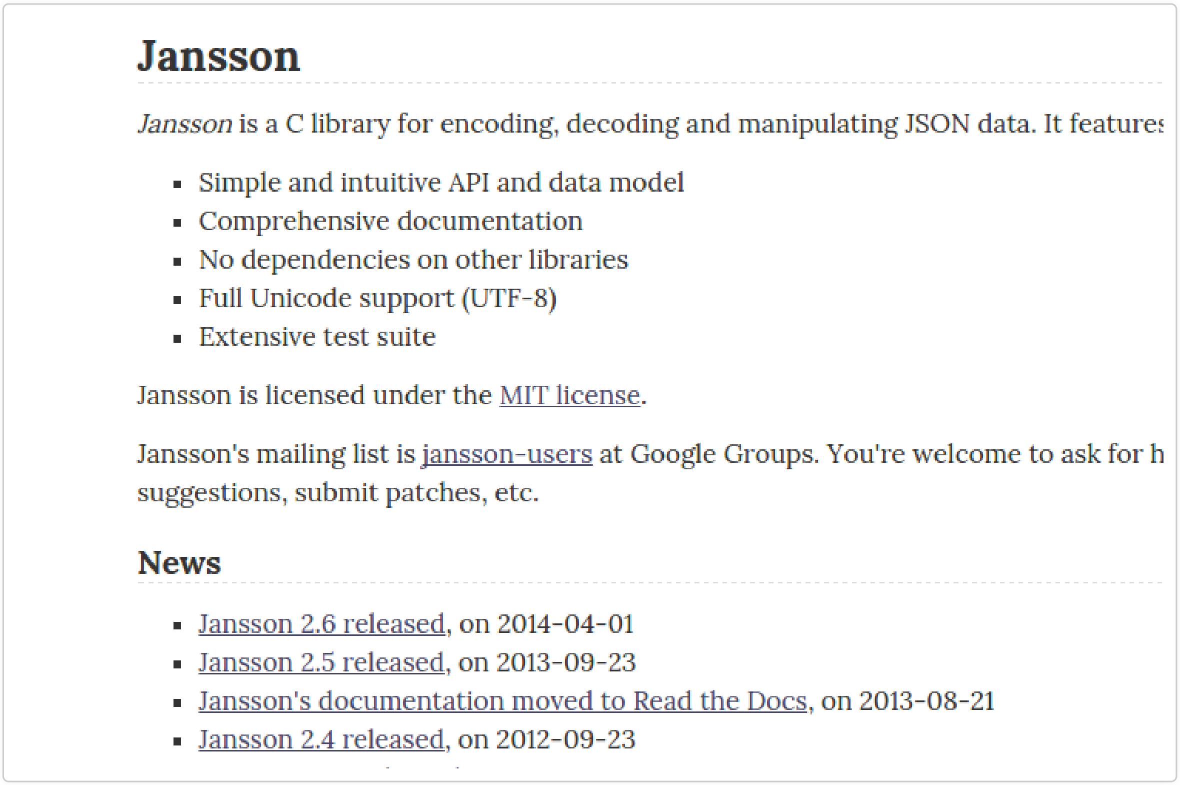
Peter

My Logbook

Thursday, February 12, 2015

C: JSON String Generation Using Jansson Library



Jansson

Jansson is a C library for encoding, decoding and manipulating JSON data. It features

- Simple and intuitive API and data model
- Comprehensive documentation
- No dependencies on other libraries
- Full Unicode support (UTF-8)
- Extensive test suite

Jansson is licensed under the MIT license.

Jansson's mailing list is jansson-users at Google Groups. You're welcome to ask for h suggestions, submit patches, etc.

News

- Jansson 2.6 released, on 2014-04-01
- Jansson 2.5 released, on 2013-09-23
- Jansson's documentation moved to Read the Docs, on 2013-08-21
- Jansson 2.4 released, on 2012-09-23

In keeping up with the Arduino-to-RPi NRF24L01+ (DIY Belkin Wemo project), I move on to testing out libraries that can parse and generate JSON strings. The library I'm using is Jansson as it seemed to have gotten quite a bit of praise of all the 20-30ish listed on the JSON.org.

This time, I'm simply testing out JSON string generation and leaving off the parsing to next time. Arduino side will also need a JSON library but that will come later.

