



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)

## «تمرین دوم درس سیستم های چند رسانه ای»

استاد خرسندی

نام:

مهدیه سادات بنیس (۹۸۳۱۷۰۲)

## بررسی محتوای باینری چند فایل تصویری با فرمت مختلف:

مشاهده محتوای باینری فایل های تصویری با **binary viewer** در سایت [hexed.it](http://hexed.it)

:tiff



بخشی از محتوای باینری آن:

Screenshot of the hexed.it binary viewer interface showing the file sample\_640x426.tiff.

The menu bar includes: New file, Open file, Save as, Undo, Redo, Tools, Settings, Help, PATREON, and PayPal.

The main window displays the file information and data inspector.

**File Information:**

- File Name: sample\_640x426.tiff
- File Size: 818,184 bytes (800 KIB)

**Data Inspector (Little-endian):**

Type	Unsigned (+)	Signed (-)
8-bit Integer	73	73
16-bit Integer	18761	18761
24-bit Integer	2771273	2771273
32-bit Integer	2771273	2771273
64-bit Integer (+)	3512974013253961	
64-bit Integer (-)	3512974013253961	
16-bit Float P.	10.57031	
32-bit Float P.	3.8833806e-39	
64-bit Float P.	1.735639774681834e-308	
LEB128 (+)	73	
LEB128 (-)	-55	
MS-DOS DateTime	1980-01-10 09:10:18 Local	
OLE 2.0 DateTime	1899-12-30 00:00:00.000 UTC	
UNIX 32-bit DateTime	1970-02-02 01:47:53 UTC	
Macintosh HFS	1904-02-02 05:13:37 Local	

The data pane shows the first few bytes of the file in hex, ASCII, and EBCDIC formats. The first byte is highlighted in blue.

```
File Name: sample_640x426.tiff
File Size: 818,184 bytes (800 KIB)

49 49 2A 00 08 7B 0C 00 B2 82 90 AD 7D 8B AA 7A I*..{.éÉ;}{éÉ;}{éÉ;}{éÉ;
8A AA 7A 8A AB 7B 8B A7 77 87 A2 71 84 9F 6E 81 è-zè%;{íowçóqäfnü
9F 70 82 9F 70 82 9E 6F 83 9B 6C 80 98 69 7D 96 fpéfpéPöälcçýíjú
67 7B 94 64 7A 94 64 7A 91 63 7D 92 64 7E 88 5C g{ödzödzaç}Ad-é\
75 82 58 70 88 61 79 81 5B 72 8B 65 7C C3 A0 B6 uéxpiayü[ríe]|\á|
85 63 7B 7C 5A 72 79 57 6F 7D 5B 73 7E 5B 73 78 ác{[ZryWo}[s~[sx
55 60 7A 54 6D 80 5A 73 7B 57 71 79 55 6F 78 54 UmzTmçZs{WqyUoxT
6E 78 54 6E 78 56 6F 77 55 6E 75 53 6C 72 50 69 nxTnxVowUnuslRpí
76 56 6E 74 54 6C 71 51 69 71 51 69 71 53 6B 72 vVntTlqQiqQiQskr
54 6C 72 54 6C 71 53 6B 73 52 6D 70 4F 6A 6E 4E TlrTlqSksRmpOjnN
66 70 50 68 71 53 6B 71 53 6B 6D 50 66 69 4C 62 fpPhqSkqSkmpfLb
AE 91 A5 75 58 6C 6E 53 64 74 59 6A 5F 44 53 6D «æñùXlnSdtvj_DSm
52 61 6D 52 61 67 4C 5B 69 50 63 68 4F 62 68 4F RamRagL[iPchObh0
64 69 50 65 68 4F 65 66 4D 63 66 4D 63 66 4D 63 diPehoefMcfMcfcMc
65 4D 63 66 4E 64 66 4F 63 66 4F 63 66 4F 61 65 eMcfnfdOfcfcOfcOae
4E 60 65 4E 5E 63 4E 5D 5F 4A 5D 60 4D 60 5F 4A N'eN^cN]_J`_M`_J
5D 63 4E 61 69 54 67 5A 45 58 55 40 53 6A 55 68 ]cNaItgZEUXUoSjUh
61 4A 5E 53 3C 50 77 5E 73 71 58 6D 5C 43 58 5E aJ^S<Pw^sqXm\cx^
45 5A 66 4D 62 56 3F 53 5B 43 59 5D 47 5C 60 4A EZFmbV2S{CVJG}`J
5F 5F 49 5E 5D 47 5C 5C 46 5B 5A 44 59 58 42 57 __I`]GV\{[ZDyXBW
5D 47 5C 59 43 58 58 42 57 59 43 58 59 43 58 5A ]G\YCXxBWYCYXCZ
44 59 5E 48 5D 65 4F 64 50 3A 4F 5A 44 59 53 3D DY^H]eoDp:OZDYS=
52 61 4B 60 57 41 56 54 3E 53 5A 44 59 56 40 55 RaK`WAVT>SZDyV@U
58 42 57 58 42 57 57 41 56 57 41 56 57 XBWXBWAVWAVWAVW
41 56 57 41 56 57 41 56 5A 44 59 55 3F 54 54 3E AVWAVWAVZDyU?TT>
```

