CS365 Digital Forensics (Spr 2022)

Assignment 4: Parse FAT32 Filesystem Images

Corrections

-Feb25@629pm: Corrected language to say that entries in the FAT are 32 *bits* long. And the entire entry should be equal to 0x0 to be considered unallocated, not just the first byte.

Introduction

In this assignment, you'll parsing a FAT32 filesystem (not FAT12 or FAT16) to recover data from it. It's a big assignment, and you'll need to spend a number of days on it. I have tried to cut down on the number of intricacies but giving you a library for three important functions that are annoying to code up. I will also attempt to give as much partial credit as possible using the autograder.

I have given you a structure to start with, which will help you and the autograder.

Part 0. Formatting (15 points)

As always, you will need to use black, isort, and flake8. (5 points each)

If you want isort to be compatible with black, be sure to give isort these arguments: `--profile black`.

If you want flake8 to be compatible with black be sure to give flake8 these arguments: `--ignore=E203,W503`.

Part 1. The Reserved Sector (60 points).

Fill out the `_parse_reserved_sector()` function. The documentation provides a list of items you need to parse out or compute. If you don't get full credit from the autograder for this function, I doubt the rest of your code will work correctly. The list of items you need to store in `self.boot` is listed in the stubbed-out function's documentation. Doing this part of the assignment will be similar to completing the MBR parsing. (13 keys, 5 points each).

You should also at this point store the contents of fat0 in `self.fat` before continuing.

Example result of filling out `self.boot`:

```
{
    "bytes_per_sector": 512,
    "sectors_per_cluster": 2,
    "reserved_sectors": 414,
    "number_of_fats": 2,
    "total_sectors": 2060288,
    "sectors_per_fat": 7985,
    "root_dir_first_cluster": 2,
    "bytes_per_cluster": 1024,
    "fat0_sector_start": 414,
    "fat0_sector_end": 8398,
    "data_start": 16384,
    "data_end": 2060287
}
```

Part 2. Parse the Cluster Chain (20 points)

Next fill out `self._get_sectors(cluster:int)`. This function reads the values stored in `self.fat` to determine the run of clusters that start from a given entry. Cluster chains are described in Carrier. Each entry is *32 bits (hence FAT32). If the entry is set to `0x0`* [bolded part fixed from previous version], then it is unallocated and you can return an empty list `[]`. (If the entry is `0x0FFFFFF7` then it is damaged, but this will never happen in our assignment.)

Otherwise, the cluster number is allocated. Convert the cluster number to a sector number, and return a list of all sectors in the cluster. For example if there are 2 sectors per cluster, and the cluster C becomes sector S, then you would return a list `[S, S+1]`. (Carrier tells you how to convert clusters to sectors in Fat.) But that's just the start of the list — if the cluster value is `C <= 0x0FFFFFF8` then add to the list with the next cluster; otherwise you hit the end-of-file marker and the chain is ended. Note that you have to iterate with a while loop or equivalent instead of using recursion. If you follow the cluster chains recursively, you'll hit Python's recursion limit.

If you convert just the first cluster into a list and don't follow the chain correctly, then some of Part 3 will work but not all of it.

Part 3. Parse the Root directory (and all others recursively)

We are going to fill out `parse_dir(self, cluster: int, parent="")`.

This part of the assignment is harder, and so we'll do it in pieces. Your goal is to grab the clusters holding the root directory and to parse each entry. The results of parsing becomes a dictionary for each entry, and you'll return a list of dictionaries.

3.1 Retrieving data (20 points)

To start, note that you know how to grab the clusters for the root directory. You computed `self.boot['root_dir_first_cluster']` in Part 1. And in Part 2, you figured out how to get back a list of sectors for a give cluster. So, now fill out this function (ignore the default `ignore_unallocated` argument for now.):

```
_retrieve_data(self, cluster: int, ignore_unallocated=False) -> bytes
```

It gets the list of sectors associated with the cluster (using `_get_sectors()`) and for each, seeks to that part of the fat32image file and reads in the sector. It returns all the data as one continuous set of bytes.