I'm using Xubuntu 14.10 and good thing Ubuntu has the Jansson library in the repository so I'm saving time by installing from apt-get:

```
sudo apt-get install libjansson-dev
```

Create a new c file. Jansson JSON object has the type **json_t**. To put anything (JSONObject, JSONArray, string, or integer), the call is json_object_set_new(). Here's a very simple example I whipped up.

```
#include <stdio.h>
#include <string.h>
#include <jansson.h>

int main(void) {

  char* s = NULL;

  json_t *root = json_object();
  json_t *json_arr = json_array();

  json_object_set_new( root, "destID", json_integer( 1 ) );
  json_object_set_new( root, "command", json_string("enable") );
  json_object_set_new( root, "respond", json_integer( 0 ));
  json_object_set_new( root, "data", json_arr );

  json_array_append( json_arr, json_integer(11) );
  json_array_append( json_arr, json_integer(12) );
  json_array_append( json_arr, json_integer(14) );
  json_array_append( json_arr, json_integer(9) );

  s = json_dumps(root, 0);

  puts(s);
  json_decref(root);

 return 0;
}
```

The code is pretty self-explanatory. I create a root JSON Object and an JSON Array that I will be appending to one of the entries in the JSON Object.

```
  json_t *root = json_object();
  json_t *json_arr = json_array();
```

Then I add 4 entries to this JSON Object: "destID", "command", "respond", and "data". The first three are of types integer (1), string ("enable"), and integer (0). The last one is of a JSON Array that I've created earlier.

```
  json_object_set_new( root, "destID", json_integer( 1 ) );
  json_object_set_new( root, "command", json_string("enable") );
  json_object_set_new( root, "respond", json_integer( 0 ));
  json_object_set_new( root, "data", json_arr );
```

I've yet added any array elements in the JSON Array. The following lines add four integer values to the array.

```
  json_array_append( json_arr, json_integer(11) );
  json_array_append( json_arr, json_integer(12) );
  json_array_append( json_arr, json_integer(14) );
  json_array_append( json_arr, json_integer(9) );
```

Finally I dump the JSON Object into a string format and output it to screen.

```
  s = json_dumps(root, 0);
  puts(s);
```

Last I have to free the JSON Object file.

```
  json_decref(root);
```

Remember to compile this with the **-ljansson**. For example:

```
gcc -o test test.c -ljansson
```

Running this will give the output:

```
{"destID": 1, "command": "enable", "respond": 0, "data": [11, 12, 14, 9]}
```

That's it for today. Now I should be able to generate strings to send from my RPi to Arduino, next up is for Arduino to parse and read the incoming JSON message.

at 12:05 AM

15 comments :

Anonymous February 25, 2016 at 4:19 AM

thanks.. it helps me lot

Reply

Unknown March 11, 2016 at 7:27 AM

And what is the puts(something)? Belongs to string.h ?

Reply

Replies

Unknown March 11, 2016 at 9:44 AM

That writes the string out to stdout. That's how you get the output of the string.

Reply

Unknown June 18, 2016 at 2:22 PM

Thanks a lot buddy, keep up the good work.

Reply

Unknown September 13, 2016 at 6:34 AM

See http://jansson.readthedocs.io/en/2.8/apiref.html when using json_dumps.

char *json_dumps(const json_t *json, size_t flags)
Returns the JSON representation of json as a string, or NULL on error. flags is described above. The return value must be freed by the caller using free().

free(s);

Reply

Replies

Unknown December 20, 2018 at 8:53 AM

thank you

Reply

Unknown May 10, 2017 at 6:12 PM

Neat and precise! Thanks for the post. Minor bug? : free(s) after puts(s)

Reply

Replies

Anonymous February 20, 2018 at 7:10 AM

No need, json_decref(root); will take care of freeing...

Unknown March 17, 2018 at 9:57 AM

I don't think so. Following the doc.

char *json_dumps(const json_t *json, size_t flags)
Returns the JSON representation of json as a string, or NULL on error. flags is described above. The return value must be freed by the caller using free().

Reply

SRIRAMVAMSI July 27, 2017 at 10:49 PM

This is a very useful example. Thanks for sharing!

Reply

Alexander Pozdneev August 27, 2017 at 9:16 AM

I believe, `json_array_append_new()` should be used instead of `json_array_append()`, as the former would steal the object.

Reply

Replies

Unknown June 29, 2019 at 8:39 PM

I think you are right,thanks for mention that.

Reply

Unknown May 1, 2018 at 1:57 AM

how can we create a json object form string?

Reply

Anonymous September 22, 2020 at 2:06 AM

I think you have a memory leak, you should use 'json_array_append_new'. Then also a free over s as someone already stated before

Reply

Unknown February 7, 2021 at 10:57 PM

how to increase the heap size of the json object

Reply

Enter Comment

Newer Post Home Older Post