در خط اول محتوای باینری:

I\*..{.éÉ;}{éÉ;}{éÉ;}{éÉ;

:png



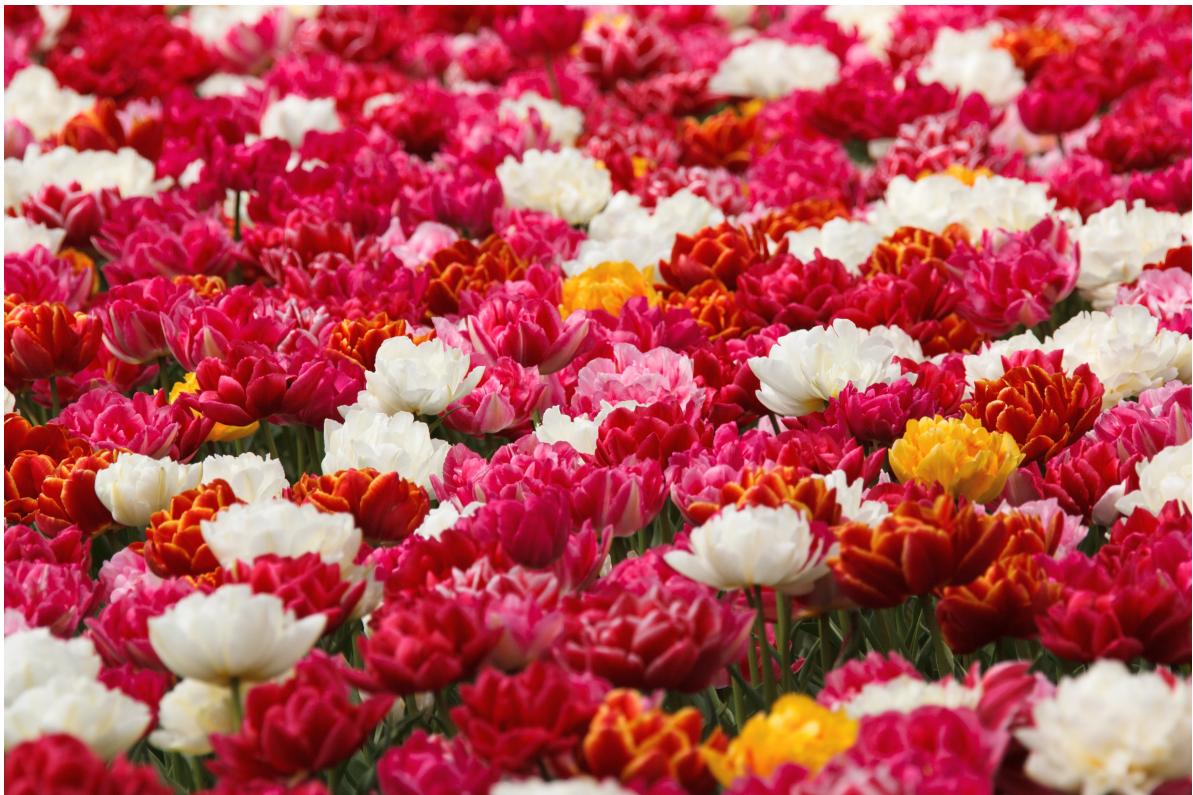
بخشی از محتوای باینری آن:

