OPERATING SYSTEMS

COMMAND LINE INTERPRETER FOR UNIX IN JAVA

GROUP 3 - PROJECT

INSTRUCTOR: DR.KARNE

GROUP MEMBERS:
The Command Line Interpreter For Unix In Java Language

Objective:

The objective of the project is to simulate a command line interpreter program which functions same as a command shell in Unix and also to allow the user to run his own commands by aliasing it with some standard commands so he can have the convenience to interpret his own commands and also to allow the user to view the commands he interpreted and clean them if needed and store them for the future use.

Introduction:

Every OS has to have some kind of mechanism by which a user can tell the OS what it should be doing, and for the OS to ask the user for instructions and to report problems. In the old days, the computer console was a hardware terminal that was hardwired into the computer, then used by the computer operator to direct the OS, and for the OS to print messages to the operator. Timesharing changed all of that. Now every interactive user needs a console to their virtual computer. When you log into a computer, it must provide you with some kind of interface -- the shell. The shell interface does a vital role in the operating system. Command line interpreter is also a Kind of shell which takes the commands from the console and interpret them one line at a time.

Command Interpreter is one of the most important systems programs for an operating system. Its function is simple: to get the next command statement and execute it. This command line interpreter is designated to facilitate the user to define his own commands and leave the task to the command line interpreter to interpret them with the standard command. Command line interpreters have the advantage that the user may issue a lot of commands in a
very terse and efficient way.

This project is to develop a command line interpreter for UNIX system. It reads a line of text from the user and interprets the content in the Unix system. It provides convenience and gives useful information to user on how to use commands correctly. The command line interpreter also creates a data file to record all aliased User commands for later uses. Users can use unalias command to remove one of alias commands. Users can also provide cleanUserCommands to clean all the records of alias commands. At the same time, users can view all records of alias commands using viewUserMapping commands.

**Features of This Command Line Interpreter:**

1. **making the own commands of the user instead of standard commands**
   
The user can make his own commands and make them function as some standard commands. This can be done by using alias command.

2. **Storing the user commands for future use**
   
The command line interpreter allows users to create their own commands and store the commands for the future use. By taking advantage of this feature, users don’t need to retype the commands they created. The program will write the commands users created into a file whenever the user exits from the program. The main purpose of this part is to reduce users inconveniences when they use it.

3. **Avoid reinterpreting standard commands**
   
   Avoid special situation – reinterpreting standard commands. The users should not be allowed to alias the standard commands with some other commands to keep the correct functionality of the commands. For example a user can be allowed to alias a command say abc as standard command(say ls) but he cannot alias standard command(ls) to some command abc.

4. **Cleaning out all the interpreted user commands**
The user can also clean all the previously interpreted (aliased) commands so he can start from the beginning again. This can be done by the command cleanUserCommands.

5. Allowing the user to view the interpreted command.
The user also has the convenience of viewing the Commands he interpreted so that he can keep a track of it. This can be done by the command viewUserMapping.

6. Unaliasing a specified user interpreted command
The user can also unalias a specified user command if he doesn't want to use it any longer. This can be done by the command unalias.

FUNCTIONALITY OF EACH CLASS IN THE PROGRAM:

1. Class CommandLineInterpreter:
   * reads old records from the aliasTable.data at beginning for uses.

   * when it executes the command it first reads user input, compare it with the command alias table. If the command is already in the table or if it is a standard command, then it executes the command. If the command does not exit then it reports error.

   * takes various appropriate actions for user inputs like: alias cleanUserCommands viewUserMapping Unalias exit

   * Provides convenience and gives useful information to user on how to use commands correctly.

2. Class CommandExecute:

This class implements runnable interface in java. There are two methods run(), takeAction().

* run() is to run the command and output the running results.
* takeAction() is to give some useful information if the run thread happens error.

3. **Class Alias:**

The function of this class is to compare user input with the hashtable. Inserting user commands, deleting user commands, and writing commands to an output file. It has six methods.

* report() returns the corresponding standard command if the command is in the hashtable table and returns itself if the command is not in the hashtable table.
* insert() adds the alias and its value to the table.
* cleanCommands() clears all the records in the table.
* remove() removes one key from the table.
* writetofile() outputs all records in the table to the file aliasTable.data.
* userCommands() outputs user commands in the format of "User Command Standard Command".

4. **Class CommandTest:**

The function of this class is to create a new process to test the input to avoid reinterpret user commands.

* Creates a new process to test the input
* Reports error when user tries to alias standard commands or already existing commands and exit the command line interpreter program.
Work Process (Design and Development):

Since we got the project assigned as the course requirement and formed the group, everyone in our group made extraordinary efforts to make this project a very successful one. Every member of our project worked in part of the code to make it to run and add various functionalities to it. The work was evenly distributed at each step among All 5 members. Each member of the group initially came up with some idea as to what kind of functionality we can implement in the command line interpreter and we came up with lot of functionalities. Finally we implemented every functionality we thought of and came up with this final version.

