1 Overview

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

2 Introduction

Cli giveus us acces to *pipes* which can manipulate and work with S2 format. Pipes works as followed. They get valid S2 format. They do something with it. Finally they send changed but still valid S2 format to the next Pipe in line. We get desired pipeline by combaning appropriate pipes. The main purpose of Cli is to parse input arguments and based on them, it build pipeline from pipes. In its curent version every pipe can be included at most ones and in predetermed unmutable order. In most cases the order of pipes doesnt matter. In thoes that does the corect order can be achieved by reapingly calling Cli, thought for some task it may be more efficent to build a program that can directly use pipes.

In next section we will briefly explain parsing and how Cli build pipes together. In the section to follow we will explain options and functionalities of coresponding pipe along side with examples of posible arguments. The implementation of pipes will not be discused here. In the last section there will be some more realistic working examples and some examples that doesnt work.

3 Parsing

For every funcionality that we want to include into our pipeline we must include apropriate pipe. We do so by passing flag (one that represent desired pipe) to the Cli followed by arguments if any is needed. Functionality, pipe, flag and arguments together are called *Option*. Order of flags doesnt matter. It is inportant thought that if option has any arguments, this arguments are directly after flag and in corect order.

If there is problem with the input arguments (flags + arguments) or file acces, Cli will stop with a brif explanation why.

Cli stats by parsing input arguments. After Cli parses flags and their argumentsit it starts including pipes coresponding to flags. Eventhoug all the pipes are complitly mutualy compatible, Cli will look only for pipes that needs input when input is given and vice versa. It isnt mandatory to include output callback, thought without it all the work will be in vain. At the end it runs the pipeline, writes any posible errors that ocured during execution of Cli and in the if everything was successful it will print **CLI finished**. Warning: From point of Cli none existing or pointless pipeline is perfectly fine.

4 Options

The structure of following subsections will be:

- functionalities of the option
- requirements if any
- Flag and arguments in the following structure:
  - -flag
  - ○ mandatory argument 1
  - ○ mandatory argument 2
    ...
  - ○ mandatory argument n
  - ∗ optional argument
- example.

As we mentioned before Cli builds pipes in predetermined order therefore we will list options in the same order as their pipes (if they represent one) will be in pipeline.

4.1 Help

Option help doesnt actualy corespond to any pipe. It prints basic info on how to use Cli. Any aditional options are discarded.

- -h

Print help :

```
-h
```

4.2 Input

Option input also doesnt corespond to any pipe. Its purpuse is to read lines from S2 file saved on disk and give them to the next pipe in line. Directories must be valid. It has optional secondary directory.

- -i

○ primary directory

∗ secondary directory

Read file1.s2 from disk :

```
-i ./directory1/directory2/file1.s2
```

4.3 Merge

Option merge merges two S2 files into one S2 file.
It needs option input with primary and secondary directory.

- -m

merge two files provided in option input :

```
-m
```

4.4 Data

This option filters lines. If we want to discard all coments there must be 0 in argument on 1st place from right to left. If we want to discard all special messagas there must be 0 in argument on 2nd place from right to left. If we want to discard all meta data there must be 0 in argument on 3rd place from right to left. If we want to discard all packets there must be 0 in argument on 4th place from right to left. If we want to discard all unknown lines there must be 0 in argument on 5th place from right to left. Warning in curent version of Data filtering we do not allow discarding meta data. Whenever we delete line with time which is placed just before line without time, we put timestamp in front of line without time.
Requirements :

- option input.

Flag and arguments :

- -fd
  - data

Discard comments and packets :

```
-fd 00110
```

4.5 Number of lines

Limits number of Special messages to 10. Limits number of unknown lines to 10. Limits number of StreamPackets to maxStreamPackets.
Requirements :

- option input.

Flag and arguments :

- -fnl
  - maxStreamPackets

Keep only first 7 stream packets:

```
-fnl 7
```

4.6 Comments

This option filters coments based on the regex provided in argument.
Requirements :

- option input.

Flag and arguments :

- -fc
- ◦ regex

Keep only comments containing word Hello :

```
-fc (.*s|)Hello(s.*|)
```

4.7 Special messages

This option filters special message. It keeps messagas that have same who and what as in arguments and suits regex.
Requirements :

- option input.

Flag and arguments :

- -fs
- ◦ who
- ◦ what
- ◦ regex

Keep only messages from T about U containing word Hello :

```
-fs T U (.*s|)Hello(s.*|)
```

4.8 Handles

This option filters packages based on handles. To include handle #i, put 1 in position i+1 (from right to left) in the argument, to exclude it, put 0. Whenever we delete packet which is placed just before line without time, we put timestamp in front of line without time.
Requirements :

- option input.