The screenshot shows the Immunity Debugger interface. The top menu bar includes New file, Open file, Save as, Undo, Redo, Tools, Settings, and Help. A PATREON and PayPal button are also present. The main window has three panes: Registers, Stack, and Dump. The Registers pane shows CPU registers with values like EIP=0x401014, ECX=0x0, and ESP=0x00000000. The Stack pane shows memory starting at address 0x00000000 with various data types. The Dump pane shows the raw binary data of the file sample\_640x426.tif. The Dump pane content starts with the IHDR header:

```
éPNG.....IHDR
00000000 89 50 4E 47 0D 0A 1A 0A 00 00 00 00 49 48 44 52
00000010 00 00 04 B0 00 00 03 20 08 06 00 00 00 33 E0 9B
00000020 02 00 00 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61
00000030 05 00 00 00 20 63 48 52 4D 00 00 7A 26 00 00 80
00000040 84 00 00 FA 00 00 00 80 E8 00 00 75 30 00 00 EA
00000050 60 00 00 3A 98 00 00 17 70 9C BA 51 3C 00 00 00
00000060 06 62 4B 47 44 00 FF 00 FF 00 FF A0 BD A7 93 00
00000070 00 00 07 74 49 4D 45 07 E4 0B 0B 08 1A 32 97 39
00000080 BF C7 00 00 80 00 49 44 41 54 78 DA EC FD 5B AC
00000090 25 D9 96 1D 86 8D 39 D7 5A 11 B1 63 EF F3 C8 93
000000A0 27 B3 B2 B2 B2 B2 F2 D6 E3 3E AA BB 6F 3F 45
000000B0 DA 12 D5 20 69 CA 34 2D A8 45 91 02 61 8B 80 06
000000C0 5A 90 CC 0F 53 80 6D 01 06 01 EB C3 9F FE 30 EC
000000D0 0F 01 FE 30 4C 8B 94 21 52 20 4D 0A 2D 4B 02 0D
000000E0 11 A2 A5 A6 E4 66 93 EC 66 B3 BB EF ED 7B 6F BD
000000F0 F3 66 E5 F3 3C F7 23 62 AD 39 FD 31 D7 8A 88 7D
00000100 32 AB BB 79 E5 76 F5 ED DA B3 70 EA E4 D9 8F 78
00000110 47 AC 35 C7 1C 63 4C AA EB 1A BB D8 C5 2E 76 F1
00000120 45 84 82 01 12 B0 C6 F2 17 A0 04 41 95 FF 22 90
00000130 23 C4 D8 81 1D A0 AC 10 11 A4 A4 20 66 68 54 30
00000140 08 37 6F DC C0 1F FC DA 57 F1 CF DD B9 8B AF DD
00000150 3C 42 3C 7D 8C 0F BE FD 6D 7C F7 37 BE F7 EF 9C
00000160 9D 5C FC 45 ED E2 BE 12 63 C3 8A C8 02 B0 42 45
00000170 40 0A 38 01 58 3D 58 01 07 8F 00 46 83 80 00 C6
00000180 CC D5 F0 4A 60 65 30 08 0C 0F 66 86 43 00 20 70
```

در خط اول محتوای باینری:

éPNG.....IHDR



بخشی از محتوای باینری آن:

Screenshot of the Beta.Exif.II\* software interface showing the binary dump of the image file.

The interface includes a menu bar with New file, Open file, Save as, Undo, Redo, Tools, Settings, and Help. There are also PATREON and PayPal links.

