Formát Redbin

Table of Contents

Specifikace verze 1

Redbin je binární formát, který přesně reprezentuje hodnoty Redu, uložené v paměti, přičemž umožňuje rychlé načítání (vyhýbaje se parsovací a ověřovací fázi prezentace textového formátu). Formát Redbin je převážně inspirován formátem REBin. Redbin umí zakódovat závaznou informaci pro slova a řídit cykly hodnot v blocích typu `any-block!`.

Uživatelské rozhraní pro formát Redbin bude zpřístupněno specifikovanými příkazy `load/binary` a `mold/binary`. Související implementace `by mohla` využít subsystém codec, jakmile bude k disposici.

Implementační omezení

- Bázová adresa v paměti pro načítaná data Redbin musí být zarovnaná na 64 bitů.

1. Formát kódování

Implicitní formát kódování je optimalizován pro rychlost dekódování, zatímco *kompaktní* formát vyžaduje menší úložný prostor (za cenu mnohem pomalejšího dekódování).

Hodnoty jsou uloženy ve formátu little-endian.

Lexikální konvence:

1. *Čísla v závorkách indikují velikost pole v bajtech.*
2. *Názvy polí následované blokem s názvem datového typu jsou držitelé místa pro případnou hodnotu.*
3. *Názvy polí následované rovnítkem mají fixní hodnotu.*

2. Záhlaví

```
magic="REDBIN" (6), version=1 (1), flags (1), length (4), size (4)

flags (option is enabled if bit is set):
    bit0: compact mode
    bit1: compressed
    bit2: symbol table
    bit3-7: <reserved>

length : počet načítaných kořenových záznamů (root records)
size   : velikost záznamů v bajtech.
```

Při použití komprese jsou data za záhlavím považovaná za payload (přenášený obsah), jenž má být komprimován. Výběr algoritmu pro kompresi závisí na implementaci.

3. Tabulka symbolů

Tabulka symbolů následuje bezprostředně za daty záhlaví. Je nepovinná a měla by být použita v případě, že slova jsou přítomná ve zbytku payloadu Redbinu. Tabulka má dvě části:

- tabulka offsetů (odsazení) pro stringovou prezentaci každého symbolu
- buffery stringů, zkončené nulou a navzájem propojené (concatenated)

Pozice symbolu v tabulce je daná jeho (nulou počínajícím) *indexem* a je použitá jako odkaz na symbol v kontextech a slovech. Sekce bufferů obsahuje UTF-8 kódované stringy s volitelnou výstelkou (padding) na konci, kvůli 64-bitovému zarovnání (alignment). Offsety v tabulce udávají velikost odsazení (v bajtech) od počátku sekce stringových bufferů k odkazovanému bufferu.

Kódování tabulky:

```
Default: length (4), size (4), offset1 (4), offset2 (4),...
Compact: TBD
```

Pole `length` obsahuje počet vstupů do tabulky. Pole `size` indikuje velikost *string bufferu* v bajtech (včetně nepovinných bajtů pro "tail padding").

Během dekódování jsou tyto symboly zahrnuty do vlastní tabulky symbolů a offsety jsou nahrazeny hodnotou ID symbolu z tabulky Redu. To jest, odkazy na symboly v záznamech Redbin jsou nepřímé odkazy na interní položky Redové tabulky symbolů, použité pouze při načítání.

Za tabulkou symbolů jsou hodnoty Redu ukládány jako záznamy postupně za sebou bez specielního oddělovače nebo markeru. Načtené hodnoty z kořenové úrovně jsou obvykle uloženy v řadách typu `block!`.