Summary:

In this project, we have introduced our project proposal and have gone through all the steps of designing the code. We actually even thought of implementing a network interface to our command line interpreter so that the program can be connected to network and also apply graphical user interface to the program but due to time constraints we dropped the idea of doing that. But still we came up with lot of functionalities for providing convenience to the user. This idea can be extended to provide very effective command line interpreter which is useful for marketing purposes.
User guide:

1. Use “java commandLineInterpreter” to start the command line interpreter program in the triton.

2. Use “alias new command=built-in command” to apply a new command to fulfill the same function of the built-in command.

3. Use “&” to separate more than one “alias new command=built-in command” command to fulfill several alias commands at the same time.

4. Use “Unalias new command” to remove one made alias command.

5. Use “cleanUserCommands” to clean all records of the made alias commands.

6. Use “viewUserMapping” to view all made alias command records.

7. Use the made new command to complete the function of the built-in command as well as the built-in command.

8. More than one built-in commands can be input in the same line using “&” to separate them and all the commands can be executed and output the running results at the same time.

9. Use “exit” to exit the command line interpreter program and go into the UNIX operation system environment.
Results

1. alias command correct, so new command can be used as if it were the alias built-in command:

$alias ef=date
$date
Fri Dec 6 18:33:26 EST 2002
$ef
Fri Dec 6 18:33:28 EST 2002

$alias mn=whoami
$whoami
lpu1
$mn
lpu1

$alias ab=ls
$ls
CommandExecute.class
CommandExecute.java
CommandTest.class
CommandTest.java
WS_FTP.LOG
alias.class
alias.java
aliasTable.data
c:\student1
commandLineInterpreter.class
commandLineInterpreter.java
jj.java
make.class
make.java
$ab
CommandExecute.class
CommandExecute.java
CommandTest.class
CommandTest.java
WS_FTP_LOG
alias.class
alias.java
aliasTable.data
c:\student1
commandLineInterpreter.class
commandLineInterpreter.java
jj.java
make.class
make.java

$alias hh=finger
$hh

Login       Name         TTY   Idle   When     Where
----------   -----------   ----   -----   ------     --------
dcicco1     Dina Ciccone  pts/12  18d     Mon 12:25  ppp-225.dialup.umbc.
sgill1      Sam Gill      pts/26  2d      Wed 14:44  triton.towson.edu
dguent1     David Guenther pts/28  3:40    Fri 11:03  tow32dhcp240.towson0
webssh      SSH Login User pts/25  2d      Wed 14:43  bgp01562194bgs.gambr
lpu1        Lixin Pu      pts/32  35     Fri 17:48  136.160.159.118
sam         Sam Houston    pts/33  9d      Tue 11:21  sandbar.towson.edu
lpu1        Lixin Pu      pts/47  9d      Tue 11:21  sandbar.towson.edu
vcei        Project for Teong-Ta pts/48  57     Fri 13:51  tow40dhcp719.towson0
root        Super-User    pts/59  2d      Wed 16:04  ford.towson.edu
vicedil     Project for Teong-Ta pts/62 1:22    Fri 15:14  tow40dhcp719.towson0

$alias gg=df
$gg

Filesystem 1k-blocks  Used   Available  Use% Mounted on
----------  -------   ----   --------  ----  ------
/dev/md/dsk/d0 4032504  1290943  2701236  32%   /
/dev/md/dsk/d3 4032504  222457  3769722  6%   /var
swap        4851520   24      4851496  0%   /var/run
swap        4882104  30608   4851496  1%   /tmp
$alias jj=du
$jj
1  ./c:\student1
38  .

$alias kk=id
$kk
uid=12231(lpu1) gid=5000(students) groups=5000(students)

$alias aa=man
$aa
usage:  man [-] [-adFlrt] [-M path] [-T macro-package ] [-s section ] name ...
       man [-M path] -k keyword ...
       man [-M path] -f file ...