The main window displays the binary dump of the file "background\_beautiful...". The dump shows hex values followed by ASCII characters and some descriptive text.

```

00000000 FF D8 FF E1 00 DC 45 78 69 66 00 00 49 49 2A 00 + β.Exif..II*
00000010 08 00 00 00 04 00 0F 01 02 00 06 00 00 00 3E 00 .....>.
00000020 00 00 10 01 02 00 15 00 00 00 44 00 00 00 12 01 .....D.....
00000030 03 00 01 00 00 00 01 00 00 00 69 87 04 00 01 00 .....iç.....
00000040 00 00 5A 00 00 00 00 00 00 00 43 61 6E 6F 6E 00 ..Z.....Canon.
00000050 43 61 6E 6F 6E 20 45 4F 53 20 35 44 20 4D 61 72 Canon EOS 5D Mar
00000060 6B 20 49 49 00 00 06 00 9A 82 05 00 01 00 00 00 .....k II...Üé.....
00000070 A8 00 00 00 9D 82 05 00 01 00 00 00 B0 00 00 00 .....{...¥é...}...É..
00000080 27 88 03 00 01 00 00 00 7D 00 00 00 03 90 02 00 .....é.....}...É..
00000090 14 00 00 00 B8 00 00 00 09 92 03 00 01 00 00 00 .....ç.....Æ...
000000A0 10 00 00 00 0A 92 05 00 01 00 00 00 CC 00 00 00 .....E.....¶...
000000B0 00 00 00 00 01 00 00 00 FA 00 00 00 08 00 00 00 .....'.
000000C0 01 00 00 00 32 30 31 30 3A 30 34 3A 32 36 20 31 .....2010:04:26 1
000000D0 32 3A 30 33 3A 31 33 00 0E 01 00 00 01 00 00 00 .....2:03:13.....
000000E0 FF ED 00 24 50 68 6F 74 6F 73 68 6F 70 20 33 2E φ.$Photoshop 3.
000000F0 30 00 38 42 49 4D 04 04 00 00 00 00 00 00 07 1C 02 0.8BIM.....
00000100 00 00 02 00 04 00 FF DB 00 43 00 03 02 02 02 .....■.C.....
00000110 02 03 02 02 02 03 03 03 03 04 06 04 04 04 04 .....'.
00000120 08 06 06 05 06 09 08 0A 0A 09 08 09 09 0A 0C 0F .....'.
00000130 0C 0A 0B 0E 0B 09 09 0D 11 0D 0E 0F 10 10 11 10 .....'.
00000140 0A 0C 12 13 12 10 13 0F 10 10 10 FF DB 00 43 01 .....■.C.....
00000150 03 03 03 04 03 04 08 04 04 08 10 0B 09 0B 10 10 .....'.
00000160 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....'.
00000170 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....'.
00000180 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....'.

```

در خط اول محتوای باینری:

# β.Exif..II\*

فایلها توی هارددیسک به شکل ۰ و ۱ نوشته میشن. هر ۸تا صفر یا یک، میشه یک بایت. تا اینجاش چیز جدیدی نداره.

حالا متنها چطوری ذخیره میشن؟

متن هم اگه ascii باشه، کاراکترها تبدیل میشن به کد اسکی ۳ مثلا از ۴۸ تا ۵۷ میشه اعداد ۰ تا ۹ و از ۶۵ تا ۹۰ میشه Z تا A (حروف بزرگ) و ...

کامپیوتر میاد کاراکترهای ما رو تبدیل میکنه به این کدهای اسکی (که عدد هستن بر مبنای ۱۰) و بعد این اعداد رو تبدیل میکنه به باینری (مبنای ۲) و ذخیره میکنه روی هارد. (دقیقا سیستم اینطوری نیست ولی میشه اینطوری تصویرش کرد)

موقع خوندن هم اگه فکر کنه این یه فایل متنی هست، میاد به قسمتهای ۸بايتی تقسیمش میکنه و فکر میکنه اینا کد اسکی هستن و میخواهد کاراکتر متناسب باهشون رو نشون بده.

وقتی شما یه فایل متنی رو باز میکنید اینکار رو انجام میده. وقتی شما یه فایل غیر متنی رو با text editor باز میکنید هم سعی میکنه همینکار رو انجام بده.

حالا اگه اون فایل غیر متنی، عکس باشه، یه text editor معمولی نمیتونه (بلد نیست) بخونتش و به روش خودش بازش میکنه. حالا اگه همون عکس رو بدیم به یه نرمافزار مثل SXIV، یه مدل دیگه بازش میکنه و میتوانه دیتاهای پیکسلهای RGB رو بخونه و نمایش بده.

برای اینکه واقعا ۰ و ۱ های داخل فایلهای باینری رو ببینیم (و بتونیم یک فایل باینری بسازیم با نوشتن ۰ها و ۱ها)، احتمالا باید بشینیم یه نرمافزار (مثلا یه اسکریپت ساده‌ی پایتون) بنویسیم که فایلهای رو بخونه و به بخش‌های ۸بیتی تقسیم کنه و نمایش بده.

یه نکته‌ی دیگه هم هست. حتی همین متنها هم همشون مثل هم نیستن.