So starting from `self.boot['root_dir_first_cluster']` you should be able to get a list of sectors that are read in as bytes() objects. Concatenate them together and you are off to the races: that's the root directory data that you've read in.

3.2 Parse Basic Directory Data (30 points total)

The directory data is a list of directory entries, each 32 bytes. Parse each entry.

Part 3.2.a (30 points): There are a set of 7 keys to compute for any/every time of entry. If the entry type is a volume, long file name, or if it is the "." or ".." entries of a directory (always the first two entries listed) then you are done. I would get this part working first.

- `parent`: parent directory. (For the root directory, the parent is the empty string).
- `dir_cluster`: cluster number of the directory (this was passed in as an argument)
- `entry_num`: entry number of the directory (within its cluster). You can use a counter (or Python's `enumerate()`)
- `dir_sectors`: sectors associated with the directory (converted from cluster with `_get_sectors()`)
- `entry_type`: type of entry (vol, lfn, dir, or other). I provided this for you as `hw4utils.get_entry_type()`.
- `name`: name of the entry. I provided this for you as `hw4utils.parse_name()`.
- `deleted`: whether the entry is marked as deleted. You need to compute this from Carrier's Table 10.5

Here's some example output

```
{"parent": "", "dir_cluster": 2, "entry_num": 0, "dir_sectors": [16384, 16385],
"entry_type": "vol", "name": "ASSIGN4", "deleted": false}


{"parent": "", "dir_cluster": 2, "entry_num": 1, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": " Information", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 2, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": "System Volume", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 4, "dir_sectors": [16386,
 16387], "entry_type": "0x20", "name": "WPSETT~1.DAT", "deleted": false}
```

Get that running first. Or the next parts will be harder to debug.

3.3 Parse Directory entries recursively (20 points)

If the entry type is a directory, then add an 8th key to the dictionary `content_cluster` that states the first cluster holding the content for the entry. Next, call this function recursively on the entry, adding to the results. When you call it recursively, set `parent + "/" + name`. Lovely. Don't call it recursively on the first two entries ("." and "..") as it's a never-ending loop!

3.4 Parse Files Content (30 points)

This last part is really entire goal of all the work we've done on the assignment. Let's recover content, deleted content, and slack data. You are almost done!

If we aren't a volume, LFN, or a directory, then we must be a file. In this part we are going to add 4 more keys to the dictionary for the entry. (So all of the above four more.)

- `filesize`: size of the entry. Last row of Carrier's Table 10.5. (5 points)
- `content_sectors` : the list of sectors associated with the content of this entry. This is not the sectors that we read the entry from; it's the file content. (5 points)
- `content`: the first 128 bytes of the entry's content (10 points)
- `slack`: the slack data (up to 32 bytes) (10 points)

You already have code to get the content_sectors! Use `_get_sectors()`.

So we are going to retrieve the contents of the file, or actually just up to 128 bytes of the file (some files aren't that long). So fill out this function stub:

```
_get_content(self, cluster: int, filesize: int) -> tuple[str, Optional[str]]
```

What does that return type mean? It means return two values (e.g., `return a, b`) but the second value might be None. This is straightforward if the file is not deleted. What you do is read in all the data from the clusters — in fact you already coded that up in `_retrieve_data()`! So, from the first sector return the first 128 bytes (or up to the filesize). From the last sector, return the first 32 bytes past the filesize.

Now consider the harder case when the file is deleted; that is, if `content_cluster` is unallocated. In that case, grab the data in that first sector anyway (adjust your code to handle that case — because above we said return `[]` when a cluster is unallocated. This is what the ignore_unallocated flag is for). In this case don't return the slack because we really don't know the correct filesize for the data written for this cluster.

See below for some sample output.

