ИТМО

Лекция №2.
Форензика. Кодировки

Система счисления

Механизм записи чисел при помощи цифр

Десятичная
Base10

Цифры = {0,1,2,3,4,5,6,7,8,9}
Примеры чисел: 998, 1009

Восьмеричная
Base8

Цифры = {0,1,2,3,4,5,6,7}
Пример чисел: 777, 107, 1337

Пример исп.: определение прав в Linux

Двоичная
Base2

Цифры = {0,1}
Примеры чисел: 10010, 10, 1111

Пример исп.: представление битовых последовательностей

Шестнадцатеричная
Base16

Цифры = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
Примеры чисел: FE3, 4ABCD1, EE00

Пример исп.: представление байтовых последовательностей

Алгоритм перевода из одной системы счисления в другую

Перевод из одной произвольной системы n в другую произвольную систему счисления m возможен через десятичную систему счисления.

(1) Перевод из системы счисления **n** в десятичную

k k-1 k-2 ... 2 1 0 (-1 -2 -3 и т.д. если есть цифры после запятой)

$$A\,B\,C\,\ldots X\,Y\,Z_{\mathbf{n}} =$$

$$= A \cdot \mathbf{n}^{k} + B \cdot \mathbf{n}^{k-1} + C \cdot \mathbf{n}^{k-2} + \ldots + X \cdot \mathbf{n}^{2} + Y \cdot \mathbf{n}^{1} + Z \cdot \mathbf{n}^{0}$$

Пример (n = 5):

3 2 1 0 -1

$$3402{,}3_{\mathbf{5}} =$$

$$= 3 \cdot \mathbf{5}^{3} + 4 \cdot \mathbf{5}^{2} + 0 \cdot \mathbf{5}^{1} + 2 \cdot \mathbf{5}^{0} + 3 \cdot \mathbf{5}^{-1} =$$

$$= 477{,}6_{10}$$

(2) Перевод из десятичной системы счисления в m

(2.1) Сначала переводится целая часть.
(2.2) Затем переводится часть после запятой.

Результат обоих операций складывается.

Рассмотрим (2.1) и (2.2) для числа $477{,}6_{10}$

(2.1) Перевод целой части из десятичной системы счисления в **m**

A – исходная целая часть числа

a_i – частные от целочисленного деления на m

b_i – остатки от деления

$A : m = a_1$ (ост. b_1)

$a_1 : m = a_2$ (ост. b_2)

$a_2 : m = a_3$ (ост. b_3)

...

$a_{k-1} : m = a_K$ (ост. b_K) $< m$

$A = \overline{a_K b_K \ldots b_3 b_2 b_1}$

Пример (m = 5):

477 : 5 = 95 (ост. 2)

96 : 5 = 19 (ост. 0)

19 : 5 = 3 (ост. 4)

477 = 3402

(2.2) Перевод части после запятой из десятичной системы счисления в **m**

A – часть исходного числа после запятой

a_i – целые части чисел, полученных от умножения на m

$$0,A \times m = a_1,b_1$$
$$0,b_1 \times m = a_2,b_2$$
$$0,b_2 \times m = a_3,b_3$$
$$\dots$$
$$0,b_{n-1} \times m = a_K,b_n$$

до тех пор, пока b_n не равен нулю, или нас не устраивает нужная точность

$$0,A = 0,\overline{a_K \dots b_3 b_2 b_1}$$

Пример (m = 5):

$$0,6 \times 5 = 3,0$$
$$0,0$$

$$0,6 = 0,3\overline{00}$$

(2) Перевод из десятичной системы счисления в m

(2.1) 3402
(2.2) 0,3

$$477{,}6_{10} = 3402 + 0{,}3 = 3402{,}3_{5}$$

Проблема точности при работе числами с запятой

То, что мы можем представить число с однозначным количеством разрядов в одной системе счисления не означает то, мы однозначно можем представить это число с однозначным количеством разрядов в другой системе счисления, и наоборот. Например:

$0.1_{10} \rightarrow 0.0001100110011001100110..._{2} \rightarrow 0.10000038146972656..._{10}$

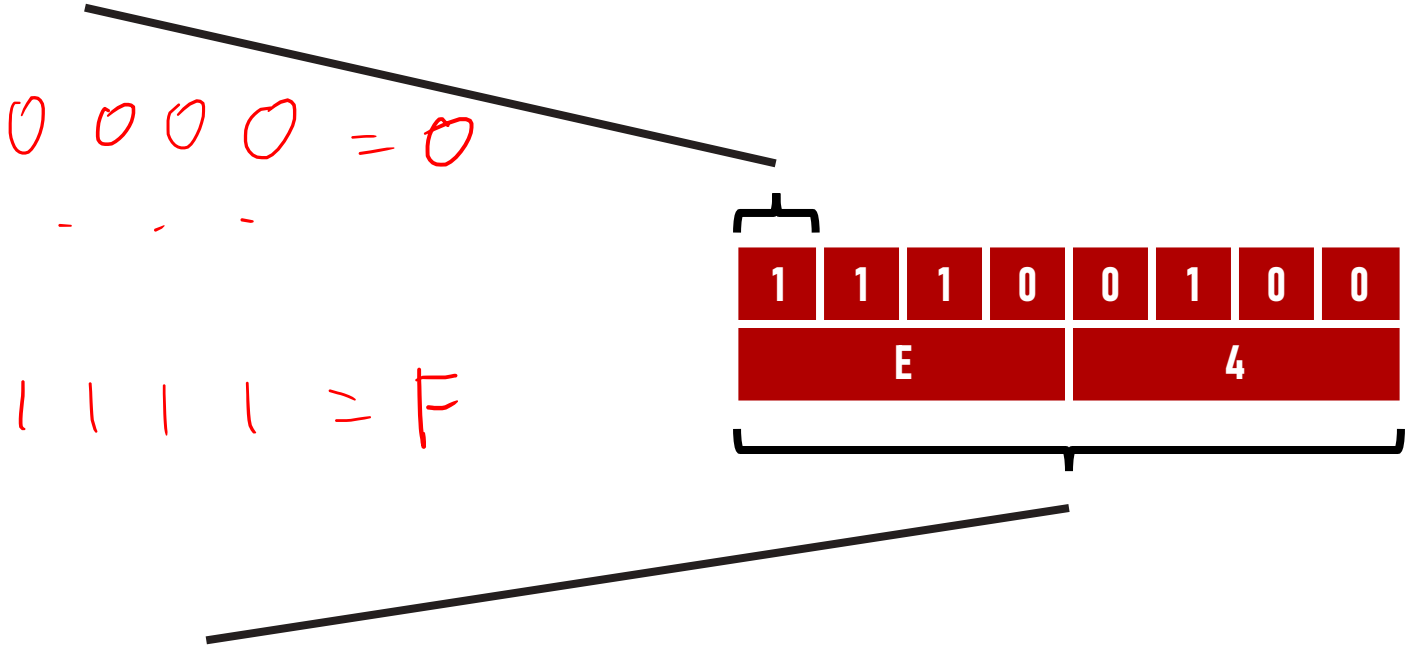
$0.2_{10} \rightarrow 0.0011001100110011001100..._{2} \rightarrow 0.19999980926513672..._{10}$

```
» a = 0.3
← 0.3
» b = 0.1 + 0.2
← 0.30000000000000004
» a == b
← false
```

```
Python 3.10 (64-bit)
>>> a = 0.3
>>> b = 0.1 + 0.2
>>> a == b
False
>>>
```

Биты и байты

Бит – элементарная единица информации, принимающая значения «0» или «1».



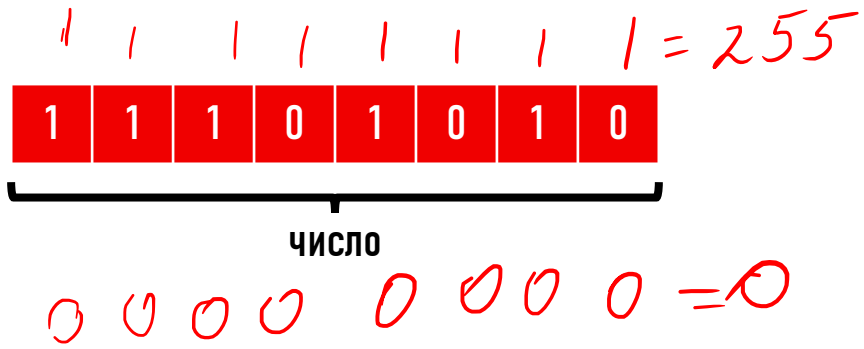
Байт = 8 бит. Байты удобно представлять, как **числа в шестнадцатеричной системе счисления** (которые иногда называют **hex-последовательностями**).