اگه میخوایم یه متن فارسی توی یه فایل ذخیره کنیم باید حتما فرمتش UTF-8 باشه. نمیشه توی فایل ascii متن فارسی (یا ژاپنی و ...) نوشت. تفاوت utf-8 و ascii که توی ۸ بایت utf-8 هر کاراکتر ۲ بایت فضا اشغال میکنه (۱۶ بیت).

اگه یه نرمافزاری فقط برای خوندن و نوشتن ascii آماده باشه، نمیتوانه utf-8 بخونه (درست نمایش نمیده کاراکترها رو)

متادیتا:

متادیتا اطلاعات پایه‌ای و خلاصه‌ای درباره یک فایل دیتا مثل عکس، ویدیو، فایل اجرایی و موارد دیگر می‌باشد. در واقع متادیتا به سوالاتی مثل چه چیزی، کجا، توسط چه کسی، چه زمانی، کدام، چطور و چرا در مورد اطلاعات اصلی پاسخ می‌دهد. همینطور متادیتا میتواند به صورت دستی تولید شود که در این صورت مهم است که از استانداردهای تعريف متادیتا پیروی کند.

چرا متادیتا مهم است؟

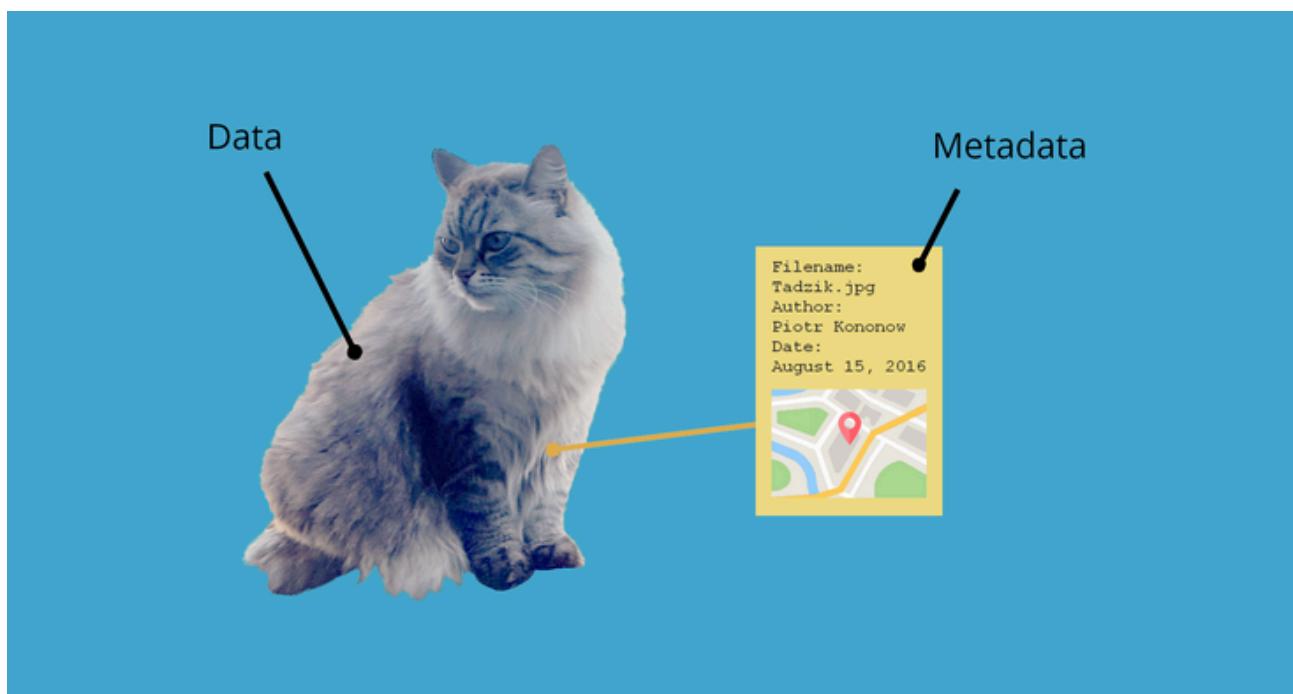
متادیتا می‌تواند از درجه اهمیت متغیری برخوردار باشد. مثلا وقتی که با دوربین دیجیتال خود عکسی را ثبت کرده‌اید و همراه آن موقعیت مکانی عکس نیز ذخیره شده است، این یک متادیتا است ولی اگر به هر طریقی از دست برود زیاد مهم نیست، چون احتمالا از طریق خاطرات مختلفی که آن روز ساخته شده