Sample output

Here's the output from Carrier's tools

```
% fsstat fat32-1.empty.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT32

OEM Name: MSDOS5.0
Volume ID: 0x767f2399
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): ASSIGN4
File System Type Label: FAT32
Next Free Sector (FS Info): 16386
Free Sector Count (FS Info): 2043902

Sectors before file system: 32768

File System Layout (in sectors)
Total Range: 0 - 2060287
* Reserved: 0 - 413
** Boot Sector: 0
** FS Info Sector: 1
** Backup Boot Sector: 6
* FAT 0: 414 - 8398
* FAT 1: 8399 - 16383
* Data Area: 16384 - 2060287
** Cluster Area: 16384 - 2060287
*** Root Directory: 16384 - 16385

METADATA INFORMATION
--------------------------------------------
Range: 2 - 32702470
Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 1024
Total Cluster Range: 2 - 1021953

FAT CONTENTS (in sectors)
--------------------------------------------
16384-16385 (2) -> EOF
16386-16387 (2) -> EOF
```

```
16388-16389 (2) -> EOF
```

```
% fls -rp fat32-1.empty.dd
r/r 3:  ASSIGN4     (Volume Label Entry)
d/d 6:  System Volume Information
r/r 39: System Volume Information/WPSettings.dat
v/v 32702467:   $MBR
v/v 32702468:   $FAT1
v/v 32702469:   $FAT2
V/V 32702470:   $OrphanFiles
```

Output of my program. I've added spaces to improve readability.

```
{
    "bytes_per_sector": 512,
    "sectors_per_cluster": 2,
    "reserved_sectors": 414,
    "number_of_fats": 2,
    "total_sectors": 2060288,
    "sectors_per_fat": 7985,
    "root_dir_first_cluster": 2,
    "bytes_per_cluster": 1024,
    "fat0_sector_start": 414,
    "fat0_sector_end": 8398,
    "data_start": 16384,
    "data_end": 2060287
}
{"parent": "", "dir_cluster": 2, "entry_num": 0, "dir_sectors": [16384, 16385],
"entry_type": "vol", "name": "ASSIGN4", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 1, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": " Information", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 2, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": "System Volume", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 0, "dir_sectors": [16386,
 16387], "entry_type": "dir", "name": ".", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 1, "dir_sectors": [16386,
 16387], "entry_type": "dir", "name": "..", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 2, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "t", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 3, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "WPSettings.da", "deleted": false}
```

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 4, "dir_sectors": [16386, 16387], "entry_type": "0x20", "name": "WPSETT~1.DAT", "deleted": false, "content_clusters": 4, "filesize": 12, "content_sectors": [16388, 16389], "content": "b'\\x0c\\x00\\x00\\x00\\x8a4`\\xdfG\\xec\\x8d\\x95'", "slack": "b'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00'"}

{"parent": "", "dir_cluster": 2, "entry_num": 3, "dir_sectors": [16384, 16385], "entry_type": "dir", "name": "SYSTEM~1", "deleted": false, "content_cluster": 3
}