Хранение целых чисел в памяти компьютера

Unsigned

Тип данных, который не содержит информацию о знаке числа

$1110\ 1010_{2} = 234_{10}$



Если число занимает один байт, то в нём могут храниться числа от 0 до 255.

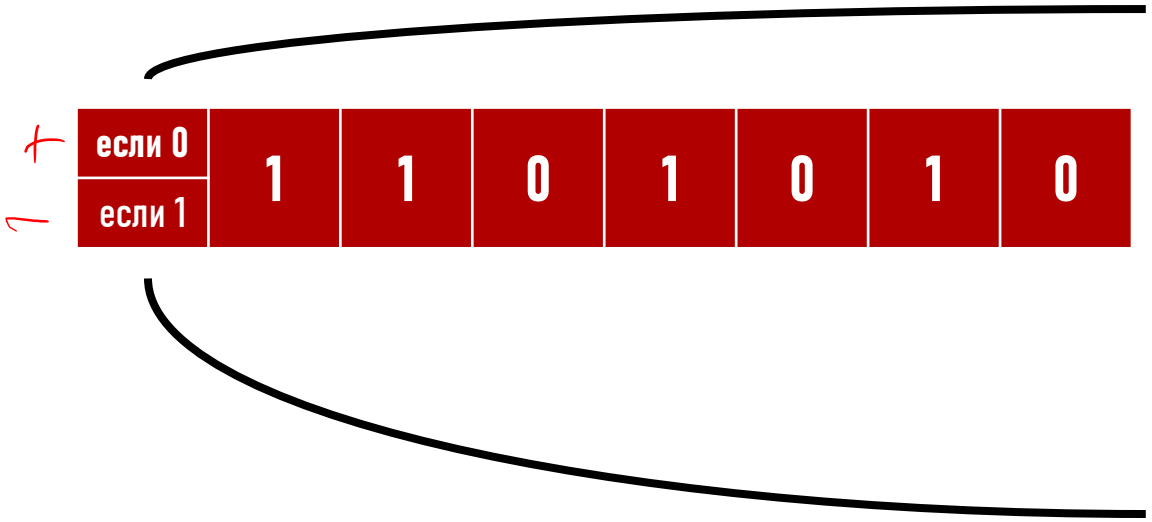
Signed

Тип данных, который содержит информацию о знаке числа

$1110\ 1010_{2*} = -22_{10}$

Если число занимает один байт, то в нём могут храниться числа от -128 до 127.

Как расшифровать signed-формат числа



тогда число в прямом коде, т.е. это просто оставшиеся биты : $1101010_2 = 106_{10}$

тогда число в дополнительном коде, т.е. вычисляется следующим образом:

1) оставшиеся биты инвертируются
   1101010 = 0010101
2) затем складываются с единицей
   0010101 + 1 = 0010110
3) и потом добавляется минус
   $-10110_2 = -22_{10}$

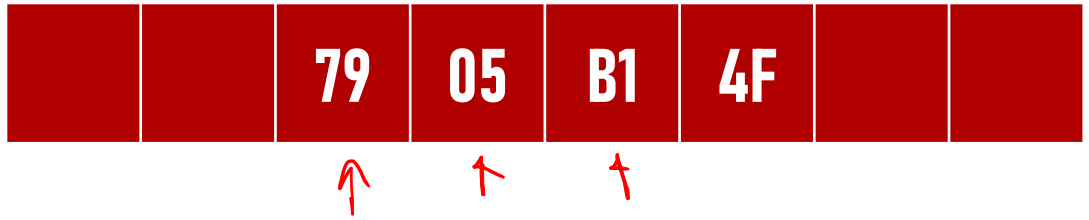
Такой формат упрощает вычисления для процессора

Порядок байт в памяти компьютера

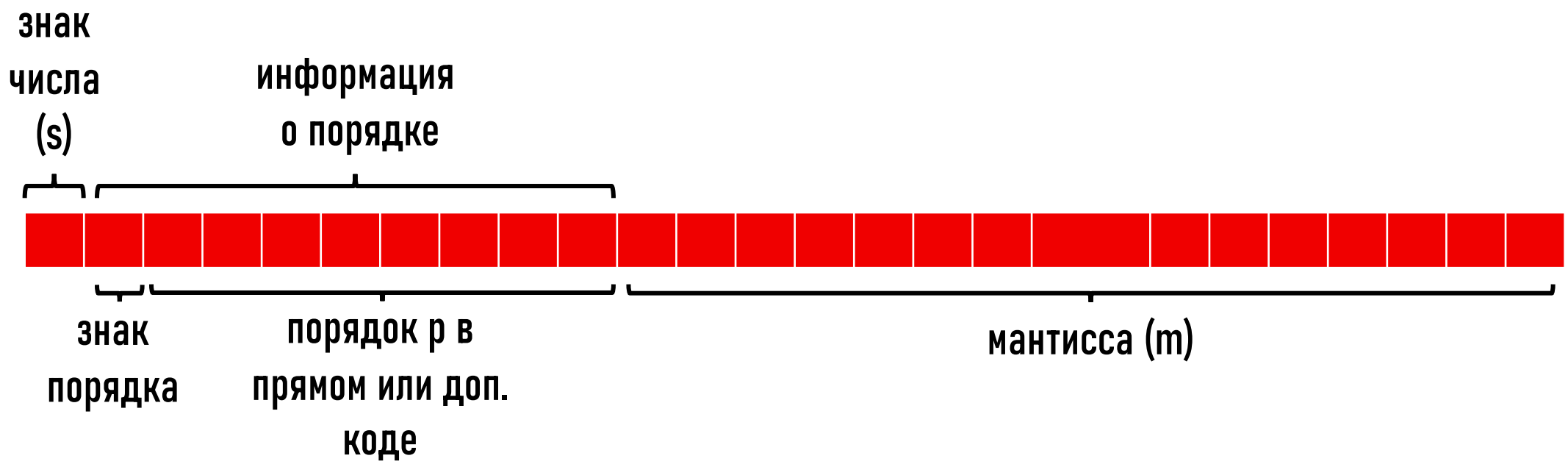
$1\,337\,001\,337_{10} = 4F\ B1\ 05\ 79_{16}$ можно представить в памяти следующим образом.

- Big Endian

- Little Endian



Хранение вещественных чисел в памяти компьютера (число с плавающей запятой, float)



$$число = (-1)^s \times 1.m \times 2^p$$

Системы счисления, являющиеся степенью двойки(Base)

Перевод можно осуществлять не через десятичную систему счисления, а через двоичную путём объединения разрядов, начиная с младших и использования следующей таблицы:

base8	base2	base16	base2	base16	base2
0	000	0	0000	8	1000
1	001	1	0001	9	1001
2	010	2	0010	A	1010
3	011	3	0011	B	1011
4	100	4	0100	C	1100
5	101	5	0101	D	1101
6	110	6	0110	E	1110
7	111	7	0111	F	1111

$1111_2 = F_{16}$

$432_8 = 11A_{16} = 100011010_2$

base8		4	3	2
base2	000	100	011	010
base16	1	1	A	

AB

Base64

Зачем нужен?

По той или иной причине нужно закодировать бинарный формат, например изображение, в текст.

А в чём проблема?

Некоторые байты нельзя представить, как текст в ASCII и UTF
(или можно, но очень в очень странном виде).

А в чём решение?

Использование **шести бит** для кодирования вместо восьми, что исключает невидимые символы из алфавита кодировки.

Сравнение таблиц ASCII и Base64

ASCII

Dec	Hex	Char	Dec
0	0	NUL	
1	1	SOH	
2	2	STX	
3	3	ETX	
4	4	EOT	
5	5	ENQ	
6	6	ACK	

Base64

Символ	Значение				Символ	Значение				Символ
	10	2	8	16		10	2	8	16	
A	0	000000	00	00	Q	16	010000	20	10	g
B	1	000001	01	01	R	17	010001	21	11	h
C	2	000010	02	02	S	18	010010	22	12	i
D	3	000011	03	03	T	19	010011	23	13	j
E	4	000100	04	04	U	20	010100	24	14	k
F	5	000101	05	05	V	21	010101	25	15	l

Например, байт 00 в ASCII в виде символа так просто не представить.

Кодировка ASC(base256)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

255
FF

Base64

А если количества байт не хватает?

Добавить такое количество байт, которого будет хватать, а группы бит из шести, которые полностью состоят из нулей, заменить на «=» при переводе. При обратном переводе важно учитывать эту особенность.

base64(4D 61) = TWE=

Пример использования Base64

Правило CSS с внедрённым фоновым изображением [источник]

```
ul.checklist > li.complete {
        margin-left: 20px;
        background: url(

l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4Ug9C9zwz3gVLMDA/A6P9/AFGGFyjO
XZtQAAAAAElFTkSuQmCC
        ) top left no-repeat;
}
```

На той же логике, что и у Base64, построено множество других кодировок, например Base85, Base58, Base32 и т.п.

