 16  
11 drwx----- 2 root root 16384 Mar 28 11:36 lost+found
```

Wypisanie parametrów tego pliku:

`$ ls -l /dev/loop18`

```
sansforensics@siftworkstation: ~  
$ ls -l /dev/loop7  
brw-rw---- 1 root disk 7, 7 Mar 28 12:10 /dev/loop7  
sansforensics@siftworkstation: ~
```

Wyjście z komendy `$ lsblk -f`

```
sansforensics@siftworkstation: ~  
$ lsblk -f  
NAME FSTYPE LABEL UUID                                FSAVAIL FSUSE% MOUNTPOINT  
loop0  
  squash                                0    100% /snap/codi  
loop1  
  squash                                0    100% /snap/core  
loop2  
  squash                                0    100% /snap/snap  
loop3  
  squash                                0    100% /snap/core  
loop4  
  ext4      f8812850-e085-48b8-a34b-bd99a2321b55  
loop5  
  ext4      f8812850-e085-48b8-a34b-bd99a2321b55  
loop6  
  ext4      f8812850-e085-48b8-a34b-bd99a2321b55  
loop7  
  ext4      f8812850-e085-48b8-a34b-bd99a2321b55  85.8M    0% /mnt/usb  
sda  
├─sda1  
│  swap      18ce61b9-bd58-4c0c-9b73-c0ddd2e83f5e          [SWAP]  
└─sda2  
   ext4      8097329e-0ecb-45ce-85c7-2fb00e14a44a 443.9G    2% /  
sansforensics@siftworkstation: ~
```

Komenda **lsblk** wypisuje informacje o wszystkich dostępnych lub podanych urządzeniach blokowych.

Z kolei **-f** odpowiada za wypisanie informacji o systemach plików.

Czym jest katalog **lost+found** i dlaczego jego i-węzeł ma numer 11?

Jest to katalog, w którym można znaleźć odzyskane kawałki uszkodzonych plików np. w sytuacji, gdy użytkownik nagle zamknie komputer w czasie, gdy jest on uruchomiony, a pliki są zapisywane na dysku twardym.

Ma numer 11, ponieważ jest to pierwszy i-węzeł dostępny dla użytkownika. W katalogu tym system przechowuje pliki odnalezione podczas wykonywania testów dysku – są to pliki oraz fragmenty „ocalonych” lub uszkodzonych plików w systemie plików.

Zadanie 3

Wejdź do katalogu `/mnt/usb` (tam, gdzie w poprzednim zadaniu zamontowaliśmy system plików). Stwórz w nim plik tekstowy (czysty tekst) – np. za pomocą komendy:

`$ sudo gedit alamakota.txt`

```
sansforensics@siftworkstation: ~  
$ sudo gedit alamakota.txt
```

W pliku tym wprowadź jakiś długi tekst (np. skopiuj treść jakiejś strony internetowej). Zapisz plik i wyjdź z katalogu.

```
alamakota.txt [Read-Only]
1 ABBA
2 Dancing Queen
3
4 Ooh
5 You can dance
6 You can jive
7 Having the time of your life
8 Ooh, see that girl
9 Watch that scene
10 Digging the dancing queen
11 Friday night and the lights are low
12 Looking out for a place to go
13 Where they play the right music
14 Getting in the swing
15 You come to look for a king
16 Anybody could be that guy
17 Night is young and the music's high
18 With a bit of rock music
19 Everything is fine
20 You're in the mood for a dance
21 And when you get the chance
22 You are the dancing queen
23 Young and sweet
24 Only seventeen
25 Dancing queen
26 Feel the beat from the tambourine, oh yeah
27 You can dance
28 You can jive
29 Having the time of your life
30 Ooh, see that girl
31 Watch that scene
32 Digging the dancing queen
33 You're a teaser, you turn 'em on
34 Leave 'em burning and then you're gone
35 Looking out for another
36 Anyone will do
37 You're in the mood for a dance
38 And when you get the chance
39 You are the dancing queen
40 Young and sweet
41 Only seventeen
42 Dancing queen
43 Feel the beat from the tambourine, oh yeah
44 You can dance
45 You can jive
46 Having the time of your life
47 Ooh, see that girl
```

Przy zapisywaniu pliku pojawiło się ostrzeżenie:

```
$ sudo gedit alamakota.txt

(gedit:7087): Tepl-WARNING **: 16:55:15.482: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
```

Odmontuj system plików:

```
$ sudo umount /mnt/usb
```

```
sansforensics@siftworkstation: ~
$ sudo umount /mnt/usb
sansforensics@siftworkstation: ~
```

Po odmontowaniu użyj komendy *debugfs*, aby poznać szczegóły informacji o stworzonym pliku:

```
sansforensics@siftworkstation: ~
$ debugfs moj-fs.raw
debugfs 1.45.5 (07-Jan-2020)
debugfs: stat alamakota.txt
```

Wartości uzyskane po wpisaniu komendy:


