A portable (OSX/Linux/Windows), simple zip library written in C

This is done by hacking awesome miniz library and layering functions on top of the miniz v1.15 API.

The Idea

... Some day, I was looking for zip library written in C for my project, but I could not find anything simple enough and lightweight. Everything what I tried required 'crazy mental gymnastics' to integrate or had some limitations or was too heavy. I hate frameworks, factories and adding new dependencies. If I must to install all those dependencies and link new library, I'm getting almost sick. I wanted something powerfull and small enough, so I could add just a few files and compile them into my project. And finally I found miniz. Miniz is a lossless, high performance data compression library in a single source file. I only needed simple interface to append buffers or files to the current zip-entry. Thanks to this feature I'm able to merge many files/buffers and compress them on-the-fly.

It was the reason, why I decided to write zip module on top of the miniz. It required a little bit hacking and wrapping some functions, but I kept simplicity. So, you can grab these 3 files and compile them into your project. I hope that interface is also extremely simple, so you will not have any problems to understand it.

Examples

- Create a new zip archive with default compression level.

```
struct zip_t *zip = zip_open("foo.zip", ZIP_DEFAULT_COMPRESSION_LEVEL, 'w');
{
    zip_entry_open(zip, "foo-1.txt");
    {
        char *buf = "Some data here...";
        zip_entry_write(zip, buf, strlen(buf));
    }
    zip_entry_close(zip);

    zip_entry_open(zip, "foo-2.txt");
```

```
    {
        // merge 3 files into one entry and compress them on-the-fly.
        zip_entry_fwrite(zip, "foo-2.1.txt");
        zip_entry_fwrite(zip, "foo-2.2.txt");
        zip_entry_fwrite(zip, "foo-2.3.txt");
    }
    zip_entry_close(zip);
}
zip_close(zip);
```

- Append to the existing zip archive.

```
struct zip_t *zip = zip_open("foo.zip", ZIP_DEFAULT_COMPRESSION_LEVEL, 'a');
{
    zip_entry_open(zip, "foo-3.txt");
    {
        char *buf = "Append some data here...";
        zip_entry_write(zip, buf, strlen(buf));
    }
    zip_entry_close(zip);
}
zip_close(zip);
```

- Extract a zip archive into a folder.

```
int on_extract_entry(const char *filename, void *arg) {
    static int i = 0;
    int n = *(int *)arg;
    printf("Extracted: %s (%d of %d)\n", filename, ++i, n);

    return 0;
}

int arg = 2;
zip_extract("foo.zip", "/tmp", on_extract_entry, &arg);
```

- Extract a zip entry into memory.

```
void *buf = NULL;
size_t bufsize;

struct zip_t *zip = zip_open("foo.zip", 0, 'r');
{
    zip_entry_open(zip, "foo-1.txt");
    {
        zip_entry_read(zip, &buf, &bufsize);
    }
    zip_entry_close(zip);
}
zip_close(zip);

free(buf);
```

- Extract a zip entry into memory using callback.

```
  struct buffer_t {
      char *data;
      size_t size;
  };

  static size_t on_extract(void *arg, unsigned long long offset, const void
*data, size_t size) {
      struct buffer_t *buf = (struct buffer_t *)arg;
      buf->data = realloc(buf->data, buf->size + size + 1);
      assert(NULL != buf->data);

      memcpy(&(buf->data[buf->size]), data, size);
      buf->size += size;
      buf->data[buf->size] = 0;

      return size;
  }

  struct buffer_t buf = {0};
  struct zip_t *zip = zip_open("foo.zip", 0, 'r');
  {
      zip_entry_open(zip, "foo-1.txt");
      {
          zip_entry_extract(zip, on_extract, &buf);
      }
      zip_entry_close(zip);
  }
  zip_close(zip);

  free(buf.data);
```

- Extract a zip entry into a file.

```
  struct zip_t *zip = zip_open("foo.zip", 0, 'r');
  {
      zip_entry_open(zip, "foo-2.txt");
      {
          zip_entry_fread(zip, "foo-2.txt");
      }
      zip_entry_close(zip);
  }
  zip_close(zip);
```