Их различие в том, что у них изменённый алфавит или (и) требуемое количество бит для одного символа.

Кодировка

Способ представления текстовых символов в виде битов.

ASCII

1 символ = 1 байт = 8 бит,
т.е. в одном байте может
256 разных символов

UTF

1 символ = 1-4 байта
(совместистимы с ASCII)

Как работают кодировки текста. Откуда появляются «кракозябры». Принципы кодирования. Обобщение и детальный разбор. – Хабр:
https://habr.com/ru/post/478636/

Кодировка

Способ представления текстовых символов в виде битов.

Переход между этими кодировками происходит без потерь

UTF-8

1 символ = 1 байт = 8 бит,

UTF-16

1 символ = 2/4 байта,

Используется для хранения строк в приложениях(напр. Java, .NET)

Как работают UTF-8 и UTF-16. В чём их отличие и что выгодней использовать в этой статье.

https://habr.com/ru/articles/312642/

Кодировка URLencode

Для чего:

кодировать или отображать зарезервированные, непечатаемые или не-ASCII символы в URL-адресах в безопасный и надежный формат, который может быть передан через Интернет

Привет Мир → %D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82%20%D0%9C%D0%B8%D1%80

iconv

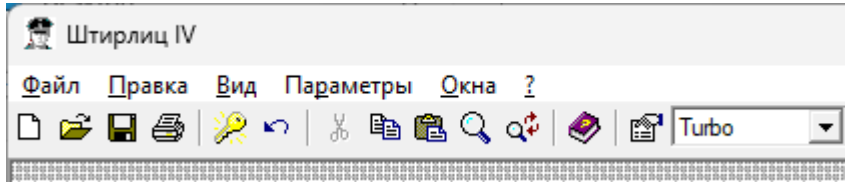
Что это - утилита UNIX для преобразования текста из одной кодировки в другую

Пример

Перекодирование всех файлов в каталоге с их заменой:

```
for i in *; do iconv -f WINDOWS-1251 -t UTF-8 "$i" >tmp; mv tmp "$i"; done
```

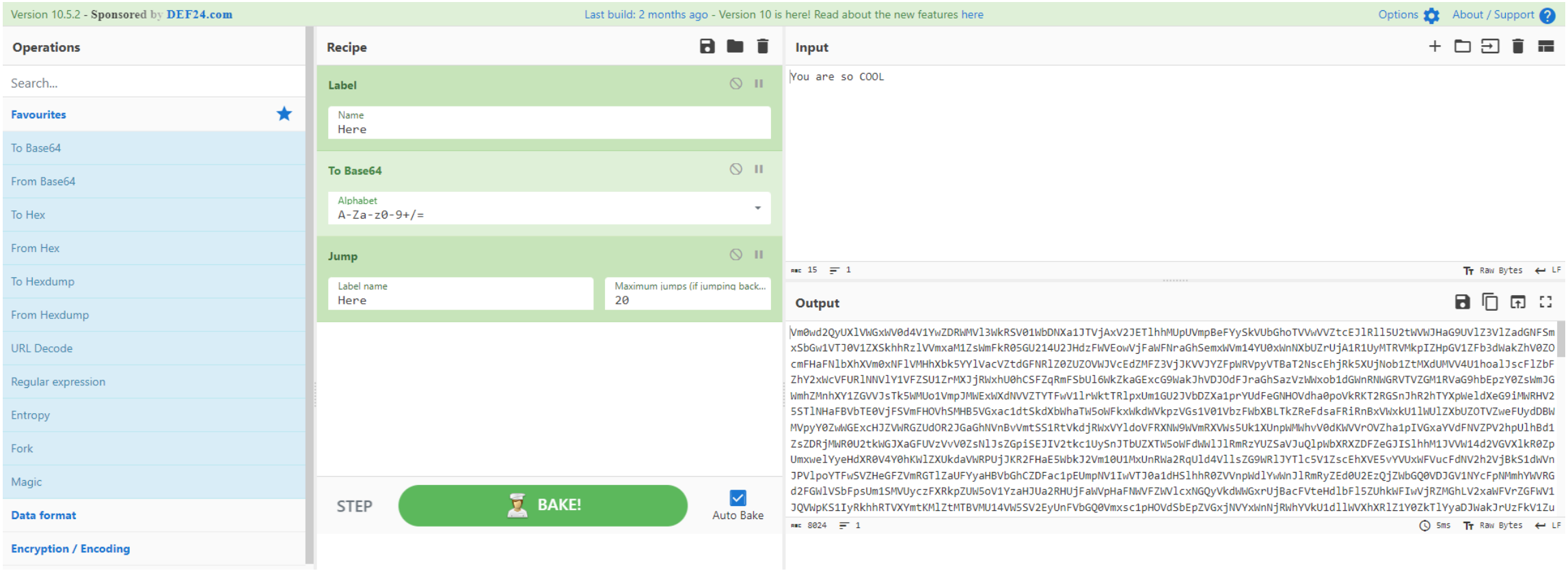
Если не хочется ручками дешифровать(только для кириллицы) - Штирлиц



Или

https://2cyr.com/decode/?lang=ru

CyberChef.



https://cyberchef.org/

https://gchq.github.io/CyberChef/

</>

Демонстрация

Форматы файлов

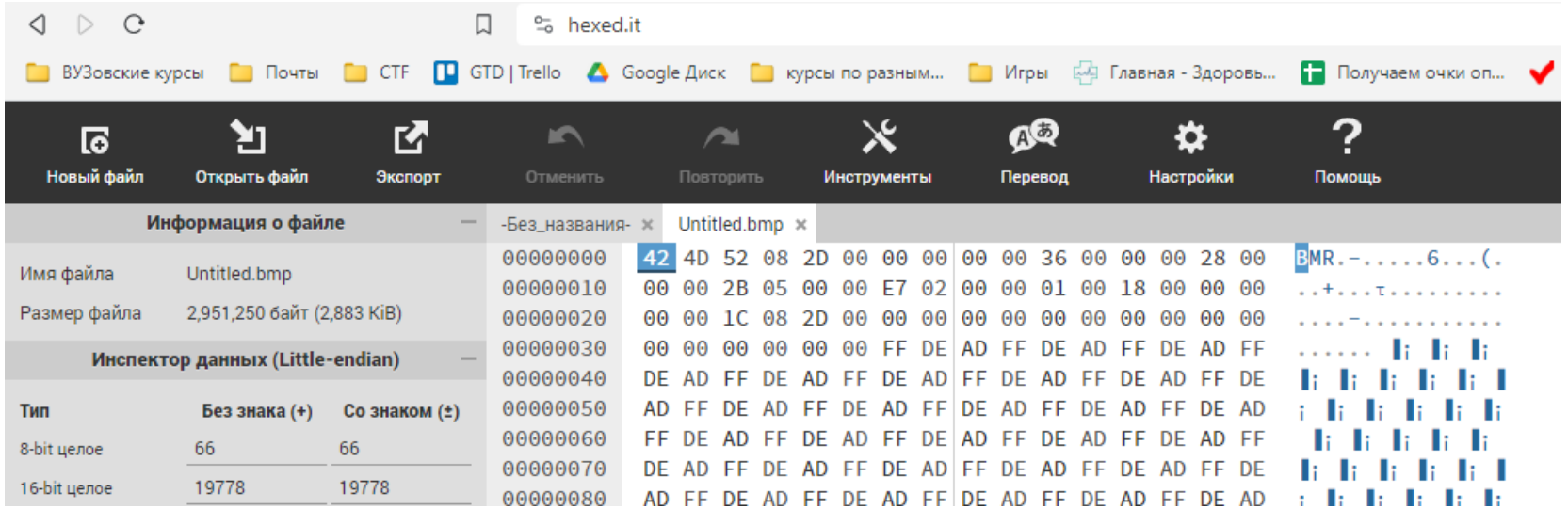
- Archive files (ZIP, TGZ)
- Image file formats (JPG, GIF, BMP, PNG)
- Filesystem images (especially EXT4)
- Packet captures (PCAP, PCAPNG)
- Memory dumps
- PDF
- Video (especially MP4) or Audio (especially WAV, MP3)
- Microsoft's Office formats (RTF, OLE, OOXML)

Как определить типа файла?

