1 Overview

This article provides some information about Cli alongside with examples on how to use it.

2 Info

Cli provide tools to some simple S2 file manipulation. In its current form it can generate basic information about S2 file, generate CSV file from data in S2 file, select desirable part of S2 file and merge two S2 files into one s2 file. We control program with flags and their arguments.

2.1 Flags

Whenever we run Cli there are two mandatory flags, task flag and input flag, followed by additional flags. Only one task flag can be used at the same time. Flags arguments must follow directly after flag and in correct order.

2.1.1 statistics

-s provides us with some statistics. It is task flag. It has no arguments. By default it will write result to standard output. If we provide **-o** it will write result to this file instead. We will get version; total time of measurement; date, time and zone of the beginning; no. of special messages, comments, definitions, time stamps, unknown, errors, streams,packets per stream,samples per stream.

2.1.2 CSV

-r partially convert S2 file into CSV format(it is irreversible, as we keep only data and their respective times). It is task flag. Has no arguments. By default it will

- write result to standard output,
- use all handles,
- start from time 0 and end on Long.MAXVALUE = 9,223,372,036,854,775,807.

If we provide **-o** it will write result to this file instead. If we provide **-h** it will only use those handles. If we provide **-t** it will only use data from that time interval.

We will get time stamps, data, and handles on the output.

2.1.3 Cut

-c cut original S2 file and save new one. It is task flag. Has no arguments. It is mandatory to accompany this flag with flag **-o**. By default it will copy original meaning:

- use all handles,
- start from time 0 and end on Long.MAXVALUE = 9,223,372,036,854,775,807.
- keep comments, special messages,...

If we provide **-h** it will only use those handles. If we provide **-t** it will only use data from that time interval. If we provide **-d** it will only use those additional data types.

2.1.4 Merge

-m merges two original S2 files and save them into new one. It is task flag. It has mandatory argument. If argument is true it check if files correspond and if they do it will merge them on same handles, do nothing otherwise. If argument is false it will give second file new handles. It is mandatory to accompany this flag with flag **-o**.

Calculating new handles Algorithm for calculating new handle for handle i is as follows. First it will check if this handle was used before. If not it will be given that handle. If it was used before it will try handles i+1, i+2,...,i+31(mod 32 is applied every time). It will use the firt one in sequence not used before. If all of them were used before it will give it handle i(in that last case that means data from handle i from first file and data from handle i from second file will be both on same handle i in merged file).

2.1.5 Input

-i is input flag. It has 1+1 arguments. First argument is file path of main input S2 file. Second argument is optional. It is file path for secondary input file and is used only in combination with flag **-m**.

2.1.6 Output

-o is additional flag. It has 1 mandatory argument. Argument is file path of output file. In case file with same file path already exist it will be overwritten. When used with

- **-s** file extension must be **.txt**
- **-r** file extension must be **.csv**
- **-c** file extension must be **.s2**
- **-m** file extension must be **.s2**

2.1.7 Handles

-h is additional flag. It is used in combination with **-r** or **-c**. It has mandatory argument. Argument is sequence of zeros and ones. If sequence contains 1 on position **i+1** from right to left it will process handle **i**. 0 on position **i+1** from right to left means this handle will not be processed. 0 left to the leftmost 1 can be omitted. Keep in mind current version of S2 supports only handles 0-31. If this flag is omitted completely it is equivalent to using this flag with argument of sequence of 32 ones.

2.1.8 Time interval

-t is additional flag. It is used in combination with **-r** or **-c**. It has 2 mandatory arguments and one optional. First argument must be smaller than second one. The arguments represent time interval on witch data should be processed. If the third arguments is **true** it will approximate comments and special messages with last explicit time and process them accordingly. Otherwise comments and special messages will be processed as they are all on time interval.

2.1.9 Data types

-d is additional flag. It is used in combination with **-c**. It has mandatory argument. Argument is sequence of zeros and ones. 1 on position i from right to left means it will keep:

- i=1 comments
- i=2 special messages
- i=3 metadata

Current version needs metadata for correct merging therefore its strongly recommended to never delete those. Omitting this flag is equivalent to **-d 111**.

3 Examples

In all examples to follow we will assume we have two **S2** files named **file1.s2** and **file2.s2** both stored in **./files/** . Examples are independent.

3.1 example

Lets say file1.s2 stores data about EKG measurement and we want to know how long did it last. For that we call Cli as follows.

Cli -s -i ./files/file1.s2

-i is always necessary and has mandatory arguments file directory. After running the program we will get statistics of **file1.s2** on standard output.

Now we want to know the actual data for the first 30s. We want them saved in file **output1.csv** for later use:

Cli -r -i ./files/file1.s2 -o ./files/output1.csv -t 0 30

Flag **-r** tels the program we want actual data in CSV format, **-o** has necessary argument directory of file in which we will save our CSV data. There is also **-t** with 2 arguments which represent time interval.

3.2 example

First we want statistics for both files.

Cli -s -i ./files/file1.s2

Cli -s -i ./files/file2.s2

Lets say the measurement on **file2.s2** is too long. We have decided we only want part of data between 45s and 75s since the beginning. We also don't want to keep special messages.We call Cli as follows.

Cli -c -i ./files/file2.s2 -o ./files/newFile2.s2 -t 45 75 -d 101

Now we want data from **file1.s2** and **newFile2.s2** to be in the same S2 file named **merged.s2**, but in the way we will latter be able to distinguish from which file data came.

Cli -m false -i ./files/file1.s2 ./files/newFile2.s2 -o ./files/merged.s2

Let say **file1.s2** has data on two handles, 0 and 1 and **file2.s2** has data on three handles, 0, 1, 31. Data from first input file will keep their original Handles(0 and 1 in this case). For the second file it will calculate new handles. WARNING mapping depends on sequence in which struct definitions are written. 0 was already used in the first file therefore it will try to give it 1(0+1). 1 was also already used therefore it will try to give it 2(0+2). This was no used before therefore in handle 0 from second file will be handle 2 in merged file. We do the same for handle 1. 1 was used in the first file. 2 was used for 0 from second file. 3 was not used before therefore 1 will be given 3 in next file. Notice that if it would calculate new handle for 1 before 0 it would result in 1->2 0->3.Since 31 was not used before it will stay the same in merged file.

3.3 example

Let say we have 2 mesurments saved on **file1.s2**. We are particularly interested in data between 25-30 seconds and 130-205 seconds in first mesurement. First we cut each part out and save it.

Cli -c -i ./files/file1.s2 -o ./files/cut1.s2 -t 25 30 -h 0

Cli -c -i ./files/file1.s2 -o ./files cut2.s2 -t 130 205 -h 0

Now we merge them back into new file. Since they have data from same initial measurement we want them to look like it.

Cli -m true -i ./files/cut1.s2 ./files/cut2.s2 -o ./files/merged.s2