SmartHR

Binary "pack" Rebooted

PerlからRubyistへの贈り物

株式会社 SmartHR
Uchio Kondo

で、誰

- Uchio Kondo (id:udzura)
  - 普段は福岡部近辺にいます
- 所属: SmartHR
- アクキー配り係として来ました

- **PR**
  - **SmartHR は27新卒エンジニア募集中だよ！**
  - **詳しい話はぜひ声かけてください**

来年2月にも、
博多で熱いイベントがあるのを
ご存知ですか...？

#PR #jimlockさんが言えって言った

https://regional.rubykaigi.org/fukuoka05/

福岡Rubyist会議05 もよろしくお願いします

- 来年2月にもまた福岡で会おうぜ...
- CfP is open! **(~ 11/18)**
- Rubyじゃなくても「作ったもの自慢」して下さい!!1;

今日のテーマ

pack / unpack ?

ここにバイナリがあるじゃろ

```
0000000 0045 5400 2515 0040 0140 5acd 11ac 0400
0000010 11ac 0300 0008 e360 0900 0100 c69c 690e
0000020 0000 0000 a4b0 0008 0000 0000 2020 4159
0000030 4350 2020 4159 4350 2020 4159 4350 2020
0000040 4159 4350 2020 4159 4350 2020 4159 4350
0000050 2020 4159
0000054
```

unpack を使ってみましょう

```perl
sub parse_icmp_packet {
    my ($packet) = @_;
    my $ip_header_len = (unpack('C', $packet) & 0x0f) * 4;
    my ($ver_ihl, $tos, $tot_len, $id, $frag_off, $ttl, $protocol, $checksum, $saddr, $daddr) =
        unpack('C C n n n C C n N N', substr($packet, 0, 20));
    my $src_ip = inet_ntoa(pack('N', $saddr));
    my $dst_ip = inet_ntoa(pack('N', $daddr));
    my $icmp_packet = substr($packet, $ip_header_len);
    my ($type, $code, $icmp_checksum, $identifier, $sequence) =
        unpack('C C n n n', $icmp_packet);
    my $data = substr($icmp_packet, 8);
    # ...
}
```

unpack を使ってみましょう

```perl
sub parse_icmp_packet {
    my ($packet) = @_;
    my $ip_header_len = (unpack('C', $packet) & 0x0f) * 4;
    my ($ver_ihl, $tos, $tot_len, $id, $frag_off, $ttl, $protocol, $checksum, $saddr, $daddr) =
        unpack('C C n n n C C n N N', substr($packet, 0, 20));
    my $src_ip = inet_ntoa(pack('N', $saddr));
    my $dst_ip = inet_ntoa(pack('N', $daddr));
    my $icmp_packet = substr($packet, $ip_header_len);
    my ($type, $code, $icmp_checksum, $identifier, $sequence) =
        unpack('C C n n n', $icmp_packet);
    my $data = substr($icmp_packet, 8);
    # ...
}
```

僕にも読めた！

```
$ perl dump.pl ping.bin
$VAR1 = {
          'ip' => {
                    'tos' => 0,
                    'header_length' => 20,
                    'protocol' => 1,
                    'version' => 4,
                    'id' => 5413,
                    'dst_addr' => '172.17.0.3',
                    'ttl' => 64,
                    'total_length' => 84,
                    'src_addr' => '172.17.0.4'
                  },
          'icmp' => {
                      'checksum' => '0x60e3',
                      'sequence' => 1,
                      'data' => '??i?  YAPC  YAPC  YAPC  YAPC  YAPC  YAPC  YA',
                      'code' => 0,
                      'type' => 8,
                      'identifier' => 9
                    }
        };
```

pack/unpack 使い所

- フォーマットを覚えればバイナリパースができる
  - パケットやメモリダンプなど...
- 一部のFFI/システムコールの直接呼びができる
  - 構造体を渡す必要があるAPIで利用
  - packで構造体によるメモリ上のデータを作成する