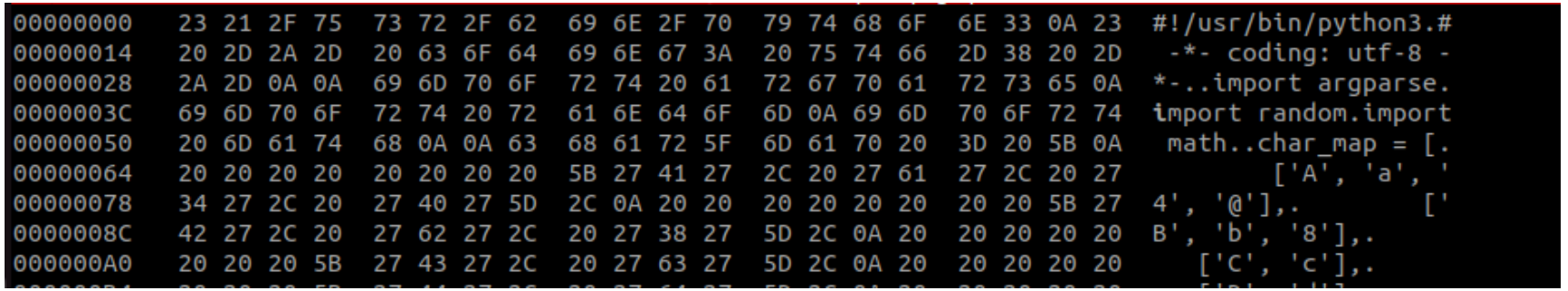
- По расширению имени (н-р: «.png»)
- По магическому числу – особой последовательности байт, характерная для конкретных форматов
- По шебангам – строке, содержащей то, каким интерпретатором должен обрабатываться код

Спецификация – подробное описание структуры файла конкретного формата.

Нех-редактор.

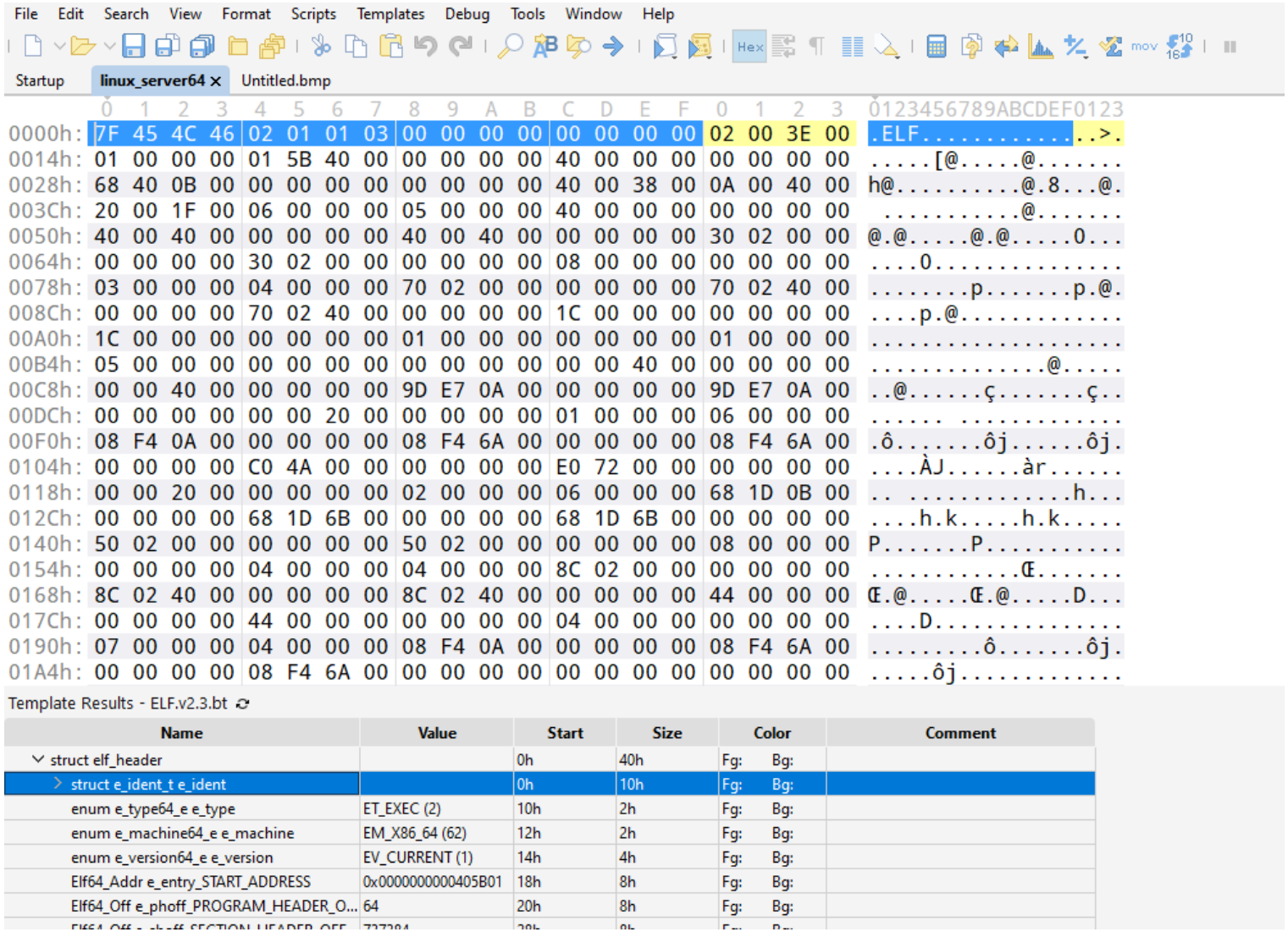


- hexed.it



-hexedit

010 editor - база.



- 010 editor

Hex-редактор.

- Очевидно

```
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 02 72  %PNG........IHDR...r
00 00 03 42 08 02 00 00 00 6D 41 B6 DB 00 00 25 F9 69 54 58  ...B.....mA¶Û..%ùiTX
```

```
25 50 44 46 2D 31 2E 35 0A 25 E4 F0 ED F8 0A 37 20 30 20 6F  %PDF-1.5.%äðíø.7 0 o
62 6A 0A 3C 3C 2F 46 69 6C 74 65 72 2F 46 6C 61 74 65 44 65  bj.<</Filter/FlateDe
```

```
7F 45 4C 46 02 01 01 03 00 00 00 00 00 00 00 00 02 00 3E 00  .ELF..............>.
01 00 00 00 01 5B 40 00 00 00 00 00 40 00 00 00 00 00 00 00  .....[@.....@.......
```

Hex-редактор.

- Очевидно

```
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 02 72  %PNG........IHDR...r
00 00 03 42 08 02 00 00 00 6D 41 B6 DB 00 00 25 F9 69 54 58  ...B.....mA¶Û..%ùiTX
```

- .png file

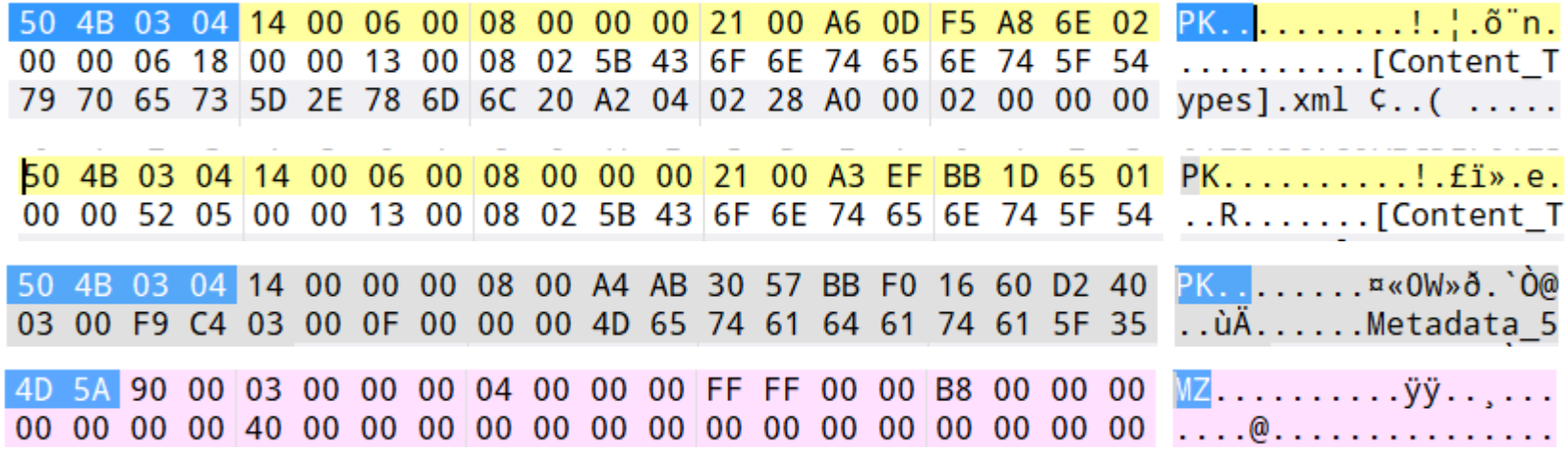
```
25 50 44 46 2D 31 2E 35 0A 25 E4 F0 ED F8 0A 37 20 30 20 6F  %PDF-1.5.%äðíø.7 0 o
62 6A 0A 3C 3C 2F 46 69 6C 74 65 72 2F 46 6C 61 74 65 44 65  bj.<</Filter/FlateDe
```