Flag and arguments :

- -fh

◦ handles

Keep only packages with handle 0,1 and 4 :

```
-fh 10011
```

4.9 Filter time

This option filters lines with time. We keep only lines inside time interval. Start is included, end is exclusive. Start and end are relative times from the time/date in metadata. If 3rd optional argument is true we give comments and special messagas last known time and filter them accordingly. Whenever we delete line with time which is placed just before line without time, we put timestamp in front of line without time.
Requirements :

- option input OR option generate

Flag and arguments :

- -ft

◦ start [s]

◦ time [s]

∗ aproximate

Keep only lines from 5th second till 15th second :

```
-ft 5 15 true
```

4.10 Change time

This option changes timespams by adding them argument. If argument is too negative (firt line with time would have negative timestamp) it changes it so the first line with time will have timestamp 0.
Requirements :

- option input.

Flag and arguments :

- -ct

◦ delay [ns]

add timestamps 2s :

```
-ct 2E9
```

4.11 Change datetime

This option changes date and time in meta data. We can only change date time backwards. This option also changes timestamps so the absolute time doesnt change. Argument must be date, time and timezone in ISO format.
Requirements :

- option input.

Flag and arguments :

- -cdt
- ◦ dateTimeZone

change date time :

```
-cdt 2018-01-31T10:30:10.554+0100
```

4.12 Process time

This option localy changes time with least squares method.
Requirements :

- option input.

Flag and arguments :

- -p

process time :

```
-p
```

4.13 Statistics

Produces basic statistics about S2 file and saves it into file provided in argument. If argument is omitted statistics will be printed to standard output insted.
Requirements :

- option input.

Flag and arguments :

- -s
- ∗ directory

print statistics :

```
-s
```

4.14 Output

This option saves results of previous pipes. Based on extension of file provided in argument it will save in either txt,csv or S2 format. If we provide only extension it will print result on standard output in coresponding format. If we provide only name, it will use curent directory. Warning : if the directory doesnt exist, it will create one.
Requirements :

- option input OR option generate.

Flag and arguments :

- -o
- ◦ directory

save result in csv format :

```
-o ./directory/file.csv
```

4.15 Generate random

This option simulates procces of generating samples on device and sending them over wifi to android. Main idea of this option is to be able to test all S2 related tools. The data itself is constant except for the counters. It makes new S2 file in PCARD standard. In order to make it more realistics it is made pseudo-randomly within boundaries provided with arguments. It will generate file in time interval provided with option filter time and then save it to output provided with option output. In order to reapet random result we provide seed for random generator. Frequency is expected frequency of device this this program tries to simulate (must be a positive value). Frequency change is faktor of how much can frequency change during simulation. Pecenting missing is aproximate faktor of packets that should be lost. Normal delay is maximum value of delay that occuers for every packet. Big dela chance is chance that packets are significantly delayed than they all come in quick burst. Big delay is maximum value for big delay. Big delay canot happen again till previous big delay has ended. Disconects are scattered randomly across whole S2 file. They can overlap or be so small that they dont cover any package at all! When disconect ocurs device ‘stops‘ recoding, resets counters and add comment ´Disconect´ . Consequently android doesnt get any packets. If we want file without ‘wifi-flaves‘ we set percentage missing, normal delay and big delay chance to zero.
Requirements :

- option output
- option filter time

Flag and arguments :

- -g2
  - seed for random [long]
  - frequency in Hz [float] (around 128 for PCARD)
  - frequency change [0..1]
  - percentage missing [0..1]
  - normal delay in s [double]
  - big delay chance [0..1]
  - big delay in s [double]
  - number of disconects

generate S2 file :

```
-g2 1 128 0.1 0.2 0.001 0.01 0.1 3 -o ./directory/file.s2 -t 5 18
```

4.16 Generate from files

This option generates S2 PCARD file from pattern provided in the files. It will make file in time interval provided with option filter time and then save it to output provided with option output. File with frequencies must be in the next order: In every line there must be time relative to the time/date in meta data and a freaquency. They must be separated with comma. This two data represent a dot. The 'dots are connected linearly. If the first dot appears after start we will use constant frequency equal to the first frequency till firts dot. If the last dot apears before end we will "reuse" dots by interpreting times as relative to the last dot in previos cicle. All frequencies must be strictly positive. At least one frequency with time bigger that 0 must be given. File disconects must contain in every line start and end of disconect relative to the time/date in metadata separated with comma. Intervals of disconect should be ordered ascendingly. If the last interval ends before the END it will reuse disconect intervals this time relative to the last interval. If we dont want to reapet the intervals, workaround is to include interval (Long.MAX_VALUE,Long.MAX_VALUE) at the end. File with pauses should be in the same format as disconects. File with delays must contain delays in nanoseconds for every packet in its own line. If we are still generating packets after we use all delays we will go from the start. All delays must be zero or bigger.
Requirements :

- option output
- option filter time

Flag and arguments :

- -g3
  - directory of Frequencies
  - directory of Disconects
  - directory of Pauses
  - directory of Delays

generate S2 file :

```
−g3 ./Input/inputFile0.csv ./Input/inputFile1.csv
./Input/inputFile2.csv ./Input/inputFile3.csv
−ft 0 10 −o ./Output/generated3.s2
```

5 Examples

In the following examples we will assume we have 2 folders *folderIn* and *folderOut* in curent directory. In folder *folderIn* we have 2 files called *file1.s2* and *file2.s2* and no others. *folderOut* is empty.

