provakil

Programming Challenge

Grep for JSON

Problem Statement

The objective is to create a tool that acts as grep[1] but for JSON log files[2]. We will call this tool *json_grep*.

The program should be invocable from the command line in the following manner:

```
json_grep PATTERN FILE
```

where *PATTERN* is the string that you need to search in every line of the *FILE*. By default (i.e without any command line flags) *json_grep* should search in both keys and values of every JSON object on each line. Apart from the default behavior of the tool, also implement the following command line flags:

Command Line Options

-k	Search only in keys of the JSON object
-v	Search only in values of the JSON object
-x	Ignore lines containing invalid JSON (see section below)
-i	Case insensitive search
-c	Show only the total number of lines matched instead of printing the matched lines
-d	Print only the lines which DO not match the pattern provided

Note: All of the above flags can be combined with each other except the flags "*-k*" and "*-v*", which are mutually exclusive.

Invalid JSON

If a line contains invalid JSON, print the following message and exit:

```
Invalid JSON on line number 23
```

Where 23 is the line number on which invalid JSON was encountered.

In case the user has passed -x option, then print no error and just continue from the next line.

[1] Grep is a very popular Unix tool for searching plain-text files for lines matching a particular pattern
[2] A JSON log file is a file where each line of the file is a valid JSON

Examples

Consider this particular log file saved as *json_sample.log*.

Consider this particular log file saved as *json_sample_err.log*.

Example 1

Input

```
./json_grep complete-request json_sample.log
```

Output

```
{"request": "/login", "verb": "GET", "host": "provakil.com", "request_id":
"1cdda501b2dc6043491df3181b0f3e08", "port": 8099, "event":
"complete-request", "time": 0.009173137, "response": 200, "start_time":
"Thu Oct 14 2021 13:14:05 GMT+0530 (India Standard Time)", "level": "info",
"message": "", "timestamp": "Thu Oct 14 2021 13:14:05 GMT+0530 (India
Standard Time)"}
```

(Because "complete-request" string is matched in the value at key "event")

Example 2

Input

```
./json_grep Java json_sample_err.log
```

Output

```
{"event": "hiring", "destination": "Provakil", "skills": "Python,
Javascript", "contact": "911123382590"}
Invalid JSON on line number 3
```

(Because "Java" appears in the "skills" key and there is invalid JSON on line 3 of the file)

Example 3

Input

```
./json_grep -x Java json_sample_err.log
```

Output

```
{"event": "hiring", "destination": "Provakil", "skills": "Python,
Javascript", "contact": "911123382590"}
```

(This did not show that there is error on line 3 because we used the "-x" flag)

Example 4

Input

```
./json_grep -v port json_sample.log
```

Output

```

```

(There will be no output because "port" does not appear in the value of any key on any line)

Example 5

Input

```
./json_grep -v 8088 json_sample.log
```

Output

```
{"request": "/forumData", "verb": "GET", "host": "provakil.com",
"request_id": "dc9be12fca2e9065799109ca97cb14a7", "port": 8088, "app":
"iOS:2.22.14", "event": "res-json", "time": 0.946825095, "level": "info",
"message": "", "timestamp": "Thu Oct 14 2021 13:14:02 GMT+0530 (India
Standard Time)"}
{"request": "/api/v2/tmUpdates/", "verb": "GET", "host": "provakil.com",
"request_id": "215be9f0ee1b76856ecc5fa8dbe50302", "port": 8088, "event":
"async-task", "time": 9.75, "task": "restrictingGroupPermissions", "level":
"info", "message": "", "timestamp": "Thu Oct 14 2021 13:14:02 GMT+0530
(India Standard Time)"}
{"since": "2021-10-14T07:32:00.723Z", "request": "/api/v2/tmUpdates/",
"verb": "GET", "host": "provakil.com", "request_id":
"215be9f0ee1b76856ecc5fa8dbe50302", "port": 8088, "event": "sync-request",
"level": "info", "message": "", "timestamp": "Thu Oct 14 2021 13:14:02
GMT+0530 (India Standard Time)"}
```

(There are 3 lines where the value of "port" key is 8088)

Example 6

Input

```
./json_grep -k 8088 json_sample.log
```

Output

```

```

(There will be no output because "8088" does not appear in any key on any line)

Example 7

Input

```
./json_grep -i Provakil json_sample_err.log
```

Output

```
{"request": "/forumData", "verb": "GET", "host": "provakil.com",
"request_id": "dc9be12fca2e9065799109ca97cb14a7", "port": 8088, "app":
"iOS:2.22.14", "event": "res-json", "time": 0.946825095, "level": "info",
"message": "", "timestamp": "Thu Oct 14 2021 13:14:02 GMT+0530 (India
Standard Time)"}
{"event": "hiring", "destination": "Provakil", "skills": "Python,
Javascript", "contact": "911123382590"}
Invalid JSON on line number 3
```

(All instances of *Provakil* are matched, irrespective of the case of the characters)

Example 8

Input

```
./json_grep -c -x -i provakil json_sample_err.log
```

Output

```
4
```

(Instead of printing all the matched lines, the tool printed only the number of lines matched because of "-c" option)

Example 9

Input

```
./json_grep -d tmUpdates json_sample.log
```

Output

```
{"request": "/forumData", "verb": "GET", "host": "provakil.com",
"request_id": "dc9be12fca2e9065799109ca97cb14a7", "port": 8088, "app":
"iOS:2.22.14", "event": "res-json", "time": 0.946825095, "level": "info",
"message": "", "timestamp": "Thu Oct 14 2021 13:14:02 GMT+0530 (India
Standard Time)"}
{"event": "hiring", "destination": "Provakil", "skills": "Python,
Javascript", "contact": "911123382590"}
{"request": "/login", "verb": "GET", "host": "provakil.com", "request_id":
"1cdda501b2dc6043491df3181b0f3e08", "port": 8099, "event":
"complete-request", "time": 0.009173137, "response": 200, "start_time":
"Thu Oct 14 2021 13:14:05 GMT+0530 (India Standard Time)", "level": "info",
"message": "", "timestamp": "Thu Oct 14 2021 13:14:05 GMT+0530 (India
Standard Time)"}
```

(Prints all the lines without the pattern *tmUpdates*)

General Instructions

- Use either Python 3 or Javascript[3] to implement the tool
- Use a hashbang on the first line of your program to make it easily executable on a Unix-like OS
- Please do not post your solution on Github, Gitlab or any other public platforms
- For parsing command line arguments:
  - Use argparse for Python. (How to)
  - Use yargs for Node.js. (You will have to use positional arguments with the default command. See link)
- Apart from the above libraries, use only standard libraries available in your chosen language
- Email us your solution with a single file (.js or .py) attached

[3] Use Node.js >= v14.0.0