- .pdf file

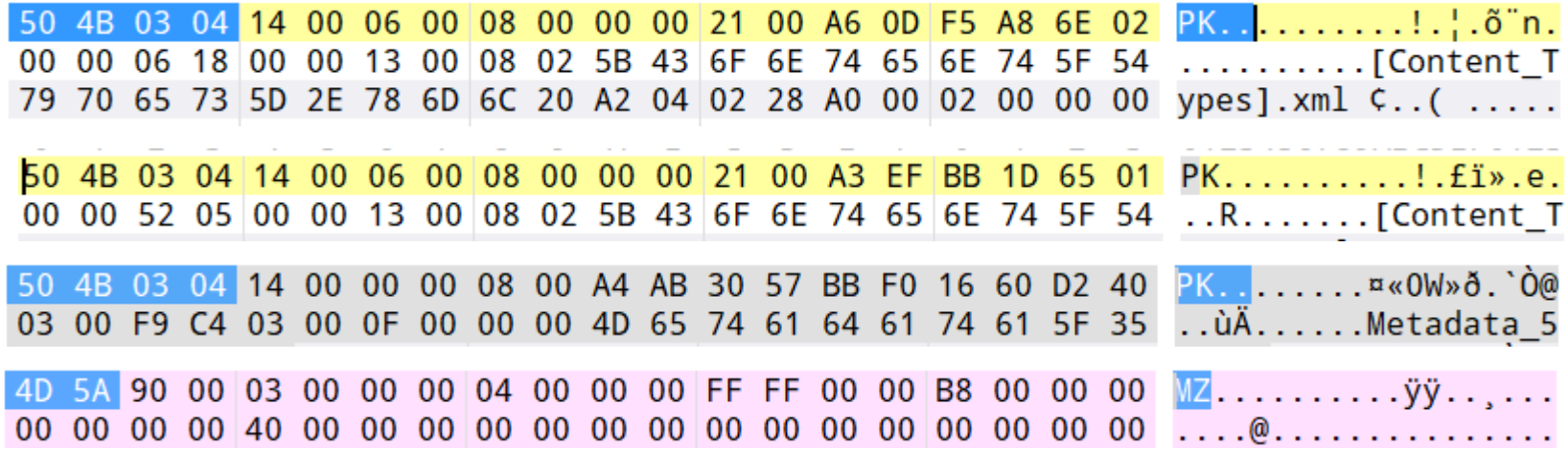
```
7F 45 4C 46 02 01 01 03 00 00 00 00 00 00 00 00 02 00 3E 00  .ELF..............>.
01 00 00 00 01 5B 40 00 00 00 00 00 40 00 00 00 00 00 00 00  .....[@.....@.......
```

- .elf file

- Чуть сложнее



- Чуть сложнее



- .pptx file

- .docx file

- .zip file

- .exe file

- Чуть сложнее

```
50 4B 03 04 14 00 06 00 08 00 00 00 21 00 A6 0D F5 A8 6E 02  PK..........!.¦.õ¨n.
00 00 06 18 00 00 13 00 08 02 5B 43 6F 6E 74 65 6E 74 5F 54  ..........[Content_T
79 70 65 73 5D 2E 78 6D 6C 20 A2 04 02 28 A0 00 02 00 00 00  ypes].xml ¢..( .....
```

- .pptx file

```
50 4B 03 04 14 00 06 00 08 00 00 00 21 00 A3 EF BB 1D 65 01  PK..........!.£ï».e.
00 00 52 05 00 00 13 00 08 02 5B 43 6F 6E 74 65 6E 74 5F 54  ..R.......[Content_T
```

- .docx file

```
50 4B 03 04 14 00 00 00 08 00 A4 AB 30 57 BB F0 16 60 D2 40  PK........¤«0W»ð.`Ò@
03 00 F9 C4 03 00 0F 00 00 00 4D 65 74 61 64 61 74 61 5F 35  ..ùÄ......Metadata_5
```

- .zip file

```
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00  MZ..........ÿÿ..¸...
00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....@...............
```

- .exe file

- Что это вообще?

```
00 00 00 20 66 74 79 70 71 74 20 20 20 05 03 00 71 74 20 20  ... ftypqt   ...qt
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 77 69 64 65  ................wide
```

- .mov audio file

Strings – выводим печатаемые символы

Минимальная длина строки

Сама утилита

Файл, который мы просматриваем

```
amogus@AMOGUS:/mnt/h/Универ/CTFClub/лекция первокурсникам$ strings -n 2 zip_for_strings.zip
PK
VGW
/m
qwertyuiopasdfghjkl.txt+NM.J-
0.2
0LqI
w3
PK
VGW
/m
qwertyuiopasdfghjkl.txt
PK
```

```
PK........ðVGW‡/mê..'
..........qwertyuiop
asdfghjkl.txt+NM.J-©
ö0.2ŽÏ,ŽÏ0LqIÍ‹w3ÌI-
..PK..........ðVGW‡/
mê..........$.......
 .......qwertyuiopas
dfghjkl.txt.. ......
...GÀ×¥óøÙ.GÀ×¥óøÙ.d
„°ŠóøÙ.PK..........i
...R.....
```

Binwalk - идентификации типов файлов

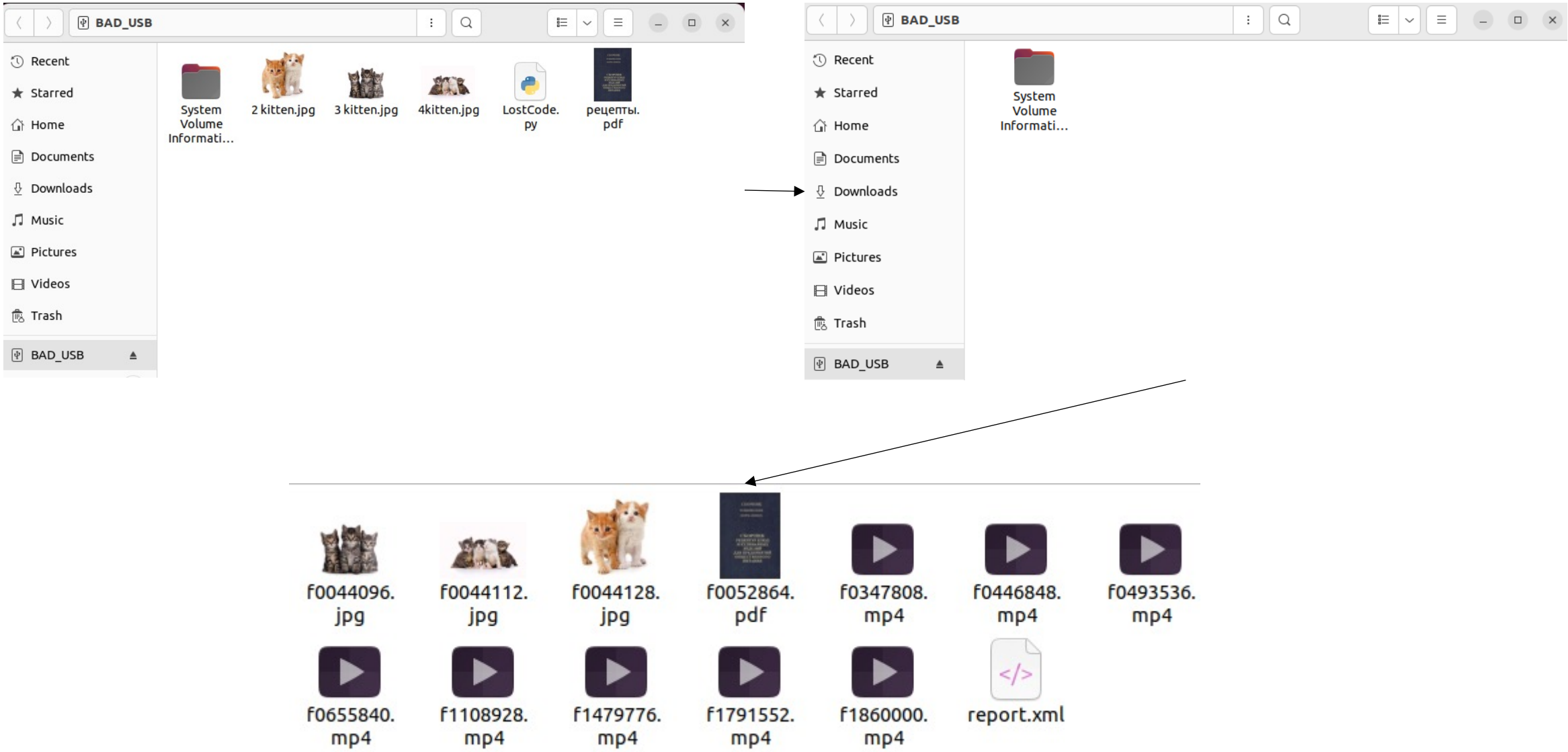
Посмотрим, какие сигнатуры найдутся в exe файле

```
amogus@AMOGUS:/mnt/c/Users/Mefistofele/Desktop/Tasks/forensic_for_itmo$ binwalk executable.3720.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
9178          0x23DA          Copyright string: "CopyrightAttribute"
116288        0x1C640         PNG image, 4800 x 1454, 8-bit/color RGBA, non-interlaced
116416        0x1C6C0         Zlib compressed data, compressed
344098        0x54022         PNG image, 800 x 600, 8-bit colormap, non-interlaced
344511        0x541BF         Zlib compressed data, best compression
420575        0x66ADF         XML document, version: "1.0"
```