こんな便利な pack / unpack
すぐにでも使いたい
ですよね…？

超入門 pack / unpack

フォーマット文字 覚え方

- サイズ
  - `CSLQ` (後述)
- エンディアン
  - 基本的にCPUに合わせてくれるので意識せず
- 符号あるなし
  - 大文字は無し、小文字は有り

まとめ

フォーマット	符号	サイズ	エンディアン
C	無し	1バイト	N/A
c	有り	1バイト	N/A
S	無し	2バイト	CPU依存
s	有り	2バイト	CPU依存
l	無し	4バイト	CPU依存
L	有り	4バイト	CPU依存
q	無し	8バイト	CPU依存
Q	有り	8バイト	CPU依存

エンディアン固定もできる

フォーマット	符号	サイズ	エンディアン
n	無し	2バイト	ビッグエンディアン
v	無し	2バイト	リトルエンディアン
N	無し	4バイト	ビッグエンディアン
V	無し	4バイト	リトルエンディアン

実は浮動小数も扱える

フォーマット	サイズ	エンディアン
`f`	4バイト	CPU依存
`d`	8バイト	CPU依存

覚えておくと便利なもの

フォーマット	説明
Z	ヌル終端文字列
b/B	ビット列
h/H	16進文字列

繰り返しもできる

フォーマット	説明
L3	uint32を3つ繰り返し読み込む
Z3	文字を3つ繰り返し読み込む
Z*	文字を、ヌル文字か終わりまで読み込む

まとめ: これだけ覚えればいい説

フォーマット	意味	覚え方
C	1bit/符号なし	Char (文字)
S	2bit/符号なし	Short (短い)
L	4bit/符号なし	Long (長いと言いつつ普通)
Q	8bit/符号なし	Quad (4つめ)
Z	char[] やつ	Zero-terminated (ゼロ終端文字)
*	いい感じに繰り返し	グロブ的な...

~~正味な話AIが書いてくれるんでは…？~~

pack の歴史

Ruby での歴史にも触れます

Perl 3

- 3.0で pack/unpack が導入されたのを発見
  - Oct 18, 1989
- それより前は辿れず...

```
#!./perl

# $Header: op.pack,v 3.0 89/10/18 15:30:39 lwall Locked $

print "1..3\n";

$format = "c2x5CCxsila6";
@ary = (1,-100,127,128,32767,12345,123456,"abcdef");
$foo = pack($format,@ary);
@ary2 = unpack($format,$foo);

print ($#ary == $#ary2 ? "ok 1\n" : "not ok 1\n");

$out1=join(':',@ary);
$out2=join(':',@ary2);
print ($out1 eq $out2 ? "ok 2\n" : "not ok 2\n");

print ($foo =~ /def/ ? "ok 3\n" : "not ok 3\n");
```

Perl 4 ～

- 4.0 にもある
  - Mar 21, 1991
- テストケース変わらず。
- 5.0 ではちょっとテストケース増えた
- これ以降は基本関数としてずっとある

```
#!./perl

# $RCSfile: pack.t,v $$Revision: 4.1 $$Date: 92/08/07 18:28:11 $

print "1..8\n";

$format = "c2x5CCxsdila6";
# Need the expression in here to force ary[5] to be numeric.  This avoids
# test2 failing because ary2 goes str->numeric->str and ary doesn't.
@ary = (1,-100,127,128,32767,987.654321098 / 100.0,12345,123456,"abcdef");
$foo = pack($format,@ary);
@ary2 = unpack($format,$foo);

print ($#ary == $#ary2 ? "ok 1\n" : "not ok 1\n");

$out1=join(':',@ary);
$out2=join(':',@ary2);
print ($out1 eq $out2 ? "ok 2\n" : "not ok 2\n");

print ($foo =~ /def/ ? "ok 3\n" : "not ok 3\n");

# How about counting bits?

print +($x = unpack("%32B*", "\001\002\004\010\020\040\100\200\377")) == 16
    ? "ok 4\n" : "not ok 4 $x\n";

print +($x = unpack("%32b69", "\001\002\004\010\020\040\100\200\017")) == 12
    ? "ok 5\n" : "not ok 5 $x\n";

print +($x = unpack("%32B69", "\001\002\004\010\020\040\100\200\017")) == 9
    ? "ok 6\n" : "not ok 6 $x\n";

print +($x = unpack("%32B*", "Now is the time for all good blurfl")) == 129
    ? "ok 7\n" : "not ok 7 $x\n";

open(BIN, "./perl") || die "Can't open ../perl: $!\n";
sysread BIN, $foo, 8192;
close BIN;

$sum = unpack("%32b*", $foo);
$longway = unpack("b*", $foo);
print $sum == $longway =~ tr/1/1/ ? "ok 8\n" : "not ok 8\n";
```

