

Shell Interpreter using Go

```
es: 350 total, 2 running, 348 sleeping, 1430 threads
rg: 1.83, 1.68, 1.76 CPU usage: 2.12% user, 6.14% sys, 91.72% idle
ibs: 279M resident, 67M data, 39M linkedit.
ons: 126614 total, 2867M resident, 130M private, 952M shared. PhysMem: 7172M used (1910M wired), 1019M un
7G vsize, 1110M framework vsize, 6051330(0) swapins, 6405357(0) swapouts.
s: packets: 67126140/66G in, 30255042/8722M out. Disks: 10949039/151G read, 6396039/99G written.

COMMAND      %CPU TIME      #TH  #WQ  #PORTS MEM    PURG  CMPRS  PGRP  PPID  STATE  BOOSTS      %C
screencaptur 0.0 00:00.05 4    3    54    2104K 36K    0B    280   280   sleeping *0[1]      0.0
top           5.0 00:00.69 1/1  0    24    3712K+ 0B     0B    48673 48606 running  *0[1]      0.0
ocspd        0.0 00:00.00 1    0    21    596K   0B     0B    48672 1     sleeping *0[1]      0.0
com.apple.iC 0.0 00:00.10 4    3    60    2784K 0B     0B    48671 1     sleeping 0[0]      0.0
bash         0.0 00:00.14 1    0    19    2368K 0B     0B    48606 48605 sleeping *0[1]      0.0
login        0.0 00:00.05 2    1    29    1804K 0B     0B    48605 4329 sleeping *0[9]      0.0
MTLCompilerS 0.0 00:00.03 2    2    21    5276K 0B     0B    48603 1     sleeping 0[2]      0.0
AddressBookS 0.0 00:00.39 4    2    111   12M    188K   0B    48602 1     sleeping 0[6]      0.0
com.apple.We 0.9 00:03.52 6    1    194   81M-   14M    0B    48586 1     sleeping *0[943]    0.0
com.apple.We 0.0 00:10.01 9    4    206   82M    71M    0B    48578 1     sleeping *0[2651]   0.0
com.apple.We 0.0 00:03.56 6    1    193   49M    42M    0B    48573 1     sleeping *0[1353]  0.0
CFNetworkAge 0.0 00:00.08 2    2    40    1512K 0B     0B    48568 1     sleeping *0[1]      0.0
QuickLookSat 0.0 00:00.35 2    1    46    12M    6784K 0B    48567 1     sleeping 0[3]      0.0
mdworker     0.0 00:00.34 3    1    64    8976K 0B     0B    48566 1     sleeping *0[1]      0.0
mdworker     0.0 00:00.27 3    1    64    7828K 0B     0B    48563 1     sleeping *0[1]      0.0
rcd          0.0 00:00.05 2    1    51    1548K 0B     0B    48545 1     sleeping *0[1]      0.0
quicklookd   0.0 00:00.42 4    1    91    9724K 32K    0B    48541 1     sleeping 0[3]      0.0
XprotectServ 0.0 00:00.04 2    2    44    2988K 0B     0B    48332 1     sleeping 0[1]      0.0
CoreServices 0.0 00:00.24 3    1    179-  4116K- 0B     0B    48330 1     sleeping *0[1]      0.0
com.apple.Cl 0.0 00:00.20 2    1    82    2292K 0B     0B    48274 1     sleeping *123[9]    0.0
Preview      0.0 00:08.84 3    1    288   54M    1384K 0B    48273 1     sleeping *0[189]   0.0
Keynote      0.0 00:09.40 4    2    381   94M    932K   0B    48263 1     sleeping *0[92]    0.0
mdworker32   0.0 00:00.50 3    1    64    5444K 0B     0B    48246 1     sleeping *0[1]      0.0
mdworker32   0.0 00:00.56 3    1    64    5412K 0B     0B    48245 1     sleeping *0[1]      0.0
printtool    0.0 00:00.03 2    2    32    1128K 0B     0B    47927 1     sleeping 0[6]      0.0
```

Abstract

Shell interpreters or Terminal Emulators or most commonly known as command line are interfaces built in operating system like windows, unix, linux etc. to facilitate interaction between users and computer by running various commands to complete a task. Shell commands are often used to call system functions and automate tasks. Shell scripts are also prevalent for that matter.