Извлечём все сигнатуры, а изображения сохраним как png файлы

```
amogus@AMOGUS:/mnt/c/Users/Mefistofele/Desktop/Tasks/forensic_for_itmo$ binwalk --dd="image:png" executable.3720.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
9178          0x23DA          Copyright string: "CopyrightAttribute"
116288        0x1C640         PNG image, 4800 x 1454, 8-bit/color RGBA, non-interlaced
116416        0x1C6C0         Zlib compressed data, compressed
344098        0x54022         PNG image, 800 x 600, 8-bit colormap, non-interlaced
344511        0x541BF         Zlib compressed data, best compression
420575        0x66ADF         XML document, version: "1.0"
```

Photorec – программа для восстановления данных



</>

Демонстрация

BMP или как оно есть

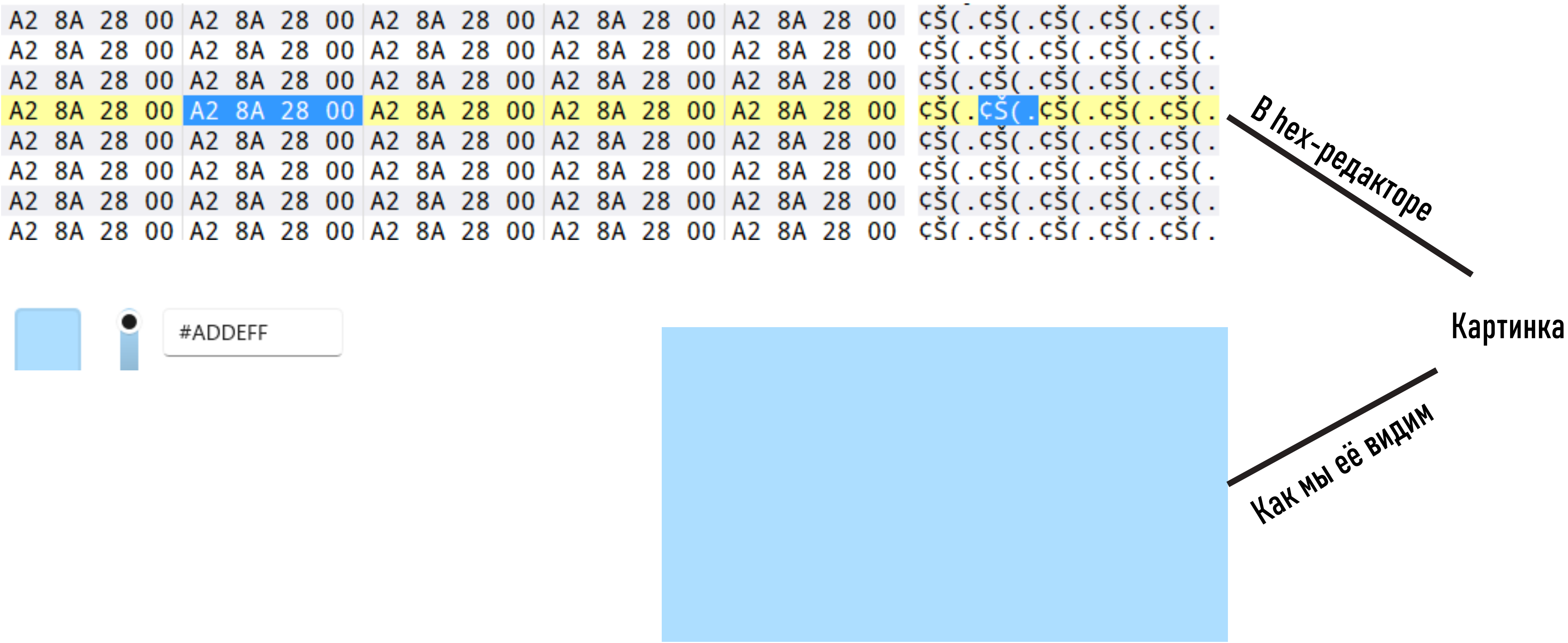
42	4D	52	08	2D	00	00	00	00	00	36	00	00	00	28	00	00	00	2B	05	BMR.-.....6...(...+.
00	00	E7	02	00	00	01	00	18	00	00	00	00	00	1C	08	2D	00	00	00	..ç.............-...
00	00	00	00	00	00	00	00	00	00	00	00	00	00	FF	DE	AD	FF	DE	AD	..............ÿÞ-ÿÞ-
FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ
AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿ
DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	Þ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-
FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ
AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	DE	AD	FF	-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿÞ-ÿ

В hex-редакторе

Картинка

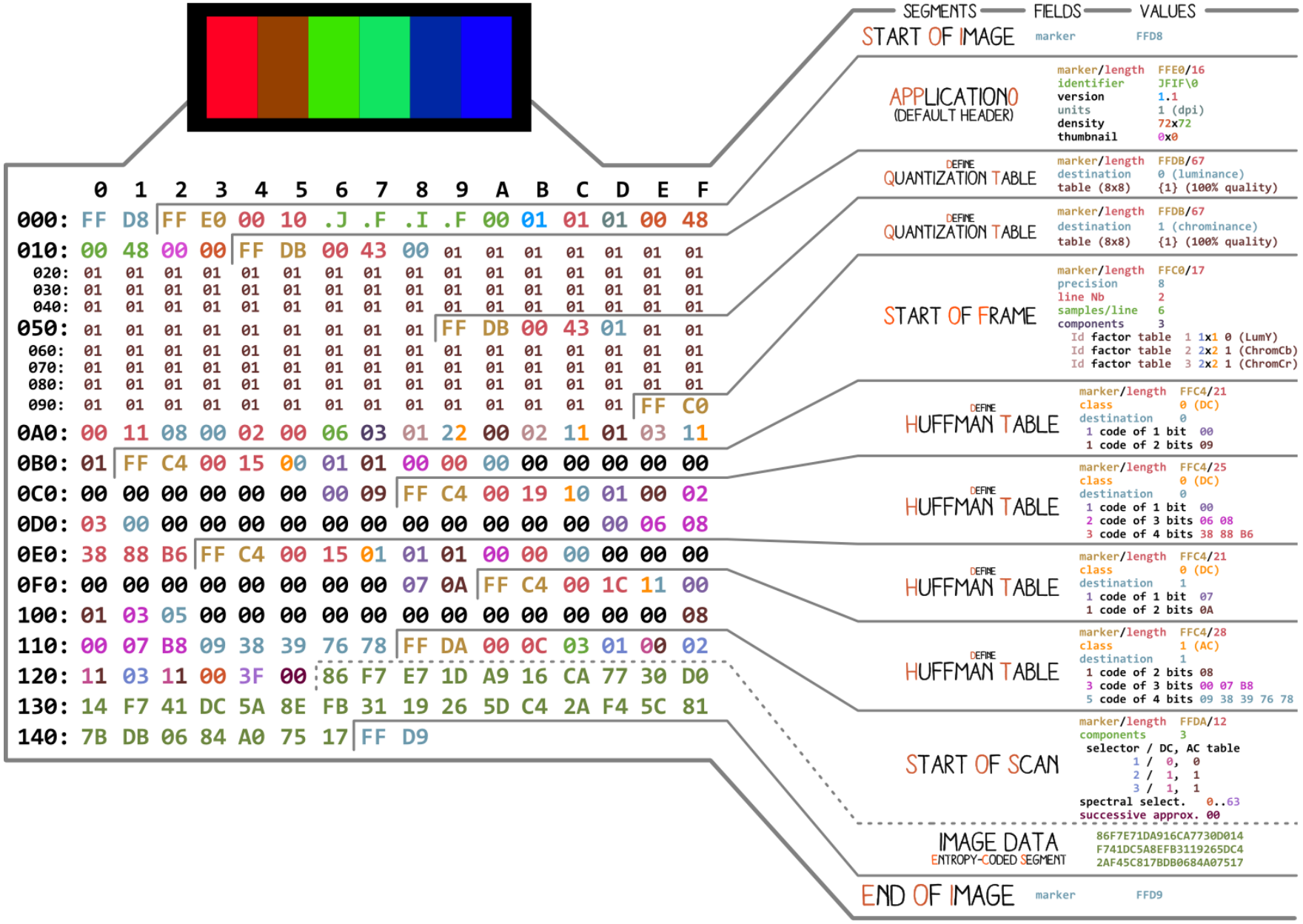
Как мы её видим

JPG



Лучше этой статьи о jpg я ещё не видел: https://parametric.press/issue-01/unraveling-the-jpeg/

