Seeker

Search tool for Logs and Files

Index

1 Purpose and functions of the script

1.1 Script purposes

1. Faster alternative than using grep
2. Safe – no possibility for typo
3. Efficient way to search for phrase in big amount of files / dataset
4. Print output from log files with x lines before / after found phrase
5. Search through many exact path defined in config file
6. Very fast tool to track file by its consistency
7. Friendly presented output

1.2 Script functions

1. Search for phrase in logfiles – after choosing function and second argument as phrase, function will check all occurrence of phrase and print +/- x lines before and after found phrase, where x is defined in `config.py` file as special variable.
2. Search for phrase in files name – function will search all files and print ones with given phrase in name. Function will print only found files not older than x days, where x is variable in `config.py` file
3. search for phrase in file – function will print list of all files which have within given phrase. Function will limit search and print output only to files from last x hours, where x in special variable declared in `config.py` file.
4. print list of files from given timestamp – function will print all files from given day, limited to given timestamp.

2 System Description

Script can be run by anyone who have access rights to script. After run by user, script check argument and run function assigned to argument. Script have 4 functions listed in point 1.2.
Each function is search method to simplify finding files by names, content or trace left in log files. Function for searching by phrase can be used also as efficient way of searching anything that user want to find. Output from search in multiple declared path is presented in human friendly form.

2.1 Work algorithm

1. Script gets arguments and loads paths from `config.py` file that will be used for search
2. Script do recursive `ls` (`os.walk`) on loaded paths from `config.py` file.
3. Script filters files found in previous point to limit quantity of files to ones that are not older than x, where x is variable in `config.py` file.
4. Script will check for phrase depending on chosen function of script: log files, name of files (designed to big amount of files) or Files (limited to last x hours, where x is variable in `config.py` file). If chosen other functions to be run, script will print all files for indicated date and limited to given timestamp.
5. Regardless the function operation will be repeat for each declared path for current hostname, or only for chosen one, if function may take long time to output. If you want to see each time output for every path, you may change special variable in `config.py` file.
6. Script will print output from search and give output, then end or go back to main menu.

3   System Architecture Model

3.1   System dependency

Script is fully written in Python, based on OOP. Script requires python to be installed. Script utilize standard Python libraries: sys, os, socket, datetime, re. Script is not dependable to any external libraries. Any additional libraries do not need to be installed. Script utilize menu library, that is provided with the script and fully written by author of the script.

3.2   Files logic

Script consist 3 files in 2 folders:

`/main folder` – contains all script files

      `seeker.py` – file containing all logic behind the Seeker script.

      `config.py` – file containing all configuration data, all variables that stands for properties of the script, so script may be customized to users preferences.

      `text.py` – file containing all text for communication with user, it can be easily translated into any other language

3.3   Script logic

Script provide 4 basic functionalities that are independent and separate from each other. Each function gives user different way of searching in multiple paths. Three of functions are strictly dedicated for searching for given by user phrase, one is for limiting number of files to given hour stamp. Last functions was designed to help in circumstances when first 3 methods were not effective despite the fact that file should be found by given phrase.

Script may be lunched only for hostnames for which paths to log files or are declared within config file. Only for declared paths script will list files in declared paths and filter them by date of last modification. Each path should be assigned to name of that path, and should be declared to hostname.

Name should describe log or files path.

4 Manual – configuration of script

Usage of script is simple, but at the beginning some basic configuration have to be set. Below manual will provide instruction how to set that configuration as well as how to customize fully script for your needs.

4.1 Basic script configuration – declaring paths, interfaces and hostnames

All below configuration variables are existing within `config.py` file.

Paths can be declared for 3 dictionaries, each specified for one script function

4.1.1 path_to_files dictionary

Dictionary stores interfaces and paths to all files you want to search through. Below example how dictionary should looks like:

```
paths_to_files = {
                'hostname_01': [
                        ( 'File Folder Name 1', '/path/to/files/1/' ),
                        ( 'File Folder Name 1', '/path/to/files/2/' ),
                                                        ],
                'hostname_02': [
                        ( 'File Folder Name 1', '/path/to/files/1/' ),
                        ( 'File Folder Name 1', '/path/to/files/2/' ),
                                                        ]
                        }
```

Where:

`'hostname_01', 'hostname_02'` – name of hostname, it should be literally the same as server host name.

`'File Folder Name 1', 'File Folder Name n'`– name of the files folder. You can name it as you want to.

`/path/to/files/n/`– it should be path to place where files are stored. Path can have subfolders, all will be listed.

4.1.2 path_to_logs dictionary

Dictionary stores names to interfaces and paths to their logfiles. Below example how dictionary should looks like:

```
paths_to_logs = {
                'hostname_01': [
                        ( 'Logs Folder Name 1', '/path/to/logs/1/' ),
                        ( 'Logs Folder Name 1', '/path/to/logs/2/' ),
                        ],
                'hostname_02': [
                        ( 'Logs Folder Name 1', '/path/to/logs/1/' ),
                        ( 'Logs Folder Name 1', '/path/to/logs/2/' ),
                        ]
                        }
```

Where:

All as in previous pint 4.1.1, except path is different, dedicated not to files but log files.

4.2 Colors

First section of config.py file contains codes of colors of bash. You may set them to empty '' if you do not want colors in bash, or you are using other TUI than BASH.

