Customized C shell built on ubuntu with alias functionality

B.U.Naveen Raj
Amrita School Of Computing
Amrita ViswaVidhyapeetham
Bengaluru, India
naveenrajbu208@gmail.com

Utpal Raj KB
Amrita School Of Computing
Amrita ViswaVidhyapeetham
Bengaluru, India
utpalraj137@gmail.com

***Abstract*— This project aims to develop a user-defined interactive shell designed over Linux system. It aims at facilitating seamless process creation and management. By using C programming, it offers robust mechanisms for analysing commands, executing processes, and managing input/output. With support for system commands and user-defined executables, Terminal provides users with a customizable environment to extend functionality according to their needs. Its interactivity and responsiveness enhance user experience, fostering efficient system navigation and program execution. We have developed alias command which is used to change current inbuilt Ubuntu command into user defined command. Ultimately, Terminal empowers Ubuntu Linux users with a versatile and dynamic command-line interface, promoting productivity and streamlining workflows in the Linux environment. This literature survey explores the design and implementation of a customized C Shell tailored for the Ubuntu operating system. The shell incorporates essential commands such as background processes, command history, directory manipulation, process discovery, execution control, job management, list directory contents, mail, make file support, process information retrieval, print working directory, and signal handling. The survey discusses the rationale behind the development of such a shell, its design considerations, implementation methodology, and potential applications.**

Keywords—shell,background process,make,C Shell

This research paper unveils a novel integration into the Unix ecosystem, featuring a distinctive addition: the alias command. While upholding essential functionalities such as command execution and process management, this shell introduces a paradigm shift by enabling users to redefine commands, thereby enhancing productivity and customization within their workflow. With a user-centric design emphasizing intuitive interaction and robust performance, notable features include command auto-completion and syntax highlighting. Moreover, users benefit from a plethora of customization options, ranging from prompt configuration to comprehensive shell scripting support. Engineered for reliability and efficiency, this shell aims to foster a collaborative community, inviting iterative enhancements to refine the Ubuntu command-line experience, positioning it as a significant contribution to the field.

RELATED WORK

In [1], The paper "Possibilistic C-Shell Clustering with Inter-Cluster Constraints" by Tsaipei Wang and James M. Keller discusses the incorporation of constraints into the possibilistic c-shell clustering (PCSC) algorithm to locate clusters with specific properties. The authors focus on the use of an additional constraint term to encourage circular prototypes with similar radii. This is achieved by adding a term to the cost function that gives differences in radii. The update equations for the prototype parameters are derived by setting the derivatives of the objective function to zero and are solved using Newton's method. The paper presents simulation results and an application to real-world image analysis, demonstrating the effectiveness of the modified algorithm in locating clusters with similar radii in noisy data

In [2]. The paper "Template based fuzzy c-shells clustering algorithm and its fast implementation" introduces a template-based fuzzy c-shells (TBFCS) clustering algorithm that improves clustering performance for any-shell type clusters, unlike existing algorithms limited to specific shapes. The TBFCS algorithm uses binary images to represent clustering prototypes and incorporates a distance metric to evaluate the similarity between data points and prototypes. Additionally, the paper discusses a fast implementation of the TBFCS algorithm using Genetic Algorithms, decomposing & reconstructing templates, and distance transform techniques to enhance computational efficiency. Experimental results demonstrate the algorithm's effectiveness in accurately clustering complex shell-shaped data sets, surpassing traditional fuzzy c-shells algorithms in identifying curved boundaries in images

In [3]. The paper presents a flexible possibilistic c-template shell clustering method that allows for greater deformation of the cluster prototypes compared to the original possibilistic c-template (PCT) algorithm. The key innovations are:

The template is divided into multiple parts, each with its own set of transform parameters (translation, scaling, rotation). A fuzzification factor controls the degree of blending between the template parts.

The update equations for the transform parameters are modified to weight the contributions of data points based on their proximity to each template part.

A post-processing step is introduced to add vertices and edges to the prototypes to better capture their deformed shapes, with the degree of deformation controlled by the fuzzification factor.

Experimental results demonstrate the ability of the proposed method to better fit complex, deformed cluster shapes compared to the original PCT algorithm. The increased flexibility comes at the cost of more complex prototype representations and optimization.