JPG


SEGMENTS
FIELDS
VALUES
START OF IMAGE
marker
FFD8
APPLICATION0
(DEFAULT HEADER)
marker/length FFE0/16
identifier JFIF\0
version 1.1
units 1 (dpi)
density 72x72
thumbnail 0x0
DEFINE QUANTIZATION TABLE
marker/length FFDB/67
destination 0 (luminance)
table (8x8) {1} (100% quality)
DEFINE QUANTIZATION TABLE
marker/length FFDB/67
destination 1 (chrominance)
table (8x8) {1} (100% quality)
START OF FRAME
marker/length FFC0/17
precision 8
line Nb 2
samples/line 6
components 3
Id factor table 1 1x1 0 (LumY)
Id factor table 2 2x2 1 (ChromCb)
Id factor table 3 2x2 1 (ChromCr)
DEFINE HUFFMAN TABLE
marker/length FFC4/21
class 0 (DC)
destination 0
1 code of 1 bit 00
1 code of 2 bits 09
DEFINE HUFFMAN TABLE
marker/length FFC4/25
class 0 (DC)
destination 0
1 code of 1 bit 00
2 code of 3 bits 06 08
3 code of 4 bits 38 88 B6
DEFINE HUFFMAN TABLE
marker/length FFC4/21
class 0 (DC)
destination 1
1 code of 1 bit 07
1 code of 2 bits 0A
DEFINE HUFFMAN TABLE
marker/length FFC4/28
class 1 (AC)
destination 1
1 code of 2 bits 08
3 code of 3 bits 00 07 B8
5 code of 4 bits 09 38 39 76 78
START OF SCAN
marker/length FFDA/12
components 3
selector / DC, AC table
1 / 0, 0
2 / 1, 1
3 / 1, 1
spectral select. 0..63
successive approx. 00
IMAGE DATA
ENTROPY-CODED SEGMENT
86F7E71DA916CA7730D014
F741DC5A8EFB3119265DC4
2AF45C817BDB0684A07517
END OF IMAGE
marker
FFD9

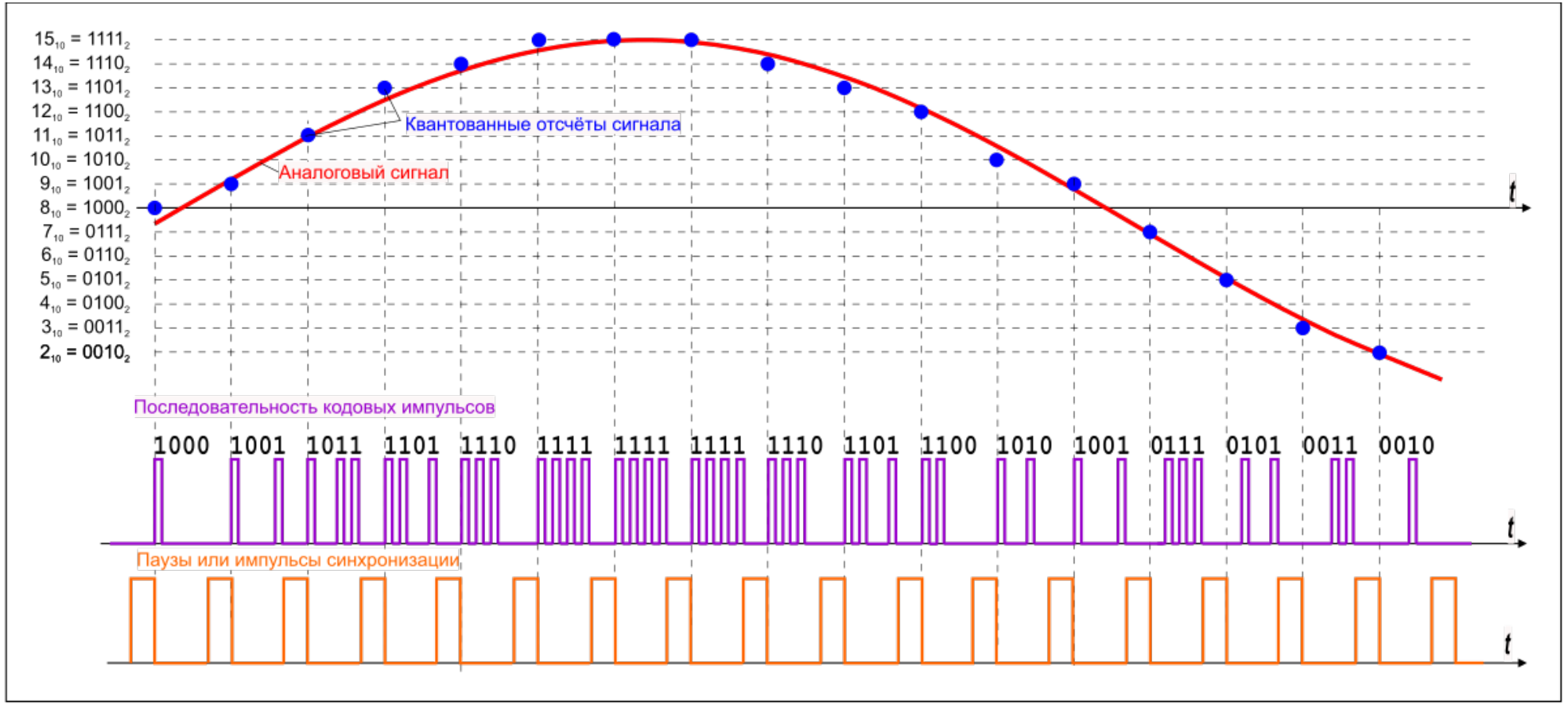
</>

Демонстрация

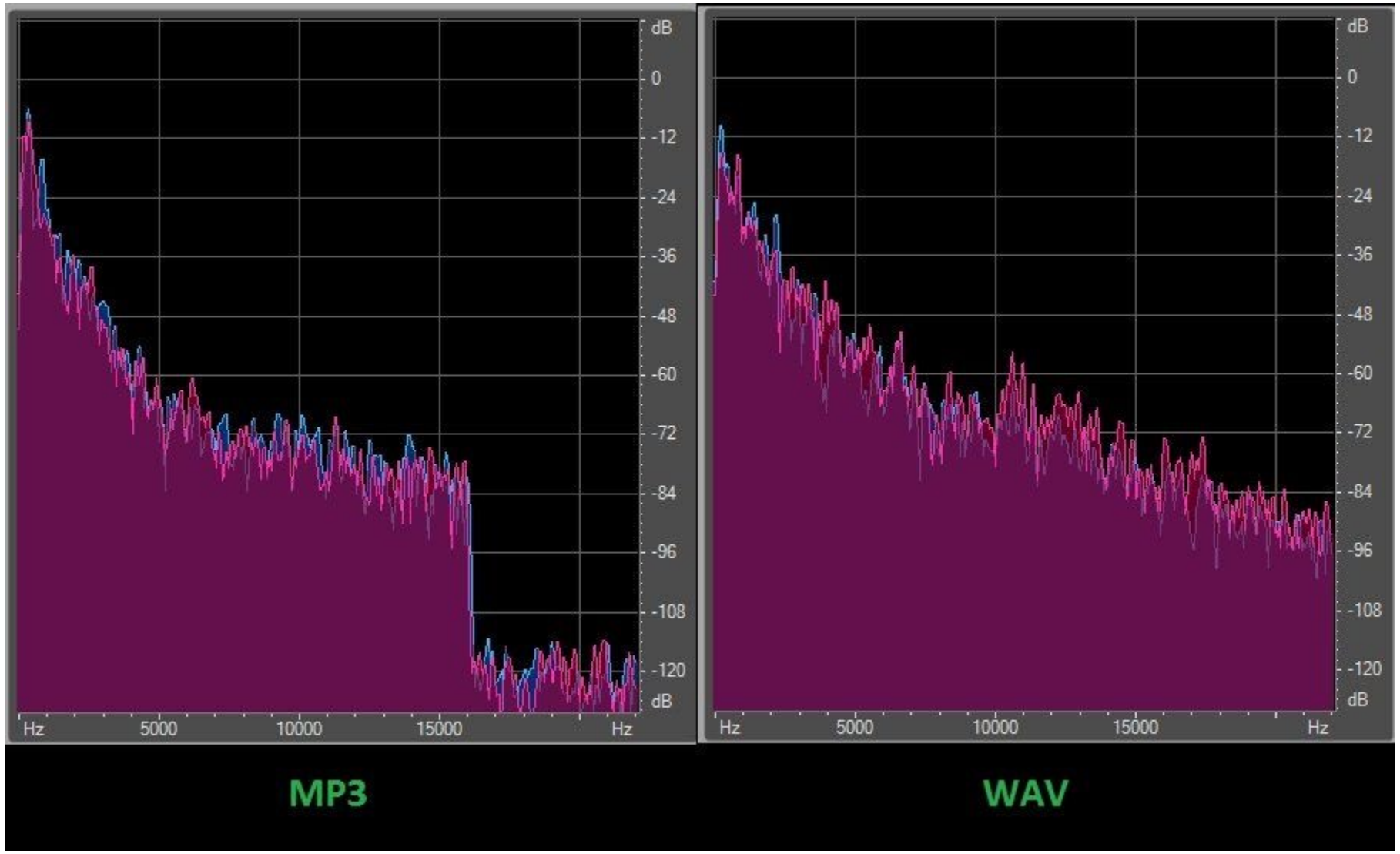
Waveform Audio File Format (WAV)