می‌توانید به راحتی مکان ثبت شدن عکس را به خاطر بیاورید. حال آن عکس را در یکی از وبسایتهايی که خدمات آپلود انجام می‌دهند بارگذاری می‌کنید تا جایش امن باشد و در صورت از دست رفتن حافظه‌ای که عکس در آن ذخیره شده مشکلی برای عکس پیش نیاید. وبسایت آدرسی برای دسترسی به آن عکس به شما می‌دهد که آن آدرس یک متادیتا محسوب می‌شود حال اگر آن آدرس از دست برود چه؟ قطعاً این دو سناریوی متادیتا از یک درجه اهمیت برخوردار نیستند، در مورد اول شما به راحتی به یاد می‌آورید عکس کجا گرفته شده ولی در مورد دوم با از دست رفتن آدرس دیگر امکان دسترسی به عکس فراهم نیست!

متادیتا مهم است چرا که در اصل اطلاعات اصلی را برای ما قابل استفاده می‌کند. ما برای استفاده از یک فایل نیاز به دانستن فرمت آن فایل داریم. متادیتا کاربردهای بسیار گسترده‌ای در زمینه‌های مختلف دارد.

### سابقه استفاده از متادیتا در تشخیص اصالت عکس‌ها

وقتی عکسی را با دوربین دیجیتال خود ثبت می‌کنید در کنار فایل اصلی (فایل عکس گرفته شده) یک فایل متادیتا نیز ذخیره می‌شود که شامل اطلاعات متفاوتی مانند تاریخ و ساعت ثبت عکس، موقعیت مکانی، فرمت عکس و اطلاعاتی درباره کیفیت عکس و پیکسل‌ها است. البته لازم به ذکر است که همانطور که بیان شد متادیتا می‌تواند به صورت دستی نیز تعریف شود. مثلاً می‌توان اسم شخص ثبت کننده عکس را نیز در فایل متادیتا ذکر کرد. سابقاً استفاده از برنامه‌های ویرایش عکس منجر به از بین رفتن این فایل متادیتا می‌شدند و بررسی اینکه عکس فایل متادیتا قابل قبولی دارد یا خیر کمکی بود به تشخیص اصالت عکس‌ها.