In[4]. The study evaluates the performance of LabView on Linux Ubuntu and Windows XP operating systems, focusing on data application measurements. It compares digital signal acquisition and processing performance using

LabView™ software on Ubuntu 8.4 and Windows XP. The research demonstrates stable and effective system operation on both OSs, measuring metrics like throughput, execution time, and CPU usage. The study emphasizes the importance of OS performance in data analysis and highlights the graphical programming environment of LabView™. Results show Linux outperforming Windows in file system operations and FFT applications, with Linux exhibiting better precision and faster performance. The analysis suggests potential for performance improvement through OS optimization.
In[5] ShellFusion is a system designed to generate answers for shell programming tasks by leveraging knowledge fusion. It uses a combination of techniques including preprocessing, word embedding, and IDF (Inverse Document Frequency) to improve the efficiency of retrieving similar questions for a query. The system consists of two components: Offline Processing and Online Answer Generation. Offline Processing extracts shell-related questions from Q&A communities and mines knowledge of shell commands from Ubuntu Manuals and TLDR tutorials. the most similar TLDR task-script pair, the top-3 similar questions with accepted scripts that use the command, and the explanations about the options of the command used in the scripts

IMPLEMENTATION

bg: This command is used to move a suspended or background process into the foreground, allowing it to continue executing.

history: Displays a list of previously executed commands, typically with their corresponding line numbers.

cd: Changes the current working directory to the specified directory.

discover: This command could be implemented to search for specific files or directories within the system.

execute: Executes a specified command or script file.

fg: Brings a background process into the foreground, allowing user interaction with it.

jobs: Lists all currently active jobs, including both background and suspended processes.

ls: Lists the contents of a directory. Options can be added to customize the output, such as showing hidden files or sorting by different criteria.

makefile: This command could execute a Makefile, typically used for compiling and building software projects.

pinfo: Retrieves and displays information about a specific process, such as its ID, parent process ID, CPU and memory usage, etc.

pwd: Prints the current working directory.

sig: Sends a signal to a specified process. This could be used for various purposes, such as terminating a process gracefully or handling specific events.

alias command_to_be_changed_to=’command’:This command replaces existing command to user defined command

APLICATION AND FUTURE SCOPE

Here, the survey explores potential applications of the customized C Shell beyond its basic functionalities. It discusses how the shell could be extended or adapted for specific use cases, such as system administration, software development, or educational purposes. Additionally, it outlines avenues for future research and development to enhance the shell's capabilities further .Auto completions, Artificial Intelligence to terminal with customization are key future scope of this project.

CONCLUSION

THE CONCLUSION SUMMARIZES THE KEY FINDINGS OF THE LITERATURE SURVEY AND EMPHASIZES THE IMPORTANCE OF DEVELOPING CUSTOMIZED SHELLS TO MEET THE DIVERSE NEEDS OF USERS AND SYSTEM ADMINISTRATORS. IT REITERATES THE SIGNIFICANCE OF THE PROJECT'S OBJECTIVES AND HIGHLIGHTS THE CONTRIBUTIONS MADE IN ADVANCING SHELL TECHNOLOGY ON THE UBUNTU PLATFORM.

REFERENCES

[1] Tsaipei Wang, James M. Keller; Possibilistic C-Shell Clustering with Inter-Cluster Constraints; 2006 IEEE International Conference on Systems, Man, and Cybernetics October 8-11, 2006, Taipei, Taiwan
[2] Gao Xin-bo, Xie Wei-xin, Litl Jian-zhuang, and Li Jie; Template based fuzzy c-shells clustering algorithm and its fast implementation;
[3] Tsaipei Wang;A Flexible Possibilistic C-Template Shell Clustering Method with Adjustable Degree of Deformation
[4] M. G. D’Elia,V. Paciello; Performance Evaluation of LabView on Linux;Ubuntu and Window XP operating systems 19th Telecommunications forum TELFOR 2011 Serbia, Belgrade, November 22-24, 2011.
[5] Neng zang, Chao lin, Xin xia;Shell fushion answer generation for Shell programming tasks via knowledge fusion; 2022 IEEE/ACM 2022 IEEE/ACM 44th International Conference on Software Engineering