Here's an image with lots of entries.

```
% fsstat  fat32-5.add-images.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT32

OEM Name: MSDOS5.0
Volume ID: 0x767f2399
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): ASSIGN4
File System Type Label: FAT32
Next Free Sector (FS Info): 17646
Free Sector Count (FS Info): 2042642

Sectors before file system: 32768

File System Layout (in sectors)
Total Range: 0 - 2060287
* Reserved: 0 - 413
** Boot Sector: 0
** FS Info Sector: 1
** Backup Boot Sector: 6
* FAT 0: 414 - 8398
* FAT 1: 8399 - 16383
* Data Area: 16384 - 2060287
** Cluster Area: 16384 - 2060287
*** Root Directory: 16384 - 16385

METADATA INFORMATION
--------------------------------------------
Range: 2 - 32702470
Root Directory: 2

CONTENT INFORMATION
```

```
--------------------------------------------
Sector Size: 512
Cluster Size: 1024
Total Cluster Range: 2 - 1021953

FAT CONTENTS (in sectors)
--------------------------------------------
16384-16385 (2) -> EOF
16386-16387 (2) -> EOF
16388-16389 (2) -> EOF
16390-16391 (2) -> EOF
16392-16393 (2) -> EOF
16394-16395 (2) -> EOF
16396-16397 (2) -> EOF
16398-17479 (1082) -> EOF
17480-17645 (166) -> EOF
```

```
% fls -rp fat32-5.add-images.dd
fls -rp fat32-5.add-images.dd
r/r 3:  ASSIGN4     (Volume Label Entry)
d/d 6:  System Volume Information
r/r 39: System Volume Information/WPSettings.dat
r/r 42: System Volume Information/IndexerVolumeGuid
r/r 7:  empty.txt
r/r 8:  nonempty.txt
r/r * 9:    _scii.txt
r/r 10: ascii.txt
r/r 11: video.mp4
r/r 14: equus_quagga.png
v/v 32702467:   $MBR
v/v 32702468:   $FAT1
v/v 32702469:   $FAT2
V/V 32702470:   $OrphanFiles
```

```
% python3.9 fsstat-solution.py fat32-5.add-images.dd
```

```
{
    "bytes_per_sector": 512,
    "sectors_per_cluster": 2,
    "reserved_sectors": 414,
    "number_of_fats": 2,
    "total_sectors": 2060288,
    "sectors_per_fat": 7985,
    "root_dir_first_cluster": 2,
    "bytes_per_cluster": 1024,
    "fat0_sector_start": 414,
    "fat0_sector_end": 8398,
    "data_start": 16384,
    "data_end": 2060287
}

{"parent": "", "dir_cluster": 2, "entry_num": 0, "dir_sectors": [16384, 16385],
"entry_type": "vol", "name": "ASSIGN4", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 1, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": " Information", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 2, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": "System Volume", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 0, "dir_sectors": [16386,
 16387], "entry_type": "dir", "name": ".", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 1, "dir_sectors": [16386,
 16387], "entry_type": "dir", "name": "..", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 2, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "t", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 3, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "WPSettings.da", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 4, "dir_sectors": [16386,
 16387], "entry_type": "0x20", "name": "WPSETT~1.DAT", "deleted": false, "conten
t_clusters": 4, "filesize": 12, "content_sectors": [16388, 16389], "content": "b