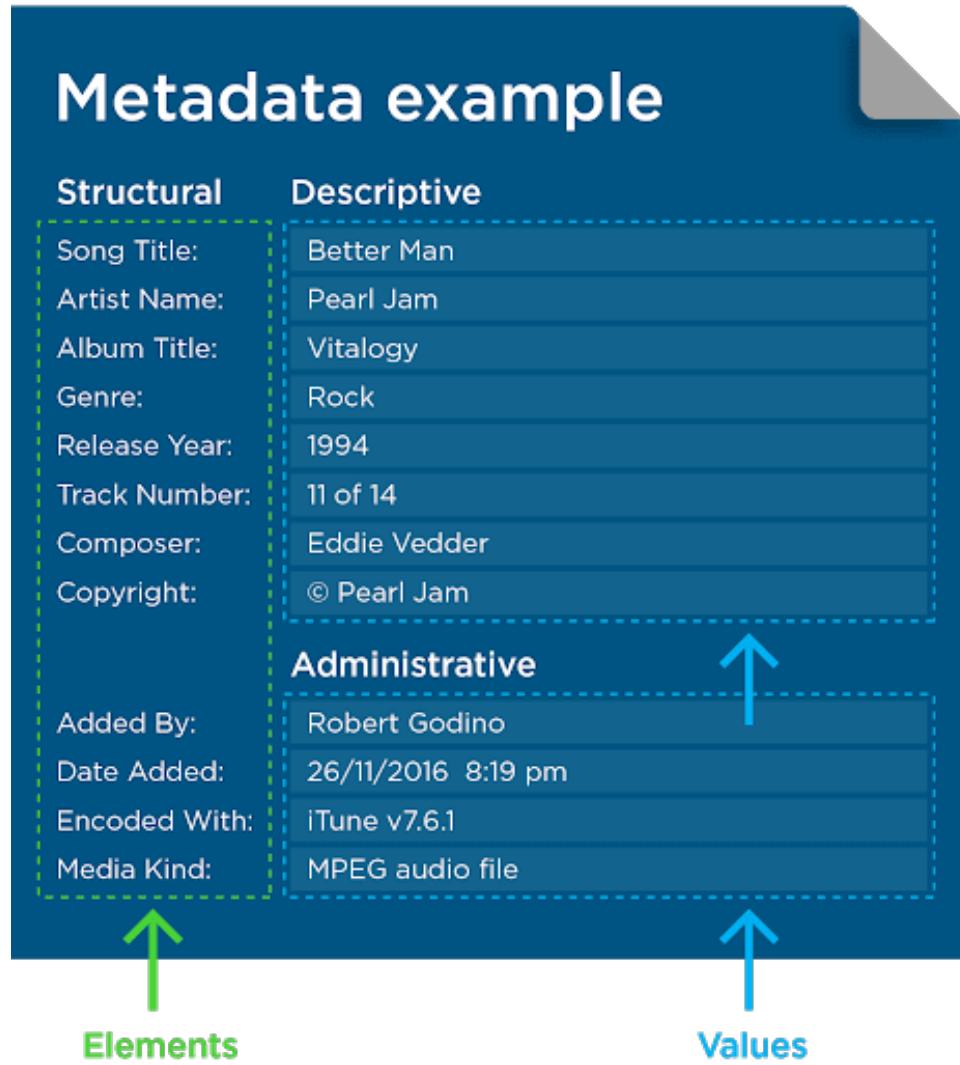


### انواع اصلی متادیتا

متادیتا دارای انواع اصلی متفاوتی است که به طور واضح نمی‌توان آن‌ها را از هم جدا کرد ولی به طور عمومی به چند دسته تقسیم می‌شوند که هر دسته تمکن‌بیشتری بر یک حوزه استفاده دارد.

- **توصیفی:** به عنوان مثال ویژگی‌هایی از قبیل عنوان، موضوع، ژانر، نویسنده، تاریخ و موارد دیگر.
- **حقوقی:** برای مثال ویژگی‌هایی مثل حق کپیرایت، مالک اثر، شرایط مجرور و موارد اینچنینی.

- تکنیکی: ویژگی‌هایی مثل نوع، حجم، تاریخ و زمان، یا ویژگی‌های مربوط به نوع فشردهسازی و مواردی از این دست.
- نگهداری: دارای اطلاعاتی برای کمک به پیدا کردن عنصر مورد نظر در یک دنباله یا جدول
- زبان‌های نشانه‌گذاری: این دست زبان‌ها از قبیل HTML نیز از متادیتا بهره می‌برند که می‌توانند شامل مواردی مثل تیتر، نام، تاریخ، لیست، پاراگراف و غیره باشد.



متا دیتا عکس ها:

:tiff عکس

<b>File Name</b>	sample_640x426.tiff
<b>File Size</b>	799 KIB
<b>File Type</b>	TIFF
<b>File Type Extension</b>	.tif
<b>Mime Type</b>	image/tiff
<b>Exif Byte Order</b>	Little-endian (Intel, II)
<b>Image Width</b>	640
<b>Image Height</b>	426
<b>Bits Per Sample</b>	8 8 8
<b>Compression</b>	Uncompressed
<b>Photometric Interpretation</b>	RGB
<b>Fill Order</b>	Normal <input checked="" type="checkbox"/>
<b>Strip Offsets</b>	8
<b>Orientation</b>	Horizontal (normal)
<b>Samples Per Pixel</b>	3
<b>Rows Per Strip</b>	426 <input checked="" type="checkbox"/>
<b>Strip Byte Counts</b>	817920
<b>Planar Configuration</b>	Chunky <input checked="" type="checkbox"/>
<b>Page Number</b>	0 1 <input checked="" type="checkbox"/>
<b>White Point</b>	0.3127000032 0.328999996 <input checked="" type="checkbox"/>
<b>Primary Chromaticities</b>	0.6399999856 0.330000013 0.3000000118 0.6000000237 0.1500000058 0.05999999844 <input checked="" type="checkbox"/>
<b>Image Size</b>	640x426
<b>Megapixels</b>	0.273
<b>Category</b>	image
<b>Raw Header</b>	49 49 2A 00 08 7B 0C 00 B2 82 90 AD 7D 8B AA 7A 8A AA 7A 8A AB 7B 8B A7 77 87 A2 71 84 9F 6E 81 9F 70 82 9F 70 82 9E 6F 83 9B 6C 80 98 69 7D 96 67 7B 94 64 7A 94 64 7A 91 63 7D 92 64 7E 88 5C 75 82 58 70 8B 61 79 81 5B 72 8B 65 7C C3 A0 B6 85 63 7B 7C 5A 72 79 57 6F 7D 5B 73 7E 5B 73 78 55 6D 7A 54 6D 80 5A 73 7B 57 71 79 55 6F 78 54 6E 78 54 6E 78 56 6F 77 55 6E 75 53 6C 72 50 69