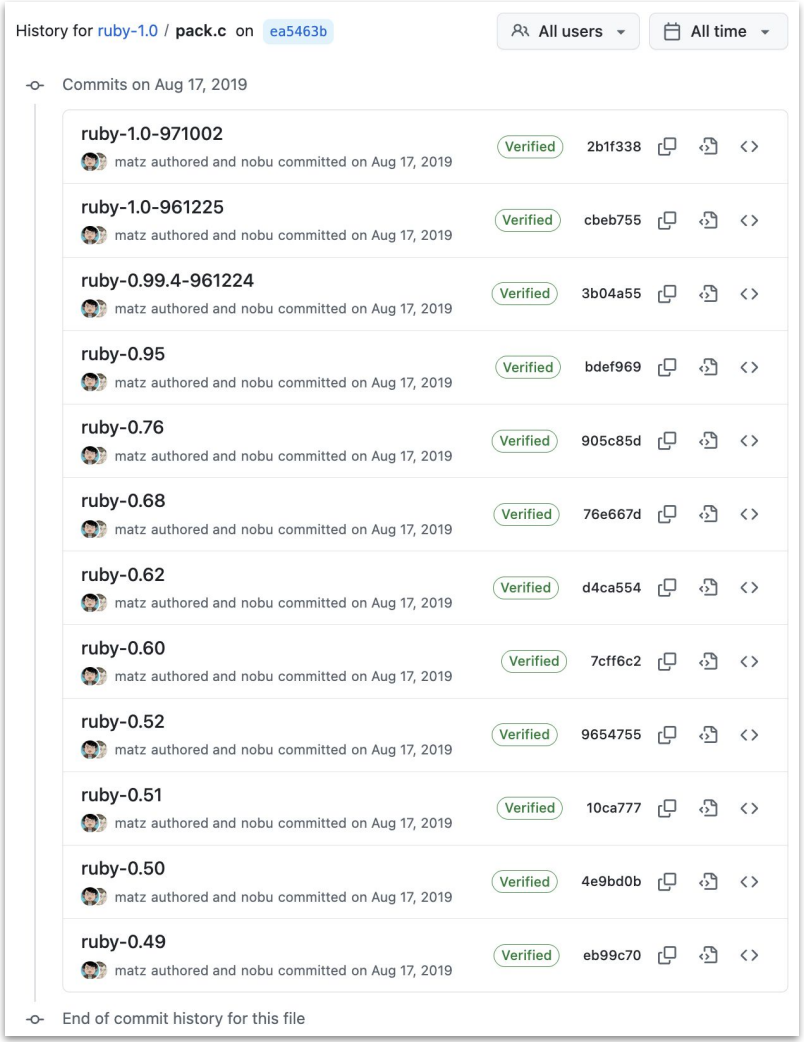
1.0より前のRuby

- 1.0 時点でメソッドが存在
  - Ruby 1.0は Dec 25, 1996 リリース
- しかも、`Array#pack` と `String#unpack` という同じインタフェース

```
878     void
879     Init_pack()
880     {
881         rb_define_method(cArray, "pack", pack_pack, 1);
882         rb_define_method(cString, "unpack", pack_unpack, 1);
883     }
```

コミットを見ると...