'\\x0c\\x00\\x00\\x00\\x8a4`\\xdfG\\xec\\x8d\\x95'", "slack": "b'\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00'"}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 5, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "Guid", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 6, "dir_sectors": [16386,
 16387], "entry_type": "lfn", "name": "IndexerVolume", "deleted": false}

{"parent": "/SYSTEM~1", "dir_cluster": 3, "entry_num": 7, "dir_sectors": [16386,
 16387], "entry_type": "0x20", "name": "INDEXE~1.", "deleted": false, "content_c
lusters": 8, "filesize": 76, "content_sectors": [16396, 16397], "content": "b'{\\x
00D\\x007\\x00E\\x00F\\x00F\\x00E\\x002\\x000\\x00-\\x008\\x004\\x001\\x000\\x
00-\\x004\\x00C\\x003\\x00A\\x00-\\x008\\x00D\\x008\\x005\\x00-\\x00E\\x000\\x00
C\\x001\\x001\\x008\\x003\\x006\\x00B\\x002\\x00E\\x005\\x00}\\x00'", "slack": "
b'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x
00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x
```

```
00'"}

{"parent": "", "dir_cluster": 2, "entry_num": 3, "dir_sectors": [16384, 16385],
"entry_type": "dir", "name": "SYSTEM~1", "deleted": false, "content_cluster": 3
}

{"parent": "", "dir_cluster": 2, "entry_num": 4, "dir_sectors": [16384, 16385],
"entry_type": "0x20", "name": "EMPTY.TXT", "deleted": false, "content_cluster":
 5, "filesize": 6, "content_sectors": [16390, 16391], "content": "b'\\xff\\xfe\\
r\\x00\\n\\x00'", "slack": "b'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00'"}

{"parent": "", "dir_cluster": 2, "entry_num": 5, "dir_sectors": [16384, 16385],
"entry_type": "0x20", "name": "NONEMPTY.TXT", "deleted": false, "content_cluster
s": 6, "filesize": 68, "content_sectors": [16392, 16393], "content": "b'\\xff\\x
feT\\x00h\\x00i\\x00s\\x00 \\x00f\\x00i\\x00l\\x00e\\x00 \\x00c\\x00o\\x00n\\x00
t\\x00e\\x00n\\x00t\\x00 \\x00i\\x00s\\x00 \\x00n\\x00o\\x00t\\x00 \\x00e\\x00m
\\x00p\\x00t\\x00y\\x00.\\x00\\r\\x00\\n\\x00'", "slack": "b'\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00'"}

{"parent": "", "dir_cluster": 2, "entry_num": 6, "dir_sectors": [16384, 16385],
"entry_type": "0x20", "name": "_SCII.TXT", "deleted": true, "content_cluster":
0}

{"parent": "", "dir_cluster": 2, "entry_num": 7, "dir_sectors": [16384, 16385],
"entry_type": "0x20", "name": "ASCII.TXT", "deleted": false, "content_cluster":
 7, "filesize": 39, "content_sectors": [16394, 16395], "content": "b'This is non
-unicode content in a file. '", "slack": "b'\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00'"}

{"parent": "", "dir_cluster": 2, "entry_num": 8, "dir_sectors": [16384, 16385],
"entry_type": "0x20", "name": "VIDEO.MP4", "deleted": false, "content_cluster":
 9, "filesize": 552999, "content_sectors": [16398, 16399, 16400, 16401, 16402, 1
6403, 16404, 16405, 16406, 16407, 16408, 16409, 16410, 16411, 16412, 16413, 1641
4, 16415, 16416, 16417, 16418, 16419, 16420, 16421, 16422, 16423, 16424, 16425,
16426, 16427, 16428, 16429, 16430, 16431, 16432, 16433, 16434, 16435, 16436, 164
37, 16438, 16439, 16440, 16441, 16442, 16443, 16444, 16445, 16446, 16447, 16448,
 16449, 16450, 16451, 16452, 16453, 16454, 16455, 16456, 16457, 16458, 16459, 16
460, 16461, 16462, 16463, 16464, 16465, 16466, 16467, 16468, 16469, 16470, 16471
, 16472, 16473, 16474, 16475, 16476, 16477, 16478, 16479, 16480, 16481, 16482, 1