<b>File Name</b>	transparent-rose-5fb25260c732c1.7530012016055220168159.png
<b>File Size</b>	704 KiB
<b>File Type</b>	PNG
<b>File Type Extension</b>	png
<b>Mime Type</b>	image/png
<b>Image Width</b>	1200
<b>Image Height</b>	800
<b>Bit Depth</b>	8
<b>Color Type</b>	RGB with Alpha
<b>Compression</b>	Deflate/Inflate
<b>Filter</b>	Adaptive
<b>Interlace</b>	Noninterlaced
<b>Gamma</b>	2.2
<b>White Point X</b>	0.3127
<b>White Point Y</b>	0.329
<b>Red X</b>	0.64
<b>Red Y</b>	0.33
<b>Green X</b>	0.3
<b>Green Y</b>	0.6
<b>Blue X</b>	0.15
<b>Blue Y</b>	0.06
<b>Background Color</b>	255 255 255
<b>Modify Date</b>	2020:11:11 08:26:50
<b>Warning</b>	[minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers)
<b>Datecreate</b>	2020-11-11T16:26:50+08:00
<b>Datemodify</b>	2020-11-11T16:26:50+08:00
<b>Image Size</b>	1200x800
<b>Megapixels</b>	0.96
<b>Category</b>	image
<b>Raw Header</b>	89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 04 B0 00 00 03 20 08 06 00 00 00 33 E0 9B 02 00 00 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 05 00 00 00 20 63 48 52 4D 00 00 7A 26 00 00 80 84 00 00 FA 00 00 00 80 E8 00 00 75 30 00 00 EA 60 00 00 3A 98 00 00 17 70 9C BA 51 3C 00 00 00 06 62 4B 47 44 00 FF 00 FF 00 FF A0 BD A7 93 00 00 00 07 74 49 4D 45 07 E4 0B 0B 08 1A 32 97 39

File Name	1.jpg
File Size	1710 KiB
File Type	JPEG
File Type Extension	jpg
Mime Type	image/jpeg
Exif Byte Order	Little-endian (Intel, II)
Make	Canon <input checked="" type="checkbox"/>
Model	Canon EOS 5D Mark II <input checked="" type="checkbox"/>
Orientation	Horizontal (normal) <input checked="" type="checkbox"/>
Exposure Time	1/250 <input checked="" type="checkbox"/>
F Number	8
Iso	125
Date Time Original	2010:04:26 12:03:13 <input checked="" type="checkbox"/>
Flash	Off, Did not fire
Focal Length	270.0 mm
Current Iptc Digest	5f0bf878616c5f867120723ac4c7b112
Application Record Version	4
Image Width	3509
Image Height	2339
Encoding Process	Baseline DCT, Huffman coding
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Aperture	8
Image Size	3509x2339
Megapixels	8.2
Shutter Speed	1/250
Focal Length35Efl	270.0 mm
Light Value	13.6
Category	image
Raw Header	FF D8 FF E1 00 DC 45 78 69 66 00 00 49 49 2A 00 08 00 00 00 04 00 0F 01 02 00 06 00 00 00 3E 00 00 00 10 01 02 00 15 00 00 00 44 00 00 00 12 01 03 00 01 00 00 00 01 00 00 00 69 87 04 00 01 00 00 00 5A 00 00 00 00 00 00 00 00 43 61 6E 6F 6E 00 43 61 6E 6F 6E 20 45 4F 53 20 35 44 20 4D 61 72 6B 20 49 49 00 00 06 00 9A 82 05 00 01 00 00 00 A8 00 00 00 9D 82 05 00 01 00 00 00 B0 00 00 00