4. Definice záznamů

Každý záznam začíná polem `header`, definovaným jako:

- bit31 : new-line flag
- bit30 : no-values flag (for contexts)
- bit29 : stack? flag (for contexts)
- bit28 : self? flag (for contexts)
- bit27 : set? flag (for words)
- bit26-16 : <reserved>
- bit15-8 : unit (used for encoding elements size in a series buffer)
- bit7-0 : type

Dále následuje popis každého jednotlivého záznamu:

4.1. Padding

```
Default: header (4)
Compact: n/a

header/type=0
```

Tento prázdný typ slotu se používá k řádnému zarovnání 64-bitových hodnot.

4.2. Datatype!

```
Default: header (4), value (4)
Compact: TBD

header/type=1
```

4.3. Unset!

```
Default: header (4)
Compact: TBD

header/type=2
```

4.4. None!

```
Default: header (4)
Compact: TBD

header/type=3
```

4.5. Logic!

```
Default: header (4), value=0|1 (4)
Compact: TBD

header/type=4
```

4.6. Block!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=5
```

Pole `head` indikuje odsazení reference bloku s použitím indexu, počínajícího nulou. Pole `length` obsahuje počet hodnot, ukládaných v bloku. Hodnoty bloku jednoduše následují za definicí bloku, bez separátorů nebo oddělovačů.

4.7. Paren!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=6
```

Stejná pravidla kódování jako pro `block!`.

4.8. String!

```
Default: header (4), head (4), length (4), data (unit*length) [, padding (1-3)]
Compact: TBD

header/type=7
header/unit=1|2|4
```

Pole `head` ná stejný význam jako u bloků. Sub-pole `unit` indikuje formátování řetězce; přípustné jsou pouze hodnoty 1, 2 a 4. Pole `length` obsahuje počet kódových bodů (codepoints), které mají být v řetězci uloženy; podporováno je až 16777215 codepoints (2^24 - 1). Řetězec je kódován ve formátu UCS-1, UCS-2 nebo UCS-4. V poli `length` se neobjeví nulová hodnota. Volitelná výstelka (padding) o velikosti 1 až 3 nulových bajtů (NUL bytes) zarovná konec záznamu typu `string!` s 32-bitovou hranicí.

4.9. File!

```
Default: header (4), head (4), length (4), data (unit*length)
Compact: TBD

header/type=8
header/unit=1|2|4
```

Tatáž pravidla kódování jako u záznamu `string!.

4.10. Url!

```
Default: header (4), head (4), length (4), data (unit*length)
Compact: TBD

header/type=9
header/unit=1|2|4
```

Tatáž pravidla kódování jako u záznamu `string!.

4.11. Char!

```
Default: header (4), value (4)
Compact: TBD

header/type=10
```

4.12. Integer!

```
Default: header (4), value (4)
Compact: TBD

header/type=11
```

4.13. Float!

```
Default: [padding=0 (4),] header (4), value (8)
Compact: TBD

header/type=12
```

Volitelné výstelkové (padding) pole je přidáno kvůli řádnému zarovnání offsetu pole `value` k 64-bitové hranici.

4.14. Context!

```
Default: header (4), length (4), symbol1 (4), symbol2 (4),..., value1 [any-type!],
value2 [any-type!], ...
Compact: TBD

header/type=14
header/no-values=0|1
header/stack?=0|1
header/self?=0|1
```

Kontexty jsou Redové hodnoty, interně používané některými datovými typy jako `function!`, `object!` a odvozenými typy. Kontext obsahuje dvě související tabulky. První je seznam slov (word entries) v kontextu, reprezentovaných jako odkazy na symboly. Druhá tabulka obsahuje seznam přiřazených hodnot pro symboly v první tabulce. Pole `length` indikuje počet záznamů v kontextu. Tyto záznamy mohou existovat pouze na kořenové úrovni, nelze je nořit jeden do druhého. Je-li nastaven flag `no-values`, znamená to, že za symboly nejsou žádné hodnoty (empty context). Je-li nastaven flag `stack?`, potom jsou hodnoty alokovány na paměti "stack" místo na paměti "heap". Flag `self?` se používá k indikaci, že je kontext schopen zacházet i se slovem, které odkazuje samo na sebe (`self` v objektech).