6483, 16484, 16485, 16486, 16487, 16488, 16489, 16490, 16491, 16492, 16493, 1649
4, 16495, 16496, 16497, 16498, 16499, 16500, 16501, 16502, 16503, 16504, 16505,
16506, 16507, 16508, 16509, 16510, 16511, 16512, 16513, 16514, 16515, 16516, 165
```

```
17, 16518, 16519, 16520, 16521, 16522, 16523, 16524, 16525, 16526, 16527, 16528,
 16529, 16530, 16531, 16532, 16533, 16534, 16535, 16536, 16537, 16538, 16539, 16
540, 16541, 16542, 16543, 16544, 16545, 16546, 16547, 16548, 16549, 16550, 16551
, 16552, 16553, 16554, 16555, 16556, 16557, 16558, 16559, 16560, 16561, 16562, 1
6563, 16564, 16565, 16566, 16567, 16568, 16569, 16570, 16571, 16572, 16573, 1657
4, 16575, 16576, 16577, 16578, 16579, 16580, 16581, 16582, 16583, 16584, 16585,
16586, 16587, 16588, 16589, 16590, 16591, 16592, 16593, 16594, 16595, 16596, 165
97, 16598, 16599, 16600, 16601, 16602, 16603, 16604, 16605, 16606, 16607, 16608,
 16609, 16610, 16611, 16612, 16613, 16614, 16615, 16616, 16617, 16618, 16619, 16
620, 16621, 16622, 16623, 16624, 16625, 16626, 16627, 16628, 16629, 16630, 16631
, 16632, 16633, 16634, 16635, 16636, 16637, 16638, 16639, 16640, 16641, 16642, 1
6643, 16644, 16645, 16646, 16647, 16648, 16649, 16650, 16651, 16652, 16653, 1665
4, 16655, 16656, 16657, 16658, 16659, 16660, 16661, 16662, 16663, 16664, 16665,
16666, 16667, 16668, 16669, 16670, 16671, 16672, 16673, 16674, 16675, 16676, 166
77, 16678, 16679, 16680, 16681, 16682, 16683, 16684, 16685, 16686, 16687, 16688,
 16689, 16690, 16691, 16692, 16693, 16694, 16695, 16696, 16697, 16698, 16699, 16
700, 16701, 16702, 16703, 16704, 16705, 16706, 16707, 16708, 16709, 16710, 16711
, 16712, 16713, 16714, 16715, 16716, 16717, 16718, 16719, 16720, 16721, 16722, 1
6723, 16724, 16725, 16726, 16727, 16728, 16729, 16730, 16731, 16732, 16733, 1673
4, 16735, 16736, 16737, 16738, 16739, 16740, 16741, 16742, 16743, 16744, 16745,
16746, 16747, 16748, 16749, 16750, 16751, 16752, 16753, 16754, 16755, 16756, 167
57, 16758, 16759, 16760, 16761, 16762, 16763, 16764, 16765, 16766, 16767, 16768,
 16769, 16770, 16771, 16772, 16773, 16774, 16775, 16776, 16777, 16778, 16779, 16
780, 16781, 16782, 16783, 16784, 16785, 16786, 16787, 16788, 16789, 16790, 16791
, 16792, 16793, 16794, 16795, 16796, 16797, 16798, 16799, 16800, 16801, 16802, 1
6803, 16804, 16805, 16806, 16807, 16808, 16809, 16810, 16811, 16812, 16813, 1681
4, 16815, 16816, 16817, 16818, 16819, 16820, 16821, 16822, 16823, 16824, 16825,
16826, 16827, 16828, 16829, 16830, 16831, 16832, 16833, 16834, 16835, 16836, 168
37, 16838, 16839, 16840, 16841, 16842, 16843, 16844, 16845, 16846, 16847, 16848,
 16849, 16850, 16851, 16852, 16853, 16854, 16855, 16856, 16857, 16858, 16859, 16
860, 16861, 16862, 16863, 16864, 16865, 16866, 16867, 16868, 16869, 16870, 16871
, 16872, 16873, 16874, 16875, 16876, 16877, 16878, 16879, 16880, 16881, 16882, 1
6883, 16884, 16885, 16886, 16887, 16888, 16889, 16890, 16891, 16892, 16893, 1689
4, 16895, 16896, 16897, 16898, 16899, 16900, 16901, 16902, 16903, 16904, 16905,
16906, 16907, 16908, 16909, 16910, 16911, 16912, 16913, 16914, 16915, 16916, 169
17, 16918, 16919, 16920, 16921, 16922, 16923, 16924, 16925, 16926, 16927, 16928,
 16929, 16930, 16931, 16932, 16933, 16934, 16935, 16936, 16937, 16938, 16939, 16
