Ruby master - Feature #18254
Add an `offset` parameter to String#unpack and String#unpack1

10/18/2021 08:04 AM - byroot (Jean Boussier)

**Status:** Closed  
**Priority:** Normal  
**Assignee:**  
**Target version:**

**Description**
When working with binary protocols it's common to have to first unpack some kind of header or type prefix, and then based on that unpack another part of the string.

For instance here's a code snippet from Dalli, the most common Memcached client:

```ruby
while buf.bytesize - pos >= 24
  header = buf.slice(pos, 24)
  (key_length, _, body_length, cas) = header.unpack(KV_HEADER)

  if key_length == 0
    # all done!
    @multi_buffer = nil
    @position = nil
    @inprogress = false
    break

  elsif buf.bytesize - pos >= 24 + body_length
    flags = buf.slice(pos + 24, 4).unpack1("N")
    key = buf.slice(pos + 24 + 4, key_length)
    value = buf.slice(pos + 24 + 4 + key_length, body_length - key_length - 4) if body_length - key_length - 4 > 0
    pos = pos + 24 + body_length

    begin
      values[key] = [deserialize(value, flags), cas]
      rescue DalliError
        end
    else
      # not enough data yet, wait for more
      break
    end
  end
@position = pos
```

**Proposal**

If unpack and unpack1 had an offset parameter, it would allow this kind of code to extract the fields it needs without allocating and copying as much strings, e.g.:

```ruby
flags = buf.slice(pos + 24, 4).unpack1("N")
```

could be:

```ruby
buf.unpack1("N", offset: pos + 24)
```

**Associated revisions**

Revision e5319dc9 - 10/26/2021 08:27 PM - byroot (Jean Boussier)

pack.c: add an offset argument to unpack and unpack1

[Feature #18254]
This is useful to avoid repeatedly copying strings when parsing binary formats

History

#1 - 10/18/2021 08:24 AM - znz (Kazuhiro NISHIYAMA)
You can use unpack1("@#{pos + 24}N").

#2 - 10/18/2021 08:32 AM - byroot (Jean Boussier)
Ah, I didn't know about it, but then you just allocated a string and converted an integer to string, so it's even slower than the slice pattern:

```ruby
# frozen_string_literal: true
require 'benchmark/ips'

STRING = Random.bytes(200)
POS = 12
Benchmark.ips do |x|
  x.report("no-offset") { STRING.unpack1("N") }
  x.report("slice-offset") { STRING.slice(POS, 4).unpack1("N") }
  x.report("unpack-offset") { STRING.unpack1("@#{POS}N") }
end
```

#3 - 10/18/2021 02:26 PM - byroot (Jean Boussier)
I submitted a pull request for it, https://github.com/ruby/ruby/pull/4984.

#4 - 10/25/2021 02:08 AM - matz (Yukihiro Matsumoto)
Sounds reasonable. Accepted.

Matz.

#5 - 10/25/2021 02:36 AM - mame (Yusuke Endoh)
Just a confirmation: the offset is byte-oriented, not character-oriented, right? There are a format "u" which is UTF-8 coding, so the behavior should be explained clearly in the document.

#6 - 10/25/2021 05:39 AM - nobu (Nobuyoshi Nakada)
As the RDoc of String#unpack states:

```
# Decodes <i>str</i> (which may contain binary data) according to the
# format string, returning an array of each value extracted. The
```

Isn't it clear that it is counted as binary?

#7 - 10/25/2021 06:55 AM - byroot (Jean Boussier)

Just a confirmation: the offset is byte-oriented, not character-oriented, right? There

Yes.

#8 - 10/25/2021 09:26 AM - duerst (Martin Dürst)
mame (Yusuke Endoh) wrote in #note-5:

> Just a confirmation: the offset is byte-oriented, not character-oriented, right? There are a format "u" which is UTF-8 coding, so the behavior should be explained clearly in the document.

This is not only a problem of "explain it in the document". In order for this offset to work well, there should be a way to know how many bytes an invocation of String#unpack consumes. In many cases, that's very easy to calculate from the format string, but in others, in particular for UTF-8, it's not easy.

#9 - 10/25/2021 09:41 AM - byroot (Jean Boussier)

That argument will indeed be pretty much worthless if you use the U format, but I don't really see it as a blocker. It is meant to help binary parsers, I don't see U making sense for these.

As for the documentation, we indeed need to be clear that it's a **byte** offset.

#10 - 10/25/2021 02:34 PM - byroot (Jean Boussier)

I extended the pull request to clearly document the offset keyword and stress that it's a byte offset. Hopefully that clears that concern.

#11 - 10/25/2021 04:55 PM - mame (Yusuke Endoh)

byroot (Jean Boussier): Thank you for adding documentation. I agree with merging.

> there should be a way to know how many bytes an invocation of String#unpack consumes.

In fact, some committers discussed this point at the dev-meeting. However, in many cases, it is trivial (or able to calculate) for a programmer how many bytes are consumed. Also, it looks difficult to provide the feature by just extending the current API design of String#unpack. So, matz concluded that those who really wants the feature should create another ticket with use case discussion and a concrete API proposal.

#12 - 10/25/2021 05:36 PM - byroot (Jean Boussier)

Agreed. The goal is to avoid slicing anyway, and to slice you need to know how many bytes you consumed.

If there's no other objections I'll merge in a day or two.

#13 - 10/26/2021 08:27 PM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|e5319dc9856298f38aa9c66ed55e39ad60e8e070.

pack.c: add an offset argument to unpack and unpack1

[Feature #18254]

This is useful to avoid repeteadly copying strings when parsing binary formats