4.15. Word!

```
Default: header (4), symbol (4), context (4), index (4)
Compact: TBD

header/type=15
header/set?=0|1
```

Pole `context` je offset od začátku sekce záznamů v souboru Redbin, odkazující na hodnotu typu `context!`. Kontext musí být umístěn před záznamem slova v seznamu záznamů Redbinu. Jestliže se `context` rovná -1, odkazuje na globální context.

Je-li definováno pole `set?`, je tento záznam následován záznamem s hodnotou typu `any-value!` a tato hodnota bude dekodérem ve správném kontextu přiřazena ke slovu. Toto vytváří dvojici

name/value, umožňující adhoc kódování hodnot jednotlivých slov, když poskytnutí sekvence hodnot pro daný kontext je příliš nákladné (většinou pro dvojice name/value v globálním kontextu).

4.16. Set-word!

```
Default: header (4), symbol (4), context (4), index (4)
Compact: TBD

header/type=16
```

Stejně jako pro `word!`.

4.17. Lit-word!

```
Default: header (4), symbol (4), context (4), index (4)
Compact: TBD

header/type=17
```

Stejně jako pro `word!`.

4.18. Get-word!

```
Default: header (4), symbol (4), context (4), index (4)
Compact: TBD

header/type=18
```

Stejně jako pro `word!`.

4.19. Refinement!

```
Default: header (4), symbol (4), context (4), index (4)
Compact: TBD

header/type=19
```

Stejně jako pro `word!`.

4.20. Issue!

```
Default: header (4), symbol (4)
Compact: TBD

header/type=20
```

4.21. Native!

```
Default: header (4), ID (4), spec [block!]
Compact: TBD

header/type=21
```

`ID` je offset do interní skokové tabulky `natives/table`.

4.22. Action!

```
Default: header (4), ID (4), spec [block!]
Compact: TBD

header/type=22
```

`ID` je offset do interní skokové tabulky `actions/table`.

4.23. Op!

```
Default: header (4), symbol (4),
Compact: TBD

header/type=23
```

`symbol` reprezentuje jméno action, native nebo function (pouze z globálního kontextu), použité jako zdroj pro tuto hodnotu op!.

4.24. Function!

```
Default: header (4), context [context!], spec [block!], body [block!], args [block!],
obj-ctx [context!]
Compact: TBD

header/type=24
```

4.25. Path!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=25
```

Stejná pravidla kódování jako pro `block!`.

4.26. Lit-path!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=26
```

Stejná pravidla kódování jako pro `block!`.

4.27. Set-path!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=27
```

Stejná pravidla kódování jako pro `block!`.

4.28. Get-path!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=28
```

Stejná pravidla kódování jako pro `block!`.

4.29. Bitset!

```
Default: header (4), length (4), bits (length)
Compact: TBD

header/type=30
```

Pole `length` indikuje počet uložených bitů, zaokrouhlený nahoru na násobek 8. Bity představují paměťová uložiště pro buffer řad typu `bitset!`. Pořadí bajtů je zachováno. Pole `bits` se doplňuje potřebným počtem nul (padding) pro zarovnání dalšího 32-bitového záznamu.

4.30. Point!

```
Default: header (4), x (4), y (4), z (4)
Compact: TBD

header/type=31
```

4.31. Object!

```
Default: header (4), context [reference!], class-id (4), on-set-idx (4), on-set-arity
(4)
Compact: TBD

header/type=32
```

Pole `on-set-idx` indikuje offset údaje `on-change*` v tabulce kontextových hodnot. Pole `on-set-arity` ukládá aritu dané funkce.

