以太坊源码解析 - RLP

RLP(Recursive Length Prefix)，叫递归长度前缀编码，它是以太坊序列化所采用的编码方式。RLP主要用于以太坊中数据的网络传输和持久化存储。

定义

RLP实际只给以下两种类型数据编码：

- byte数组
- byte数组的数组，称之为列表

byte数组

规则1：对于值在[0, 127]之间的单个字节，其编码是其本身。

例：a 的编码是 97 。

规则2： 如果byte数组长度l <= 55，编码的结果是数组本身，再加上128+l作为前缀。

例：abc 编码结果是131 97 98 99，其中131=128+len("abc")，97 98 99依次是a b c

规则3： 如果数组长度大于55， 编码结果第一个是183加上字符串长度所占用的字节数，然后是数组长度的本身的编码，最后是byte数组的编码。

编码下面这段字符串：

```
The length of this sentence is more than 55 bytes, I know it because I pre-designed it
```

这段字符串共86个字节，而86的编码只需要一个字节，那就是它自己，因此，编码的结果如下：

```
184 86 84 104 101 32 108 101 110 103 116 104 32 111 102 32 116 104 105 115 32 115 101 110 116 101 110 99 101 32 105 115 32 109 111 114 101 32 116 104 97 110 32 53 53 32 98 121 116 101 115 44 32 73 32 107 110 111 119 32 105 116 32 98 101 99 97 117 115 101 32 73 32 112 114 101 45 100 101 115 105 103 110 101 100 32 105 116
```

其中前三个字节的计算方式如下：

`184 = 183 + 1`，因为数组长度86编码后仅占用一个字节。`86`即数组长度`86`84是T的编码

编码一个重复1024次"a"的字符串，其结果为：`185 4 0 97 97 97 97 97 97 ...`。

1024 二进制 00000100 00000000 ，一个字节为8位，所以 1024 的字节数为 2 4 0 是数组长度的编码

列表

规则4：如果列表长度小于55，编码结果第一位是192加列表长度的编码的长度，然后依次连接各子列表的编码。

例6：["abc", "def"]的编码结果是 `200 131 97 98 99 131 100 101 102`。

其中 `abc` 的编码为 `131 97 98 99` , `def` 的编码为`131 100 101 102`。两个子字符串的编码后总长度是`8`，因此编码结果第一位计算得出：`192 + 8 = 200`。

规则5：如果列表长度超过55，编码结果第一位是247加列表长度的编码长度所占用的字节数，然后是列表长度本身的编码，最后依次连接各子列表的编码。

```
["The length of this sentence is more than 55 bytes, ", "I know it because I pre-designed it"]
```
的编码结果是:

```
248 88 179 84 104 101 32 108 101 110 103 116 104 32 111 102 32 116 104 105 115 32 115 101 110 116 101 110 99 101 32 105 115 32 109 111 114 101 32 116 104 97 110 32 53 53 32 98 121 116 101 115 44 32 163 73 32 107 110 111 119 32 105 116 32 98 101 99 97 117 115 101 32 73 32 112 114 101 45 100 101 115 105 103 110 101 100 32 105 116
```
其中前两个字节的计算方式如下：

列表长度 88 = 86 + 2，在规则3的示例中，长度为86，而在此例中，由于有两个子字符串，每个子字符串本身的长度的编码各占1字节，因此总共占2字节。

列表长度的编码长度为 1第1个字节为248 = 247 +1第2个字节为 88

第3个字节179依据**规则2**得出179 = 128 + 51第55个字节163同样依据**规则2**得出163 = 128 + 35

RLP解码

解码时，首先根据编码结果第一个字节f的大小，执行以下的规则判断：

- 如果f∈ [0,128)， 那么它是一个字节本身。
- 如果f∈[128,184)，那么它是一个长度不超过55的byte数组，数组的长度为 `l=f-128`
- 如果f∈[184,192)，那么它是一个长度超过55的数组，长度本身的编码长度`ll=f-183`,然后从第二个字节开始读取长度为ll的bytes，按照BigEndian编码成整数l，l即为数组的长度。
- 如果f∈(192,247]，那么它是一个编码后总长度不超过55的列表，列表长度为`l=f-192`。递归使用规则1~4进行解码。
- 如果f∈(247,256]，那么它是编码后长度大于55的列表，其长度本身的编码长度`ll=f-247`,然后从第二个字节读取长度为ll的bytes,按BigEndian编码成整数l，l即为子列表长度。然后递归根据解码规则进行解码。

ASCII

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUT	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f

7	BEL	39	,	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	–	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	/	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	`
31	US	63	?	95	_	127	DEL