Library for Converting Data to and from C Structs for Lua 5.1/5.2

(download)

This library offers basic facilities to convert Lua values to and from C structs. Its main functions are `struct.pack`, which packs multiple Lua values into a struct-like string; and `struct.unpack`, which unpacks multiple Lua values from a given struct-like string.

The fist argument to both functions is a *format string*, which describes the layout of the structure. The format string is a sequence of conversion elements, which respect the current endianess and the current alignment requirements. Initially, the current endianess is the machine's native endianness and the current alignment requirement is 1 (meaning no alignment at all). You can change these settings with appropriate directives in the format string.

The elements in the format string are as follows:

- `" "` (empty space) ignored.
- `"!n"` flag to set the current alignment requirement to *n* (necessarily a power of 2); an absent *n* means the machine's native alignment.
- `">"` flag to set mode to big endian.
- `"<"` flag to set mode to little endian.
- `"x"` a padding zero byte with no corresponding Lua value.
- `"b"` a signed `char`.
- `"B"` an unsigned `char`.
- `"h"` a signed `short` (native size).
- `"H"` an unsigned `short` (native size).
- `"l"` a signed `long` (native size).
- `"L"` an unsigned `long` (native size).
- `"T"` a `size_t` (native size).
- `"in"` a signed integer with *n* bytes. An absent *n* means the native size of an `int`.
- `"In"` like `"in"` but unsigned.
- `"f"` a `float` (native size).
- `"d"` a `double` (native size).
- `"s"` a zero-terminated string.
- `"cn"` a sequence of exactly *n* chars corresponding to a single Lua string. An absent *n* means 1. When packing, the given string must have at least *n* characters (extra characters are discarded).
- `"c0"` this is like `"cn"`, except that the *n* is given by other means: When packing, *n* is the length of the given string; when unpacking, *n* is the value of the previous unpacked value (which must be a number). In that case, this previous value is not returned.

Lua API

- `struct.pack (fmt, d1, d2, ...)`

  Returns a string containing the values `d1`, `d2`, etc. packed according to the format string `fmt`.

- `struct.unpack (fmt, s, [i])`

  Returns the values packed in string `s` according to the format string `fmt`. An optional `i` marks where in `s` to start reading (default is 1). After the read values, this function also returns the index in `s` where it stopped reading, which is also where you should start to read the rest of the string.

- `struct.size (fmt)`

  Returns the size of a string formatted according to the format string `fmt`. The format string should contain neither the option `s` nor the option `c0`.

Examples

- The code `print(struct.size("i"))` prints the size of a machine's native `int`.

- To pack and unpack the structure

  ```
  struct Str {
    char b;
    int i[4];
  };
  ```

  you can use the string `"<!4biiii"`.

- If you need to code a structure with a large array, you may use `string.rep` to automatically generate part of the string format. For instance, for the structure

  ```
  struct Str {
    double x;
    int i[400];
  };
  ```

  you can build the format string with the code `"d"..string.rep("i", 400)`.

- To pack a string with its length coded in its first byte, use the following code:

  ```
  x = struct.pack("Bc0", string.len(s), s)
  ```

  To unpack that string, do as follows:

  ```
  s = struct.unpack("Bc0", x)
  ```

  Note that the length (read by the element `"B"`) is not returned.

- Suppose we have to decode a string `s` with an unknown number of doubles; the end is marked by a zero value. We can use the following code:

  ```
  local a = {}
  local i = 1         -- index where to read
  while true do
    local d
    d, i = struct.unpack("d", s, i)
    if d == 0 then break end
    a[#a + 1] = d
  end
  ```

- To pack a string in a fixed-width field of 10 characters padded with blanks, do as follows:

  ```
  x = struct.pack("c10", s .. string.rep(" ", 10))
  ```

License