CSE 2005 – OPERATING SYSTEMS

J- Component

Slot: L55+56

Faculty In charge: Prof. SELVAKUMAR K.

Project Title: Expect-Lite Program using Bash Script

by

Name	Registration Number
Sameeran Bandishti	17BCE0267
Abhisu Jain	17BCE0777
Swati Singhvi	17BCE2037

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ▪ CHENNAI
www.vit.ac.in

CERTIFICATE

This is to certify that the project work is a bona fide work, submitted by Sameeran Bandishti - 17BCE0267, Abhisu Jain – 17BCE0777, Swati Singhvi – 17BCE2037 under the supervision of Prof. Selvakumar K. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Abstract:

A command to create custom command line commands that can be webbed together to create a personalized command line interface. The command will allow users to automate tasks and chain existing commands using bash scripts which can be executed by a single word. We will also be creating a customized command line interface to exhibit the capabilities of this project.

Keywords:

Command Line Interface, Expect-Lite, Bash scripts, Task automation, custom commands

Introduction:

A lot of the terminal commands that we use are chained. Meaning, to achieve a particular output, we go through typing a few separate commands that could very well be combined into a single command. Effectively automating tasks for the user.

For example:

When using a python IDE such as PyCharm, it is recommended to use a virtual environment to prevent errors due to PyCharm's updates. Every time PyCharm is opened, the user has to manually create a new environment. Also, if the project files are present on a storage medium that hasn't been mounted yet, the disk needs to be mounted and PyCharm has to be given access to the mounted disk.

The complete process specified above can be automated and linked to a single command.

The major issue with creating said commands is that the process is fairly complex and the aim of this project is to reduce that complexity by effectively automating the task of automating tasks.

Related Works:

We have borrowed the aim of this project from **Expect-Lite**.

"Expect-lite is a mature, quick and easy command line automation tool. Written in expect, it is designed to directly map an interactive terminal session into an automation script. expect-lite scripts use special character(s) at the beginning of each line to indicate the action. Basic expect-lite scripts can be created by simply cutting and pasting text from a terminal window into a script, and adding '>' '<' characters. No knowledge of expect is required!"

Unlike **Expect-Lite**, our project will not use 'expect' language. Also, the user will be asked to code the bash scripts on a text editor instead of copying and pasting the terminal commands.

Proposed Methodology/ Algorithms:

The following is the algorithm to create a user-defined terminal command:

1. Start
2. Open text editor
3. Code the desired bash script
4. Save it
5. Ensure that the user has full access to the location at which the file is saved
6. Ensure that name of the file is the same that you want your command to be
7. Open the properties of the text file and make it executable
8. Get root permissions on the terminal
9. Use mv command on the terminal to move this file to /usr/bin
10. Close the current terminal window and open a new one
11. Stop

The following is the algorithm of the proposed terminal command:

1. Start
2. Wait for command to be called
3. Ask user for command name
4. Open text editor and allow user to code the bash script
5. Save the code with the entered command name /tmp directory
6. Run the bash script to ensure there is no error
7. Change the permissions of the file and make it executable
8. Move the file to usr/bin
9. Close the current terminal window and open a new one
10. Stop

The following is the algorithm of the command 3Term:

1. Start
2. Open first terminal window
3. Open second terminal window
4. Open third terminal window
5. Stop

The following is the algorithm of the command mountD:

1. Start
2. Get root permissions
3. Input alias name for the new drive
4. make a new directory in media/username/aliasname
5. mount the drive onto the above alias

6. Stop

The following is the algorithm of the command Mig:

1. Start
2. Assess required Django migrations
3. Perform required Django migrations
4. Run Django Testing Server locally
5. Stop

The following is the algorithm of the command NetSec:

1. Start
2. Open 3 new terminals
3. Opens www.overthewire.org
4. Open google keep
5. Stop

The following is the algorithm of the command FrontEndDev:

1. Start
2. Mount hard disk
3. Open www.w3schools.com, www.uigradients.com, www.fontawesome.io, www.getbootstrap.com
4. Open gitkraken
5. Open atom
6. Stop

The following is the algorithm of the command BackEndDev:

1. Start
2. Mount hard disk
3. Open mongo DB
4. Open PyCharm professional
5. Stop

The following is the algorithm of the command PhotoEditing:

1. Start
2. Mount disk drive
3. Opens photoshop
4. Opens illustrator
5. Open photo gallery
6. Enters CreativeMode
7. Stop

The following is the algorithm of the command change_background:

1. Start
2. Check if specified file path exists or not
3. If it does then change the background to specified image path
4. Else print “error” statement
5. Stop

The following is the algorithm of the command CasualMode:

1. Start
2. Open Spotify
3. Open WhatsApp for desktop
4. Change desktop background
5. Stop

The following is the algorithm of the command CreativeMode:

1. Start
2. Open Spotify
3. Open gimp
4. Change desktop background
5. Stop

The following is the algorithm of the command OfficeMode:

1. Start
2. Open mailspring
3. Open WhatsApp for desktop
4. Open google drive, docs and calendar
5. Change desktop background
6. mount drive
7. Stop

CODE FOR MAKING A COMMAND

```
#!/bin/bash

cd Desktop
echo Please Enter the Name of the Command
read name
nano $name.sh
sudo chmod +x $name.sh
sudo chown sameeran $name.sh
echo "We will now test the command"
./$name.sh
check=0
while [ $check = 0 ]
do
echo "Please type 'Yes' if it worked as expected"
read status
if [ $status = "Yes" ]
then
check=1
elif [ $status = "No" ]
then
check=0
nano $name.sh
sudo chmod +x $name.sh
echo "We will now test the command"
./$name.sh
fi
done
echo Please fill in the following details about the
command
echo Does the command have a specific syntax ?
read ans
if [ $ans = "Yes" ]
then
echo Please enter the syntax
read syntax
fi
echo Please enter what the function performs
read performs
cd /usr/bin
echo "echo Command - $name" >> MyCommands
if [ $ans = "Yes" ]
then
echo "echo Syntax - $syntax" >> MyCommands
fi
echo "echo Performs - " >> MyCommands
echo "echo $performs" >> MyCommands


cd
cd Desktop
mv $name.sh $name
sudo mv $name /usr/bin
```

Result and discussion:

Command automation is definitely a step in the right direction in both bundling and redefining existing redundant command chains and also for ease of use for the user when he or she uses generally chained commands. The ‘makecommand’ command, which is the product of this project, is very helpful in reducing the hastle involved in making a user defined command.

A very important part of making commands is documentation. The ‘makecommand’ makes this task easier by prompting the user to input information about the command and then reformatting this information into a manual page format and appending it to the output of a command named ‘MyCommands’.

Conclusion and Future Work:

In this early prototype of the project, the user needs to code his bash scripts on his own. But in future iterations, we aim to make this part of the process automated as well. We will offer the user a drag and drop coding environment where users without any previous knowledge of bash script would be able to make their own commands as well.

Our current prototype also has no debugger for the bash script. We ask the user to manually recheck the code if an error is detected. But we will also be working on a automated debugger to help the user correct the bash scripts more easily.

The Linux Terminal is one the most useful and powerful tools that a developer possesses. It's applications are in every field pertaining to computers. The problem is that most of the commands that the terminal uses are old and sometimes redundant. Another problem is that sometimes, even small tasks require multiple lines of commands to execute. Developers are forced to write multiple lines of codes over and over again. These redundancies and low efficiency is what we have tried to tackle with this project.

References:

https://www.geeksforgeeks.org/
https://expect-lite.sourceforge.net/