- **Без сжатия нет потерь**

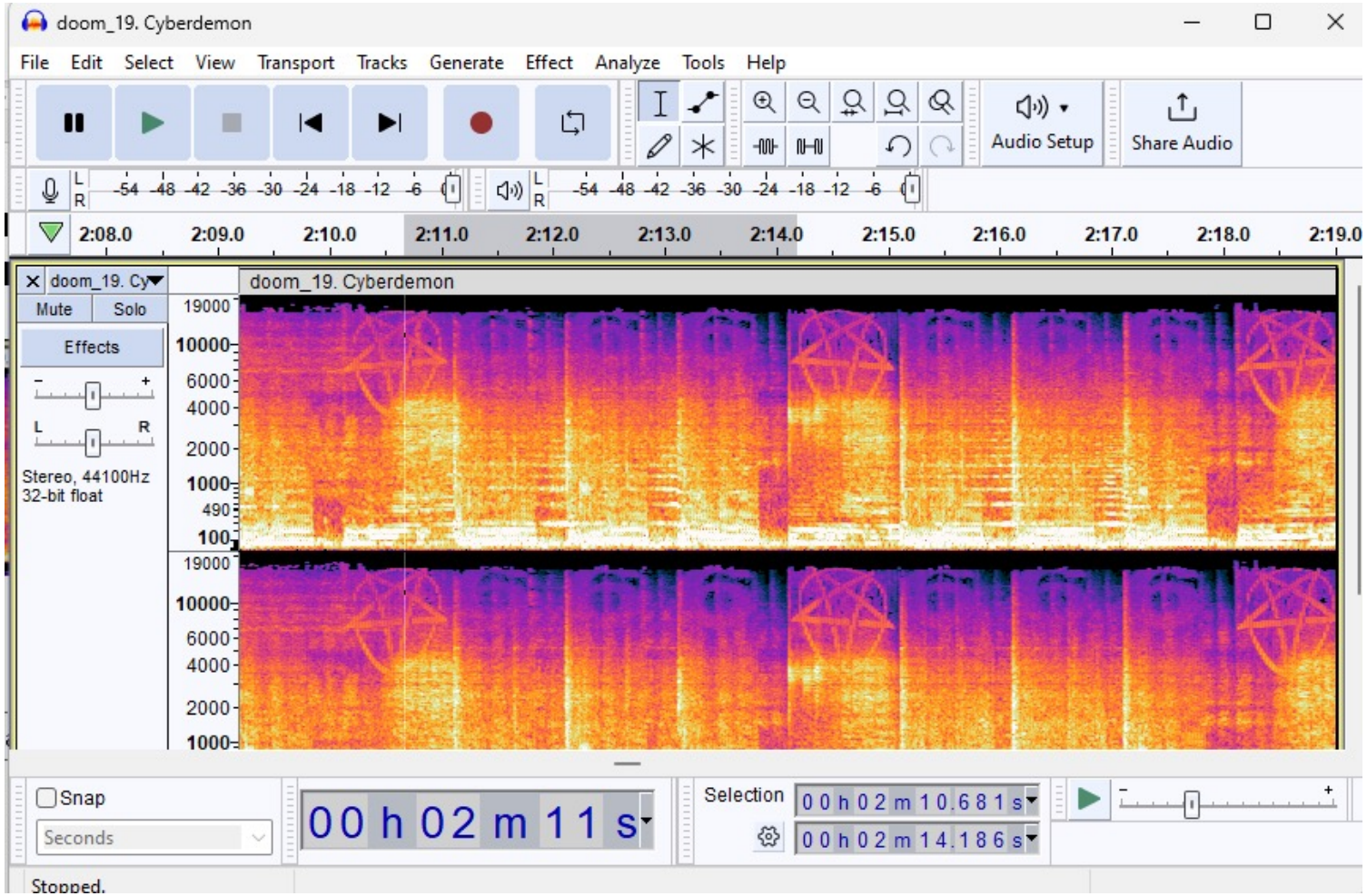
Как оно работает?
1. поток разбивается на малейшие отрезки
2. каждый такой отрезок времени пишется текущее значение аналогового сигнала в двоичной форме



WAV


dB
0
-12
-24
-36
-48
-60
-72
-84
-96
-108
-120
dB
Hz
5000
10000
15000
Hz
MP3
dB
0
-12
-24
-36
-48
-60
-72
-84
-96
-108
-120
dB
Hz
5000
10000
15000
Hz
WAV

Знаменитая спектрограмма в треке из игры Doom 2016



Как я посмотрел спектрограмму? – В программе “Audacity”

</>

Демонстрация

Metadata.

Метаданные файла – это данные о данных (об их составе, содержании, статусе, происхождении, местонахождении, качестве, форматах, объёме, условиях доступа, авторских правах и т. п.

Тулзы для поиска метаданных в файле:

1) exif
2) exfitool
3) Identif (для изображений)
4) ...

Метаданные должны отвечать на вопросы:

Почему?	Кто?	Что?	Где?	Когда?	Как?
Почему мы храним эти данные?	Кто создал эти данные?	Каковы бизнес-определения элементов данных?	Откуда взялись эти данные?	Когда были созданы эти данные?	Как эти данные форматируются?
Каково назначение и использование этих данных?	Кто использует эти данные?	Каковы бизнес-правила для этих данных?	Где хранятся эти данные?	Когда последний раз обновлялись эти данные?	Как много баз данных / источников хранят эти данные?
Каковы бизнес-драйверы для использования этих данных?	Кто является распорядителем этих данных?	Какие аббревиатуры / акронимы элементов данных?	Где используются переиспользуются эти данные?	Как долго должны храниться эти данные?	
	Кто владеет этими данными?	Каковы технические стандарты именования для реализации базы данных?	Где находится резервная копия этих данных?	Когда эти данные нужно удалять?	
	Кто регулирует / аудирует эти данные?	Каков уровень безопасности / конфиденциальности этих данных?	Есть ли региональные политики конфиденциальности / безопасности по регулированию этих данных?		

</>

Демонстрация

Сжатие данных – как это работает?

Главный признак данных, которые можно сжать - избыточность

Основной принцип алгоритмов сжатия базируется на том, что в любом файле, содержащем неслучайные данные, информация частично повторяется. Дальше идут математика с её моделями, что позволяет нам сжимать данные с потерями и без.

Gzip x bzip2 x XZ

Всё это разные утилиты, использующие разные алгоритмы сжатия данных.

К каждому типу архивирования используется своя тулза для разархивирования

Замечательная статья о где, сравниваются эти форматы сжатия в различных областях.

https://www.rootusers.com/gzip-vs-bzip2-vs-xz-performance-comparison/

Утилиты для сжатых данных без разархивации

- for 7z: 7z x -so example.7z
- for bz2: bzcat example.bz2
- for gz: zcat example.gz
- for xz: xzcat example.xz
- for zip: zcat example.zip
- for **bz2**, **gz** and **xz:** less example.bz2/gz/xz

</>

Демонстрация