5.1 Working

Here we will provide some working examples that can be copied and used.

Lets say we forgot what is in file *file1.s2*. We want some basic info to help us remember. Later we wont need it therefore we dont save it. Call Cli as followed:

```
Cli -i ./folderIn/file1.s2 -s
```

We realize this isnt file we wanted. We quickly look into second file. As we mentioned before the order in which options appear doesnt matter.

```
Cli -s -i ./folderIn/file2.s2
```

This is the one. We are actualy only interested in packets from **30s** till **125s**. We want the result saved in *folderOut* in new file caled *file3.s2*.

```
Cli -i ./folderIn/file2.s2 -ft 30 125 -fd 1100 -o ./folderOut/file3.s2
```

we are not interested in all the comment and special messages in this time interval, therefore we only keep comments which include word **time** and **warnings** from **recording device** that begin with **t1**. We want it in txt format so we can read it directly.

```
Cli -fc .*time.* -fs 1 w t1.* -i ./folderOut/file3.s2 -o ./folderOut/file4.txt
```

Our first coworker asked us to get him data from *file1.s2* on **handle=2** in **csv** format.

```
Cli -i ./folderIn/file1.s2 -fh 100 -o ./folderOut/coworker1.csv
```

After some work he realizes the timestaps are not as expected and asks us to fix them as much as possible. We override previous file.

```
Cli -p -i ./folderIn/file1.s2 -fh 100 -o ./folderOut/coworker1.csv
```

Our second coworker asked us to get him data from *file1.s2* and *file2.s2* in **one file**.

```
Cli -m -i ./folderIn/file2.s2 ./folderIn/file1.s2 -o ./folderOut/coworker2.s2
```

Our second coworker realizes that times are delayed for **0.001s** and dateTime in metadata should be **25.2.2017 15h 31min UTC** and asks us to fix it.

```
Cli -ct -1000000 -cdt 2017-02-25T15:31:00.000+0000
-i ./folderOut/coworker2.s2 -o ./folderOut/coworker2fixed.s2
```

For testing purposes want to create file starting at 1s and ending at 10s. From the start till the 2nd second we want constant frequency of 50. After that and till 4th second we want frequency to slowly linearly rise to 100. From 4th to 7th second we want constant frequency of 100. And till 8th second we want frequency to drop down to 80. After that we want to repeat the cicle till the end. frequencies.csv should be as followed.

```
2000000000,50
4000000000,100
7000000000,100
8000000000,80
```

Note that from 8th till 10th second frequeny will be dropping from 80 to 50. IF we would set the END after 10s the frequency would be rissing from 10th second till 12th second than from 12th second till 15th second it would be constantly 100 and so on.

We want one disconect in interval [2.9s,3.0s] and no pauses. disconects.csv chould contain

```
2900000000,3000000000
10000000000,10000000000
```

We included one disconect after 10s to make sure we dont get disconects with intervals (5.9s-6s) and (8.9s-9s). pauses.csv should be empty.

We want odd Packets with 0ns delay while even should have delay 1000ns. delays.csv chould contain

```
0
1000
```

We call CLI with next command line.

```
Cli -g3 ./Input/frequencies.csv ./Input/disconects.csv
./Input/pauses.csv ./Input/delays.csv
-ft 1 10 -o ./Output/generated3.s2
```

5.2 Misconceptions

Here we will provide some examples that dont work, not as expected or suprisingly work.

One might expect that the next command line will save statistics into file123.txt. What will actually happes is statistics will get printed on stdout and S2 file itself will be saved in txt format.

```
Cli -s -i ./folderIn/file1.s2 -o ./folderOut/file123.txt
```

The next command line is valid but useless since it dosnt save the work done.

```
Cli -i ./folderIn/file1.s2 -ft 10 20
```

The next command line is invalid since option output requires directory.

```
Cli -i ./folderIn/file1.s2 -ft 10 20 true -o
```

IF we only provide name, it will assume we want to use current directory. Therefore the next 2 command lines have the same functionality.

```
Cli -i ./folderIn/file1.s2 -ft 10 20 true -o ./file.s2
Cli -i ./folderIn/file1.s2 -ft 10 20 true -o file.s2
```

The next command line will ignore option generate because we provided input. Therefore the next 2 command lines has the same functionality.

```
Cli -i ./folderIn/file1.s2 -g 1 128 0.1 0.2 0.001 0.01 0.1 3 -o ./directory/file.s2 -t 5 18
Cli -i ./folderIn/file1.s2 -o ./directory/file.s2 -t 5 18
```

The next command line will ignore option change time because we didnt provided input. Therefore the next 2 command lines has the same functionality.

```
Cli -g2 1 128 0.1 0.2 0.001 0.01 0.1 3 -o ./directory/file.s2 -t 5 18
Cli -g2 1 128 0.1 0.2 0.001 0.01 0.1 3 -o ./directory/file.s2 -t 5 18 - ct 1000
```

The next command line will ignore all options except help. Therefore the next 2 command lines has the same functionality.

```
Cli -h -i ./folderIn/file1.s2 -o ./directory/file.s2 -t 5 18
Cli -h
```