Package ‘fitdc’

September 18, 2016

Title Garmin FIT File Decoder

Version 0.0.1

Description A pure R package for decoding activity files written in the FIT (``Flexible and Interoperable Data Transfer'') format. A format that is fast becoming the standard for recording running and cycling data. Details of the FIT protocol can be found at <https://www.thisisant.com/resources/fit>.

Depends R (>= 3.3.1)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL `https://github.com/jmackie4/fitdc`

RoxygenNote 5.0.1

NeedsCompilation no

Author Jordan Mackie [aut, cre]

Maintainer Jordan Mackie <`jmackie@protonmail.com`>

Repository CRAN

Date/Publication 2016-09-18 08:56:52

R topics documented:

`fitdc` *fitdc: R package for decoding FIT files.*

Description

A pure R FIT file decoder. After having a package pulled from CRAN due to licensing issues with the FIT SDK, the only way to bring that package back from the dead was to put this together. As far as I'm concerned, it is fit for purpose, but I am aware it is not a *complete* implementation. If this does not meet your needs in its current state, feel free to submit a patch.

Details

Anyhow, performance is suprisingly good, and the package is written to have no external dependencies. I hope you find it useful!

`read_fit` *Decode a FIT file*

Description

Decode a FIT file

Usage

```
read_fit(file_path)
```

Arguments

`file_path`	string; path to the FIT file to be read.

Value

decoded *data* messages from the FIT file.

Examples

```
## An example of generating a table of record messages
## from the file provided with this package:

fp <- system.file("extdata/example.fit", package = "fitdc")
data_mesgs <- read_fit(fp)

## Filter out the record messages:

is_record <- function(mesg) mesg$name == "record"
records <- Filter(is_record, data_mesgs)
```

```
format_record <- function(record) {
  out <- record$fields
  names(out) <- paste(names(out), record$units, sep = ".")
  out
}

records <- lapply(records, format_record)

## Some records have missing fields:

colnames_full <- names(records[[which.max(lengths(records))]])
empty <- setNames(
  as.list(rep(NA, length(colnames_full))),
  colnames_full)

merge_lists <- function(ls_part, ls_full) {
  extra <- setdiff(names(ls_full), names(ls_part))
  append(ls_part, ls_full[extra])[names(ls_full)]  # order as well
}

records <- lapply(records, merge_lists, empty)
records <- data.frame(
  do.call(rbind, records))

head(records)  # voila
```

unpack *Read and unpack bytes from a binary file connection*

Description

This function is exported mainly for my own benefit, but maybe others will find it useful. It is written to bring the python syntax for binary file reading to R. See the source code of this package for usage examples.

Note a limitation of this approach to binary file reading is that *reading* and *unpacking* are inseparable, which can cause headaches in some cases.

Usage

```
unpack(fmt, conn, endianness = "little", n = 1, ...)
```

Arguments

`fmt`	a format character according to the python struct library docs. The following are currently supported: `"xbBhHiIs"`.
`conn`	a connection returned by `file`.
`endianness`	string; passed to `readBin`. One of `"big"` or `"little"`.

`n`	integer; the number of records to read. Also passed to `readBin`. **NOTE:** this argument is ignored if `fmt = "I"`, due to the way unsigned integers have to be hacked together.
`...`	additional arguments to be passed to `readBin`.

Value

a "scalar" value according to `fmt`.

Index