- 最初の最初からあるくね？



CVSのエントリログを見ると

```
/pack.c/0.18/Fri Jun  3 00:15:17 1994 Fri Jun  3 00:15:17 1994//
```

- Jun 3, 1994 からあるように見える
- 最初の最初からPerlを意識して？実装していた

本当のところは今度Matzに聞きます

あれ？　今日来ていないんですか...？

ここまでのまとめ

- pack/unpack はPerlからRubyに来た
- 逆にいえばRubyの `String#unpack` を知っていれば大体同じノリでPerlでも使えるんでは!!1;

ということで... 実践編

例: Rubyの文字列を表現した構造体(3.4) ※ 抜粋

```c
typedef uintptr_t VALUE;
struct RString {
    struct RBasic basic;
    long len;
    union {
        struct { ...snip... } heap;
        struct { char ary[]; } embed;
    } as;
};
struct RBasic {
    VALUE flags;
    const VALUE klass;
    // #if ... snip...
};
```

短い文字列はこういう構造体と一致するはず（64bit CPU）

```
struct RString_ {
    uintptr_t flags; // Q
    const uintptr_t klass; // Q
    long len; // Q
    char ary[]; // Z*
};
```

Rubyのメモリ上の文字列を取り出す

- Perlから直接メモリを確認してみよう
- ライブデモします！！

資料として後から追う用のコードサンプル

まずRubyのStringのアドレスを出す魔法

```ruby
# Showing Ruby "Magic" at YAPC
class String
  alias original_inspect inspect
  def inspect; super; end
  alias inspect_with_address inspect
  alias inspect original_inspect
end
```

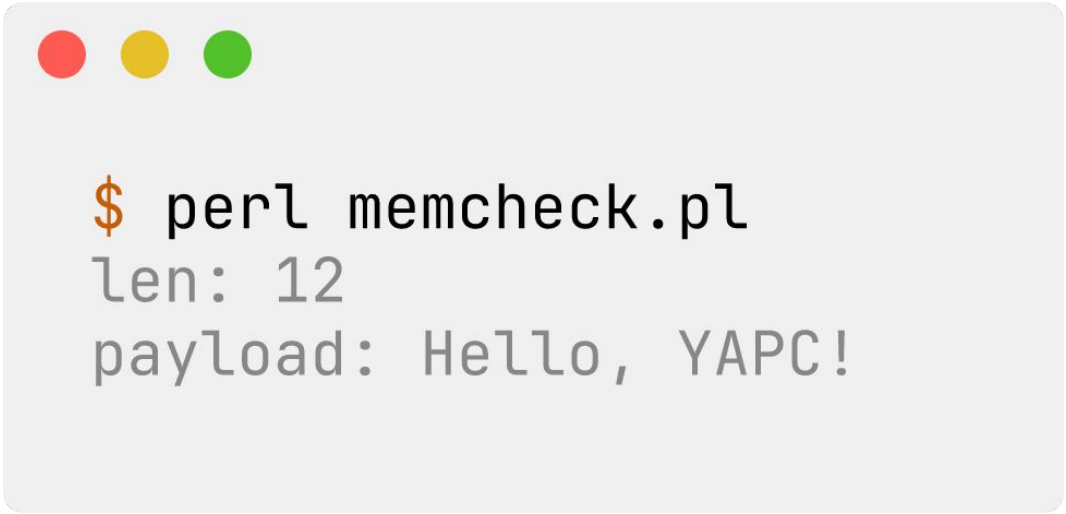
```ruby
str = "Hello, YAPC!" #⇒ "Hello, YAPC!"
puts str.inspect_with_address
#<String:0x0000fe263bd10528>
p $$
#⇒ 17855
sleep # keep process alive
```

Rubyのメモリを直接確認してみよう

```
use Data::Dumper;
open my $fh, '<:raw', "/proc/17855/mem" or die "$!\n";
seek $fh, 0x0000fe263bd10528, 0;
read $fh, my $data, 100;
close $fh;
# ...
```

```
# ...
my ($f, $c, $len, $payload) = unpack("QQQZ*", $data);
print "len: $len\n";
print "payload: $payload\n";
```

見えました



まとめ

Rubyistだけど、Perlのpack完全に理解しました!!1;

おしまい

本発表の写真: 丸ごと福岡・博多 より適当に

packとunpack、楽しんでみてね！

~~け◯た食堂風~~