Often, shells include features like wildcarding, piping, here documents, control structures, pipelining. C programming language has been used since its inception for creating command line tools and utility. Even the most famous shell (bash) is coded in C.

```

arachni.ruby - Ruby
File Edit View Bookmarks Settings Help
arachni.ruby
Results thus far:
16 issues have been detected.
1 DOM-based Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/ in cookie_dom input 'username' using .
2 Cross-Site Request Forgery at http://testhtml5.vulnweb.com/contact in form.
3 Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/ in cookie input 'username' using GET.
4 DOM-based Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/ in link_dom input 'url' using GET.
5 Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/comment in link input 'id' using GET.
6 Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/report in link input 'id' using GET.
7 Cross-Site Scripting (XSS) at http://testhtml5.vulnweb.com/like in link input 'id' using GET.
8 Unencrypted password form at http://testhtml5.vulnweb.com/login in form input 'password' using POST.
9 Common directory at http://testhtml5.vulnweb.com/samples/ in server.
10 Password field with auto-complete at http://testhtml5.vulnweb.com/login in form.
11 Allowed HTTP methods at http://testhtml5.vulnweb.com/ in server.
12 Interesting response at http://testhtml5.vulnweb.com/ in server.
13 Interesting response at http://testhtml5.vulnweb.com/login in server.
14 Interesting response at http://testhtml5.vulnweb.com/admin/ in server.
15 Cookie set for parent domain at http://testhtml5.vulnweb.com/ in cookie.
16 Httponly cookie at http://testhtml5.vulnweb.com/ in cookie.

Audited 14 pages.
Sent 13501 requests.
Received and analyzed 13499 responses.
In 0s-04ms
Average: 62.8627146453446 requests/second.

Currently auditing http://testhtml5.vulnweb.com/comment?id=cn081139ec72384b7ced1bdc0e228ba
Burst response time sum 15.421210000000000 seconds
Burst response count 1
Burst average response time 15.28588231703448285 requests
Burst average 7.84394987616339 requests/second
Timed-out requests 284
Original max concurrency 28
Throttled max concurrency 28

Status: Suspending
Waiting for 82 Browser cluster jobs to finish.

```

```

UNLOAD 42 No message of desired type
UNPAVIL 119 No Xenix semaphores available
ENPROGRESS 115 Operation now in progress
EXTATSBY 26 Text file busy
UNTERPRY 39 Can't read file empty
UNTRAC 10 Untraced
UNTRACES 13 Permission denied
UNTRMFE 24 Too many open files
UNTRMFE 7 Argument list too long
UNTRMFE 96 Stream pipe error
UNTRMFE 82 Attempting to link in too many shared libraries
UNTRMFE 43 Identifier removed
UNTRMFE 14 Static file handle
UNTRMFE 133 Memory page has hardware error
UNTRMFE 92 Protocol not available
UNTRMFE 27 File too large
UNTRMFE 15 Block device required
UNTRMFE 105 No buffer space available
UNTRMFE 70 Communication error on send
UNTRMFE 5 Input/output error
UNTRMFE 8 Exec format error
UNTRMFE 11 Resource temporarily unavailable
UNTRMFE 127 Key was rejected by service
UNTRMFE 127 Key has expired

SYNOPSIS
NAME
    editp-
    add-
    path/to/
    patch
    add-
    path/to/
    patch
    DESCRIPTION
        Press ENTER or type command to

```

Introduction

At the time, C was the language used for developing system level faculties ranging from programming embedded devices to big powerful computers. C is also quite good at machine level operations.

However in C, object-oriented features, namespaces and runtime type checking is not available. In systems of past, simple instructions were to be executed. That was not a problem with C which follows only imperative paradigm of programming language.

C is dangerous partly because assembly language is dangerous. We will always need some layer on top of assembly that is mostly unchecked and reflects back to how cpu instructions work. This is probably something we must live with until we have processors with the notion of type checking.

C is dangerous partly because of swaths of undefined behaviour and loose typing and because of the stupid standard library which isn't necessarily a core language problem as other libraries can be used. The standard library should be replaced with any of the sane libraries that different projects have written for themselves to avoid using libc. It's perfectly possible not to have `memcpy()` or `strcpy()` like `minefields` or `strtok()` or `strtol()` which introduce the nice invisible access to internal static storage, fixed by a re-entrant variant like `strtok_r()`, or require you to do multiple checks to determine how the function actually failed. The problem here is that if there are X standards, adding one to replace them all will make it X+1 standards.