$alias pp=pwd
$pp
/export/home/st1/l/p/u/lpu1/cosc519/new
```bash
$alias qq=cal
$qq
  December 2002
  1  2  3  4  5  6  7
  8  9 10 11 12 13 14
 15 16 17 18 19 20 21
 22 23 24 25 26 27 28
 29 30 31
$alias rr=who
$rr
  dcicco1  pts/12  Nov 18 12:25  (ppp-225.dialup.umbc.edu)
  sqill4   pts/26  Dec  4 14:44  (triton.towson.edu)
  dguent1  pts/28  Dec  6 11:03  (tow32dhcp240.towson01.md.comcast.net)
  webssh   pts/25  Dec  4 14:43  (bgp01562194bgs.gambri01.md.comcast.net)
  lpu1     pts/32  Dec  6 17:48  (136.160.159.118)
  sam      pts/33  Nov 26 11:21  (sandbar.towson.edu)
  lpu1     pts/47  Dec  6 18:30  (136.160.159.118)
  vcei     pts/48  Dec  6 13:51  (tow40dhcp719.towson01.md.comcast.net)
  root     pts/59  Dec  4 16:04  (ford.towson.edu)
  vicdil   pts/62  Dec  6 15:14  (tow40dhcp719.towson01.md.comcast.net)
$alias ll=ps
$ll
   PID   TTY      TIME  CMD
  28990 pts/47  0:01  java
  28957 pts/47  0:00  bash
$alias uu=hostname
$uu
  triton
```
$alias yy=at
$yy
usage: at [-c|-k|-s] [-m] [-f file] [-p project] [-q queue] -t time
      at [-c|-k|-s] [-m] [-f file] [-p project] [-q queue] timespec
      at -l [-p project] [-q queue] [at_job_id...]
      at -r at_job_id ...

2. **View all alias records users have made. So ef date should be in the table:**

$viewUserMapping
<table>
<thead>
<tr>
<th>User Command</th>
<th>Standard command</th>
</tr>
</thead>
<tbody>
<tr>
<td>gg</td>
<td>df</td>
</tr>
<tr>
<td>jj</td>
<td>du</td>
</tr>
<tr>
<td>mn</td>
<td>whoami</td>
</tr>
<tr>
<td>pp</td>
<td>pwd</td>
</tr>
<tr>
<td>yy</td>
<td>at</td>
</tr>
<tr>
<td>ll</td>
<td>ps</td>
</tr>
<tr>
<td>rr</td>
<td>who</td>
</tr>
<tr>
<td>uu</td>
<td>hostname</td>
</tr>
<tr>
<td>ef</td>
<td>date</td>
</tr>
<tr>
<td>hh</td>
<td>finger</td>
</tr>
<tr>
<td>kk</td>
<td>id</td>
</tr>
<tr>
<td>qq</td>
<td>cal</td>
</tr>
<tr>
<td>ab</td>
<td>ls</td>
</tr>
<tr>
<td>aa</td>
<td>man</td>
</tr>
</tbody>
</table>

3. **Use unalias command to delete ef command:**

$unalias ef
$viewUserMapping
<table>
<thead>
<tr>
<th>User Command</th>
<th>Standard command</th>
</tr>
</thead>
<tbody>
<tr>
<td>gg</td>
<td>df</td>
</tr>
<tr>
<td>jj</td>
<td>du</td>
</tr>
<tr>
<td>mn</td>
<td>whoami</td>
</tr>
<tr>
<td>pp</td>
<td>pwd</td>
</tr>
</tbody>
</table>
yy       at
ll       ps
rr       who
uu       hostname
hh       finger
kk       id
aq       paste
qq       cal
ab       ls
aa       man
$ef
Error executing thread
You can use 'ls /bin' to search any command wanted.
You can use 'man command' to learn use the command.
You can use 'cleanUserCommands' to delete all user commands
You can use viewUserMapping to view all mapping of user commands to standard commands
You can use 'make new command=build-in command' to apply your familiar command for the same function.
You can also use 'unalias User command' to delete make command.
You can use 'exit' to exit this program.
$viewUerrMapping
Error executing thread
You can use 'ls /bin' to search any command wanted.
You can use 'man command' to learn use the command.
You can use 'cleanUserCommands' to delete all user commands
You can use viewUserMapping to view all mapping of user commands to standard commands
You can use 'make new command=build-in command' to apply your familiar command for the same function.
You can also use 'unalias User command' to delete make command.
You can use 'exit' to exit this program.

4. Avoid re-interpreter standard commands and gives direction on how to use alias command correctly if alias command is not correct:

$alias date=ef
$Error trying to interpret standard command
or already existing aliased command
You can only use 'alias new command=build-in command'.
*****Exiting the program!*****
[~/cosc519/new]

5. **Give useful information and provides convenience to users:**

$c
Error executing thread
You can use 'ls /bin' to search any command wanted.
You can use 'man command' to learn use the command.
You can use 'cleanUserCommands' to delete all user commands
You can use viewUserMapping to view all mapping of user commands to standard commands
You can use 'make new command=build-in command' to apply your familiar command for the same function.
You can also use 'unalias User command' to delete make command.
You can use 'exit' to exit this program.

6. Use **cleanUserCommands to clean all records of alias commands:**

$viewUserMapping
User Command   Standard command
gg            df
jj            du
mn            whoami
pp            pwd
yy            at
ll            ps
rr            who
uu            hostname
hh            finger
kk            id
aq            paste
qq            cal
ab            ls
aa            man
$cleanUserCommands
$viewUserMapping
User Command   Standard command