4.3 Name, description and menu functions description

Second section named Menu arguments. Stores script name, description of the script and all menu options that are printed for menu and for help. Letters assigned to options (functions) should not be changed, but any other values may be changed according to other language or preferences.

4.4 Paths to menu and to script

Next section should not be change at all, because locations will be generated automatically regardless the script path.

4.5 Limitations for functions

Sections contains only one dictionary, that looks like below:

```
do_limit_to_one_choosen_interface_by_hostname = {
                                    'hostname_01': False,
                                    'hostname_02': True,
                                    }
```

Dictionary stores information about each configured hostname, which can be set only to 2 basic information: True or False. If you want any phrase in files or name of files to be searched only for one path, then set to True. If you want to search for phrase in all paths then set to False. Setting to True is good idea if some paths (or all) have really big amount of files, so output search for phrase in all paths files would take even several minutes to complete. If you want to speed up with output, set to True for current hostname and check the only one path you suppose file to be there.

4.6 Search for phrase in logs section

First 2 variables are header variables. Better do not change them.

> **Tip.**
>
> Every variable with header in name better should not be change until you want to modify output format for function. Even slightest change may cause you unreadable output from function.
>
> Each time variables that ends with `header` and `config_header` should be edited simultaneously and sum of sign digit should be equal.

`chec_logs_from_last_how_many_hours = 24 #` Assign to variable digit stands for number of hours, and if any log file last modification is older than assign value, then will be excluded from searching process. Another words, the bigger is value, the more log files will be checked and the more search process will take. Usually if there will appears need to check logs, administrator have to check log files from last several hours. But if there will appear situation that you will have to check logs from longer period of time, all you have to do is change value assigned to variable.

`counter_difference = 80 #` Value stored in this variable tells the log search algorithm, that when phrase will be found in log file 3 things:
1) how many lines print before found phrase
2) how many lines print after found phrase
3) how many lines after found phrase continue to search for next occurrence of phrase

4.7 Check name of each file

Header variables were described in previous point 4.6

`limmit_found_files_to_last_x_days = 30 #` Variable stores digit which tells the function for searching for phrase in file that how maximally old can be file, and if older then will be omitted . Main purpose of variable is to omit files that are too old. Value can be changed to any other integer digit to be most useful for script user.

4.8 Check for phrase in files

`check_fiels_from_last_how_many_hours = 48` # Variable stores digit, which tells that files older than assigned value will be excluded from opening file and searching for phrase within the file. So automatically each file older than number of hours given in variable, will be excluded from checking. The bigger value is, the more output should give (except unique data phrases) and significantly more time user of script will have to wait for output.

4.9 Print files from timestamp

Except header variable, section of function contains 3 variables:

`timestamp_pattern = [ r'\d\-\d', r'\d\-\d\d', r'\d\d\-\d\d' ]` # Variable stores as regular expression patterns, the way how timestamp hours can be passed to script function. If given hours timestamp match the pattern, then function will do its logic, otherwise will return message that it could not recognize way of providing timestamp. List should not be modified by someone who do not know python re library.

`date_pattern = [ r'\d-\d-20\d\d', r'\d-\d\d-20\d\d', r'\d\d-\d-20\d\d', r'\d\d-\d\d-20\d\d' ]` # Similar as with upper variable for timestamp, but this variable stores list of patterns for regular expressions. Expressions are patterns that checks if date entered by user fit to date that will be parsed to special format for date comparison process to exclude all files with date different to entered one. List should not be modified by someone who do not know the Python re library.

`dates_in_timestamp = [ 'today', 'yesterday', 'other date' ]` # Variable stores a list of sub options to be chosen after function was lunched. Another sub options should not be added to the list without writing functionalities to main file – `seeker.py` . Nevertheless name of list values can be change to preferred ones by user.

5 Manual – how to use script

5.1 Start command

Script can be lunched using python command to run python file:
```
/path/to/script/seeker.py
```

5.2 Arguments

Not specifying any argument, or if h argument was given, script will lunch help and print all possible arguments. Each time when lunching script, at least one argument is mandatory. Script may be lunched from prompt with arguments, or bring to menu. All arguments list:

- 0 - print output without color (grey)
- 1 - use color
- a - search for phrase in logfiles
- b - search for phrase in files names
- c - search for phrase in files
- d - print files from timestamp
- h - help
- m - menu
- q - quit (only for menu)

From upper list options a, b and d needs additional argument that stands for searched phrase.

5.3 Example of usage

Example how use script:
```
#1
/path/to/script/seeker.py c phrase_in_file
#2
/path/to/script/seeker.py d
#3
/path/to/script/seeker.py a phrase_for_log_files
#4
/path/to/script/seeker.py b part_of_file_name
```

5.4 Example outputs

```
dl d baab

Choose interface:
1. KIS 1
2. KIS 2

Choosen: 2
*****************************************************************
|      INTERFACE      |      PATH TO MESSAGES ( SEARCHED PHRASE )      |
*****************************************************************

|            KIS 2 |                  /applications/kisclient-2/messages/   |
*****************************************************************
Not found phrase in any message
```

6   Contact

If you have any questions, concerns or maybe you found bug within the code, then please do not hesitate to contact with the author of Seeker script.

E-mail: tomasz.wojcik.88@gmail.com