Problem Description

Today, complex operations as well as performance are of paramount importance. Bash shell and command line tools built in that era needs to be optimised, improved and improvised to be able to

support the new type of workload and adapt to Modern data oriented and concurrent culture. Serious optimisations in C and already written scripts is demanded but not possible at such a scale. Moreover, we need to perform large computations from written scripts often now. Adding new functions and modules is often required to readily update and upgrade system and maintain security.

KeyWords

Shell script-

A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup, logging, etc. is called a wrapper.

Terminal Emulators-

A terminal emulator, terminal application, or term, is a program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term *terminal* covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window.

Reflective programming-

Reflective Programming is that you reflect upon/inspect the code at run time. Which means you are not aware of the actual name of the method or the field in the class but you can enumerate what are the fields or methods available for a given class and then invoke which ever you want to from the enumerated list.

Imperative programming-

In computer science, imperative programming is a programming paradigm that uses statements that change a program's state. In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform. Imperative programming focuses on describing *how* a program operates.

Dataframes:-

A DataFrame is the most common Structured API and simply represents a table of data with rows and columns. The list of columns and the types in those columns the schema. A simple analogy would be a spreadsheet with named columns. The fundamental difference is that while a spreadsheet sits on one computer in one specific location, a Spark DataFrame can span thousands of computers. The reason for putting the data on more than one computer should be intuitive: either the data is too large to fit on one machine or it would simply take too long to perform that computation on one machine.

Working Methodology

We read the string and parse it, and call the following functions from the given code. If the command we enter doesn't match then the sitwell function we are making would search in the dictionaries and would make suggestions to the user on the basis of the typed commands.

Related Works

The Data frames are existing and the command line tools that provide basic functionalities to the user also exist and some data analysis tools also exist which could be very helpful for the user to analyze the data he/ she has.

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Developers and those with engineering responsibilities are fond of calling terminal their home. Anyone with a Unix system has to frequently interact with the Terminal in one way or the other. And customization has always been a big part of how much the Terminal can be used to improve productivity, create unique experiences, and manage the system to improve the workflow. Command line tools are scripts, programs, and libraries that have been created with a unique purpose, typically to solve a problem that the creator of that particular tool had himself.

There are thousands of Big Data tools out there for data analysis today. Data analysis is the process of inspecting, cleaning, transforming, and modelling data with the goal of discovering useful information, suggesting conclusions, and supporting decision making.

Explanation--

1. ls - list directory contents
2. cat - concatenate files and print on the standard output
3. pwd - print name of current/working directory

-
4. cd - changes directory
 5. Sitwell - create a custom dictionary of added command and temporary functions when needed. Added support for custom scripts. Also tries suggestions for invalid commands.
 6. rm, rmdir - remove files or directories, remove empty directories
 7. touch -change file timestamps
 8. mkdir - make directories
 9. clear-clear terminals
 10. mv-move (rename) files
 11. Analyze - read csv or json file and present it description and dimensions of it.
 12. Cp - copy file
 13. Clear - clear the screen
 14. exit or q- cause normal process termination

References

- [1] https://en.wikipedia.org/wiki/Imperative_programming
- [2] [https://en.wikipedia.org/wiki/Reflection_\(computer_programming\)](https://en.wikipedia.org/wiki/Reflection_(computer_programming))
- [3] https://en.wikipedia.org/wiki/Unix_shell
- [4] https://en.wikipedia.org/wiki/Shell_script
- [5] <https://golang.org/>

[6] Moore, S., Dimoulas, C., King, D., & Chong, S. (2014, October). SHILL: A Secure Shell Scripting Language. In *OSDI* (pp. 183-199).

[7] Ousterhout, J. K. (1998). Scripting: Higher level programming for the 21st century. *Computer*, 31(3), 23-30.

[8] Gavrilov, A., & Novitskaya, J. (2001). The toolkit for development of hybrid expert systems. In *Science and Technology, 2001. KORUS'01. Proceedings. The Fifth Russian-Korean International Symposium on* (Vol. 1, pp. 73-75). IEEE.

[9] Breuker, J., Winkels, R., & Sandberg, J. (1987, August). A shell for intelligent help systems. In *Proceedings of the 10th international joint conference on Artificial intelligence-Volume 1* (pp. 167-173). Morgan Kaufmann Publishers Inc..