940, 16941, 16942, 16943, 16944, 16945, 16946, 16947, 16948, 16949, 16950, 16951
, 16952, 16953, 16954, 16955, 16956, 16957, 16958, 16959, 16960, 16961, 16962, 1
6963, 16964, 16965, 16966, 16967, 16968, 16969, 16970, 16971, 16972, 16973, 1697
4, 16975, 16976, 16977, 16978, 16979, 16980, 16981, 16982, 16983, 16984, 16985,
16986, 16987, 16988, 16989, 16990, 16991, 16992, 16993, 16994, 16995, 16996, 169
97, 16998, 16999, 17000, 17001, 17002, 17003, 17004, 17005, 17006, 17007, 17008,
 17009, 17010, 17011, 17012, 17013, 17014, 17015, 17016, 17017, 17018, 17019, 17
020, 17021, 17022, 17023, 17024, 17025, 17026, 17027, 17028, 17029, 17030, 17031
, 17032, 17033, 17034, 17035, 17036, 17037, 17038, 17039, 17040, 17041, 17042, 1
```

```
7043, 17044, 17045, 17046, 17047, 17048, 17049, 17050, 17051, 17052, 17053, 1705
4, 17055, 17056, 17057, 17058, 17059, 17060, 17061, 17062, 17063, 17064, 17065,
17066, 17067, 17068, 17069, 17070, 17071, 17072, 17073, 17074, 17075, 17076, 170
77, 17078, 17079, 17080, 17081, 17082, 17083, 17084, 17085, 17086, 17087, 17088,
 17089, 17090, 17091, 17092, 17093, 17094, 17095, 17096, 17097, 17098, 17099, 17
100, 17101, 17102, 17103, 17104, 17105, 17106, 17107, 17108, 17109, 17110, 17111
, 17112, 17113, 17114, 17115, 17116, 17117, 17118, 17119, 17120, 17121, 17122, 1
7123, 17124, 17125, 17126, 17127, 17128, 17129, 17130, 17131, 17132, 17133, 1713
4, 17135, 17136, 17137, 17138, 17139, 17140, 17141, 17142, 17143, 17144, 17145,
17146, 17147, 17148, 17149, 17150, 17151, 17152, 17153, 17154, 17155, 17156, 171
57, 17158, 17159, 17160, 17161, 17162, 17163, 17164, 17165, 17166, 17167, 17168,
 17169, 17170, 17171, 17172, 17173, 17174, 17175, 17176, 17177, 17178, 17179, 17
180, 17181, 17182, 17183, 17184, 17185, 17186, 17187, 17188, 17189, 17190, 17191
, 17192, 17193, 17194, 17195, 17196, 17197, 17198, 17199, 17200, 17201, 17202, 1
7203, 17204, 17205, 17206, 17207, 17208, 17209, 17210, 17211, 17212, 17213, 1721
4, 17215, 17216, 17217, 17218, 17219, 17220, 17221, 17222, 17223, 17224, 17225,
17226, 17227, 17228, 17229, 17230, 17231, 17232, 17233, 17234, 17235, 17236, 172
37, 17238, 17239, 17240, 17241, 17242, 17243, 17244, 17245, 17246, 17247, 17248,
 17249, 17250, 17251, 17252, 17253, 17254, 17255, 17256, 17257, 17258, 17259, 17
260, 17261, 17262, 17263, 17264, 17265, 17266, 17267, 17268, 17269, 17270, 17271
, 17272, 17273, 17274, 17275, 17276, 17277, 17278, 17279, 17280, 17281, 17282, 1
7283, 17284, 17285, 17286, 17287, 17288, 17289, 17290, 17291, 17292, 17293, 1729
4, 17295, 17296, 17297, 17298, 17299, 17300, 17301, 17302, 17303, 17304, 17305,
17306, 17307, 17308, 17309, 17310, 17311, 17312, 17313, 17314, 17315, 17316, 173
17, 17318, 17319, 17320, 17321, 17322, 17323, 17324, 17325, 17326, 17327, 17328,
 17329, 17330, 17331, 17332, 17333, 17334, 17335, 17336, 17337, 17338, 17339, 17
340, 17341, 17342, 17343, 17344, 17345, 17346, 17347, 17348, 17349, 17350, 17351
, 17352, 17353, 17354, 17355, 17356, 17357, 17358, 17359, 17360, 17361, 17362, 1
7363, 17364, 17365, 17366, 17367, 17368, 17369, 17370, 17371, 17372, 17373, 1737
4, 17375, 17376, 17377, 17378, 17379, 17380, 17381, 17382, 17383, 17384, 17385,
17386, 17387, 17388, 17389, 17390, 17391, 17392, 17393, 17394, 17395, 17396, 173