```

Inode: 12   Type: regular   Mode: 0644   Flags: 0x80000
Generation: 1390409572   Version: 0x00000001
User: 0    Group: 0    Size: 1094
File ACL: 0
Links: 1   Blockcount: 8
Fragment: Address: 0    Number: 0    Size: 0
ctime: 0x624a28e4 -- Sun Apr 3 23:08:20 2022
atime: 0x624a28f6 -- Sun Apr 3 23:08:38 2022
mtime: 0x624a28e4 -- Sun Apr 3 23:08:20 2022
Inode checksum: 0x00005d84
EXTENTS:
(0):1846

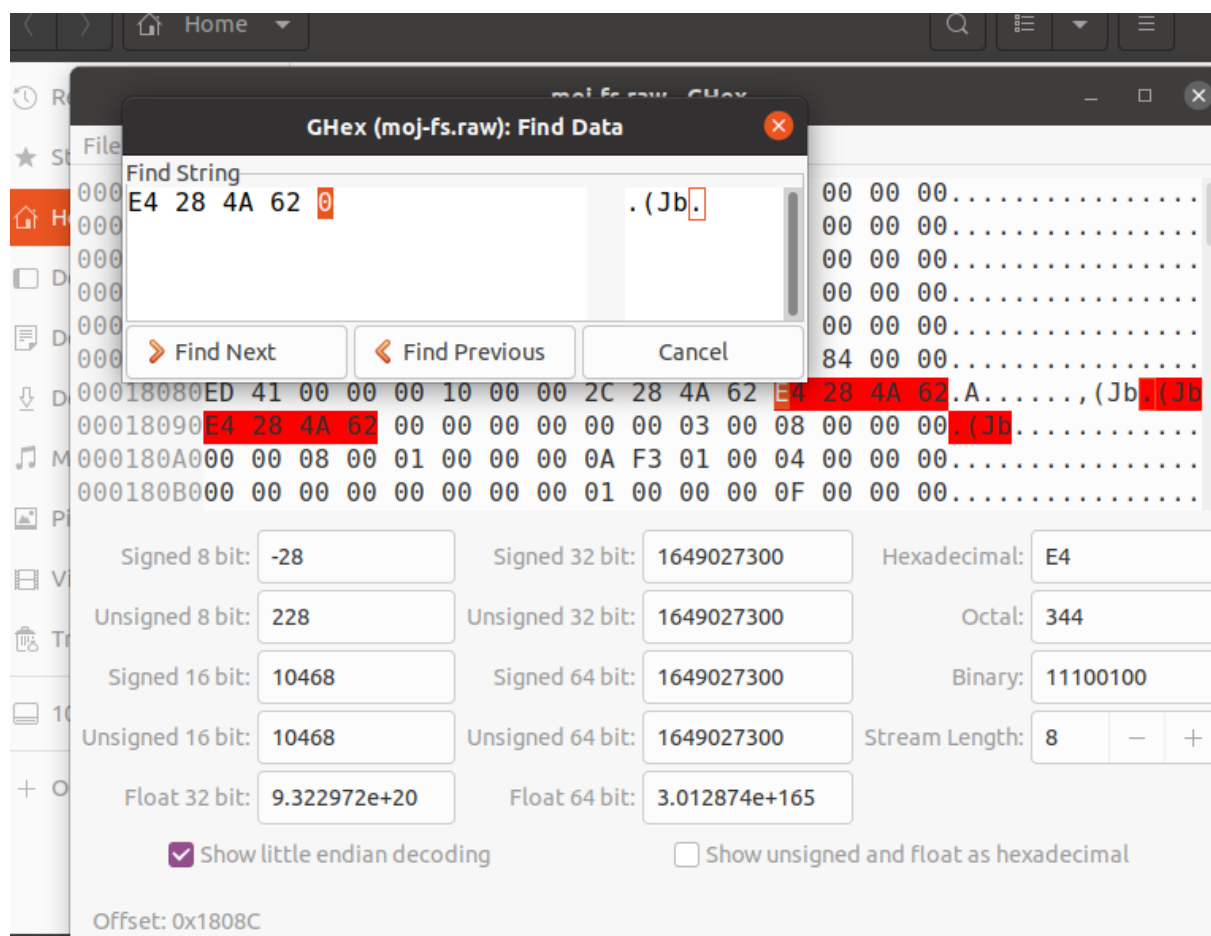
```

Wyświetli to informacje o stworzonym przez Ciebie pliku (np. numer i-węzła). Otwórz ponownie plik `moj-fs.raw` za pomocą `ghex` lub innego edytora szesnastkowego. Postaraj się zlokalizować i-węzeł opisujący stworzony przez Ciebie plik.

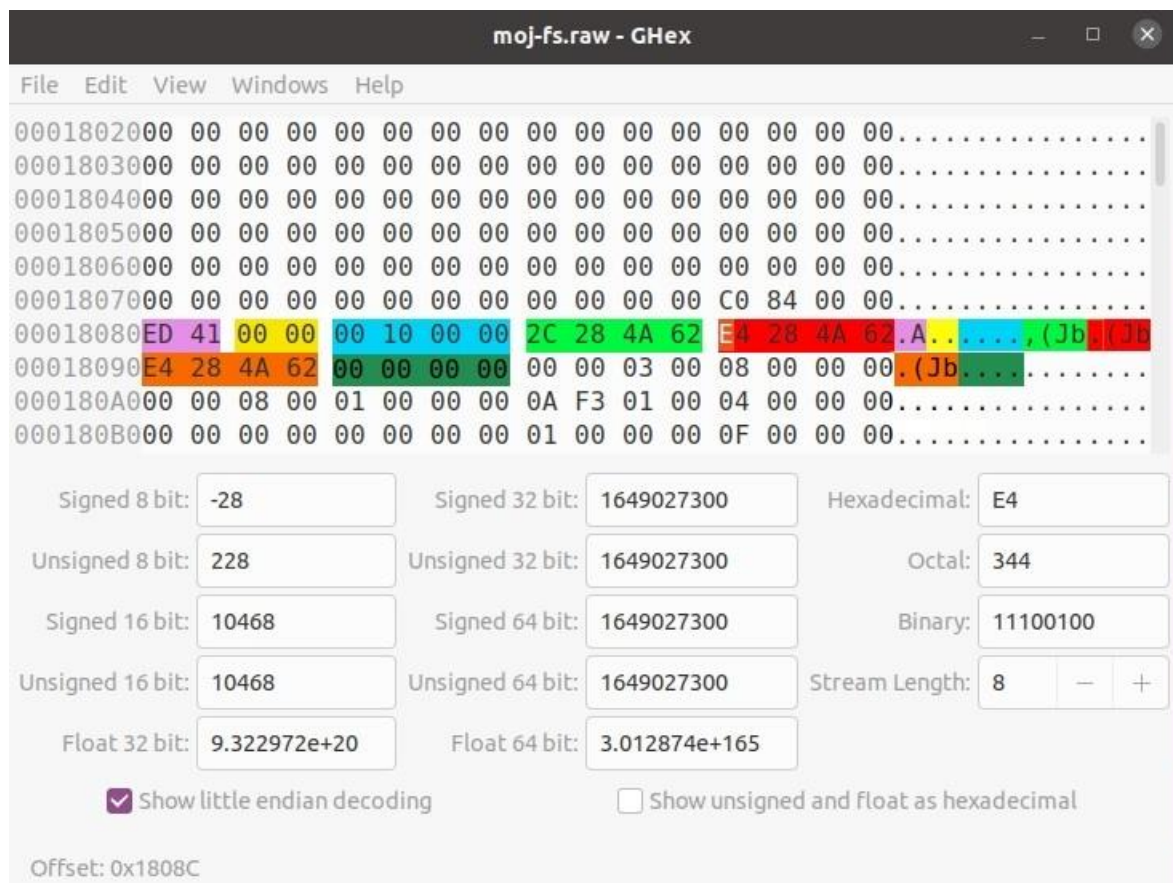
Należy znaleźć i-węzeł Skorzystano z jedynej możliwej danej jaką jest wyświetlana data utworzenia / dostępu / modyfikacji, która została wywołana w komendzie `$ debugfs moj-fs.raw`, ma ona wartość `0x624a28e4`.

Następnie skonwertowano wartość, aby uzyskać niezbędny format, który następnie zostanie użyty w edytorze szesnastkowym (`e4 28 4a 62`).

Zlokalizowanie i-węzła:



Zaznaczenie odpowiednich bajtów:



Różowy – 2 bajty – tryb pliku

Żółty – 2 bajty – niższe 16 bitów UID właściciela

Niebieski – 4 bajty – niższe 32 bity rozmiaru w bajtach

Jasnozielony – 4 bajty – czas dostępu (access time)

Czerwony – 4 bajty – czas zmiany (change time)

Pomarańczowy – 4 bajty – czas modyfikacji (modification time)

Ciemnozielony – 4 bajty – czas usunięcia (deletion time)

Zrzut ekranu z komendy `$strings`

```
sansforensics@siftworkstation: ~  
$ strings moj-fs.raw  
/mnt/usb  
lost+found  
alamakota.txt  
,(Jb  
lost+found  
alamakota.txt  
/mnt/usb  
,(Jb  
,(Jb  
7:_1  
,(Jb  
You can dance  
You can jive  
Having the time of your life  
Ooh, see that girl  
Watch that scene  
Digging the dancing queen  
Friday night and the lights are low  
Looking out for a place to go  
Where they play the right music
```