4.32. Typeset!

```
Default: header (4), array1 (4), array2 (4), array3 (4)
Compact: TBD

header/type=33
```

4.33. Error!

```
Default: header (4), context [reference!]
Compact: TBD

header/type=34
```

4.34. Vector!

```
Default: header (4), head (4), length (4), values (unit*length)
Compact: TBD

header/type=35
```

Pole `unit` indikuje velikost elementu vektoru, danou hodnotami: 1, 2, 4 nebo 8 bajtů. Pole `values` uchovává seznam hodnot. Hodnoty musí být doplněný nulovými bajty pro zarovnání dalšího záznamu s 32-bitové hranici (je-li `unit` roven 1 nebo 2).

4.35. Pair!

```
Default: header (4), x (4), y (4)
Compact: TBD

header/type=37
```

4.36. Percent!

```
Default: [padding=0 (4),] header (4), value (8)
Compact: TBD

header/type=38
```

Procentní hodnota je uložena jako 64-bitový float. Volitelné pole `padding` (výstelka) je přidáno pro řádné přisazení pole `value` k 64-bitové mezi.

4.37. Tuple!

```
Default: header (4), array1 (4), array2 (4), array3 (4)
Compact: TBD

header/type=39
```

4.38. Map!

```
Default: header (4), length (4), ...
Compact: TBD

header/type=40
```

Pole `length` obsahuje počet elementů (klíče + hodnoty), které mají být uloženy v mapě. Elementy mapy jednoduše sledují definici délky; žádné separátory nebo vymezovače nejsou požadovány.

4.39. Binary!

```
Default: header (4), head (4), length (4), ...
Compact: TBD

header/type=41
```

Stejná pravidla kódování jako pro `block!`

4.40. Time!

```
Default: [padding=0 (4),] header (4), value (8)
Compact: TBD

header/type=43
```

Časová hodota je uložena jako 64-bitový float. Volitelné pole výstelky (padding) je přidáno za účelem řádného přiřazení pole `value` k 64-bitové mezi.

4.41. Tag!

```
Default: header (4), head (4), length (4), data (unit*length)
Compact: TBD

header/type=44
header/unit=1|2|4
```

Stejná pravidla kódování jako pro `string!`.

4.42. Email!

```
Default: header (4), head (4), length (4), data (unit*length)
Compact: TBD

header/type=45
header/unit=1|2|4
```

Stejná pravidla kódování jako pro `string!`.

4.43. Date!

```
Default: header (4), date (4), time (8)
Compact: TBD

header/type=47
```

Datum je vložen do 32-bitové hodnoty typu integer! (stejně jako u`red-date!`). Časový údaj je uložen jako 64-bitový float.

4.44. Money!

```
Default: header (4), currency (1), amount (11)
Compact: TBD

header/type=49
header/sign=1|0 (bit 14)
```

TBTL: `amount` is a sequence of nibbles representing the money integral and decimal part (22 digits) in network byte order. If `sign` is set, the amount is interpreted as negative. `currency` is an integer value (0 for generic money, < 255 for existing currency code).

4.45. Ref!

```
Default: header (4), head (4), length (4), data (unit*length)
Compact: TBD

header/type=50
header/unit=1|2|4
```

Tatáž pravidla kódování jako pro `string!`.

4.46. Reference! anchor: reference[]

```
Default: header (4), count (4), index1 (4), index2 (4), ...
Compact: TBD

header/type=255
```

Tento speciální typ záznamu ukládá odkaz k již načtené hodnotě typu `any-block!` nebo `object!`. To umožňuje ukládání cyklů v Redbinu. Odkaz je vytvořen z cesty k načtené hodnotě (za předpokladu, že kořenové hodnoty jsou uloženy v bloku). Každé pole `index` ukazuje na řadu nebo na hodnotu objektu, k němuž se má postupně přejít. Pole `count` indikuje počet indexů, jimiž se má projít. Má-li

být jeden z indexů aplikován na objekt, pak odkazuje na odpovídající pole objektu (0 ⇒ 1. pole, 1 ⇒ 2. pole, …). Všechny indexy počínají nulou.