97, 17398, 17399, 17400, 17401, 17402, 17403, 17404, 17405, 17406, 17407, 17408,
 17409, 17410, 17411, 17412, 17413, 17414, 17415, 17416, 17417, 17418, 17419, 17
420, 17421, 17422, 17423, 17424, 17425, 17426, 17427, 17428, 17429, 17430, 17431
, 17432, 17433, 17434, 17435, 17436, 17437, 17438, 17439, 17440, 17441, 17442, 1
7443, 17444, 17445, 17446, 17447, 17448, 17449, 17450, 17451, 17452, 17453, 1745
4, 17455, 17456, 17457, 17458, 17459, 17460, 17461, 17462, 17463, 17464, 17465,
17466, 17467, 17468, 17469, 17470, 17471, 17472, 17473, 17474, 17475, 17476, 174
77, 17478, 17479], "content": "b'\\x00\\x00\\x00\\x18ftypmp42\\x00\\x00\\x00\\x0
0isommp42\\x00\\x00\\x18|moov\\x00\\x00\\x00lmvhd\\x00\\x00\\x00\\x00\\xd0\\x07
\\xcaL\\xd0\\x07\\xcaL\\x00\\x00\\x02X\\x00\\x00)@\\x00\\x01\\x00\\x00\\x01\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00@\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00'", "slack": "b
'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x0
```

```
0\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x0
0'"}

{"parent": "", "dir_cluster": 2, "entry_num": 9, "dir_sectors": [16384, 16385],
"entry_type": "lfn", "name": "png", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 10, "dir_sectors": [16384, 16385],
 "entry_type": "lfn", "name": "equus_quagga.", "deleted": false}

{"parent": "", "dir_cluster": 2, "entry_num": 11, "dir_sectors": [16384, 16385],
 "entry_type": "0x20", "name": "EQUUS_~1.PNG", "deleted": false, "content_cluste
rs": 550, "filesize": 84992, "content_sectors": [17480, 17481, 17482, 17483, 174
84, 17485, 17486, 17487, 17488, 17489, 17490, 17491, 17492, 17493, 17494, 17495,
 17496, 17497, 17498, 17499, 17500, 17501, 17502, 17503, 17504, 17505, 17506, 17
507, 17508, 17509, 17510, 17511, 17512, 17513, 17514, 17515, 17516, 17517, 17518
, 17519, 17520, 17521, 17522, 17523, 17524, 17525, 17526, 17527, 17528, 17529, 1
7530, 17531, 17532, 17533, 17534, 17535, 17536, 17537, 17538, 17539, 17540, 1754
1, 17542, 17543, 17544, 17545, 17546, 17547, 17548, 17549, 17550, 17551, 17552,
17553, 17554, 17555, 17556, 17557, 17558, 17559, 17560, 17561, 17562, 17563, 175
64, 17565, 17566, 17567, 17568, 17569, 17570, 17571, 17572, 17573, 17574, 17575,
 17576, 17577, 17578, 17579, 17580, 17581, 17582, 17583, 17584, 17585, 17586, 17
587, 17588, 17589, 17590, 17591, 17592, 17593, 17594, 17595, 17596, 17597, 17598
, 17599, 17600, 17601, 17602, 17603, 17604, 17605, 17606, 17607, 17608, 17609, 1
7610, 17611, 17612, 17613, 17614, 17615, 17616, 17617, 17618, 17619, 17620, 1762
1, 17622, 17623, 17624, 17625, 17626, 17627, 17628, 17629, 17630, 17631, 17632,
17633, 17634, 17635, 17636, 17637, 17638, 17639, 17640, 17641, 17642, 17643, 176
44, 17645], "content": "b\"\\x89PNG\\r\\n\\x1a\\n\\x00\\x00\\x00\\rIHDR\\x00\\x0
0\\x00\\xab\\x00\\x00\\x00\\xb1\\x08\\x06\\x00\\x00\\x00\\xbb\\xa5M\\xe4\\x00\\x
00\\x00\\x01sRGB\\x00\\xae\\xce\\x1c\\xe9\\x00\\x00\\x00\\x04gAMA\\x00\\x00\\xb1
\\x8f\\x0b\\xfca\\x05\\x00\\x00\\x00\\tpHYs\\x00\\x00\\x0e\\xc2\\x00\\x00\\x0e\\
xc2\\x01\\x15(J\\x80\\x00\\x00\\xff\\xa5IDATx^l\\xfdu\\xb8\\x95e\\xd7\\xfd\\x0f
\\xcf\\xbdvww\\xd1\\xdd-!\\x8a\\xa0X\\xd8b'*\\x8a\\x8d\\x9d\\xa8\\x18\\x88b\\x80
\\xdd(\"", "slack": "b''"}
```