HUMAN-COMPUTER INTERACTION ENHANCEMENT FOR LINUX CLI APPLICATION USING TELEGRAM BOT PIPING

Herlambang Rafli Wicaksono[1)], Fadel Azzahra[2)], Hermawan Setiawan[3)]

[1-3)] Software Cryptography Engineer, Politeknik Siber dan Sandi Negara

Correspondence Author: herlambang.rafli@student.poltekssn.ac.id

Article Info :

Article History :

Received :
05 July 2024
Revised :
20 July 2024
Accepted :
06 August 2024
Available Online :
28 August 2024

Keyword :

***human-computer interaction* (HCI), *linux command line interface* (CLI), telegram bot.**

ABSTRACT

This paper proposes an innovative method to enhance Human-Computer Interaction (HCI) for Linux Command Line Interface (CLI) applications by integrating Telegram bot piping. Despite the power of CLI, its complexity often deters users, particularly those less experienced with technology. We introduce a solution leveraging Telegram bots to provide a more intuitive and accessible interface for interacting with CLI tools. Users can issue commands via Telegram, receiving real-time feedback and executing tasks without extensive CLI knowledge. Our study includes design, implementation, and evaluation phases, demonstrating high user satisfaction with the Telegram bot integrated CLI applications. The approach significantly improves usability, reduces the learning curve, and broadens the accessibility of Linux systems. The user feedback indicated high satisfaction with the Telegram bot integrated CLI applications, with users finding it effective, user-friendly, and preferable over traditional CLI interfaces with user satisfaction score of 95.3%

1. INTRODUCTION

Human-Computer Interaction (HCI) is a multidisciplinary field that examines the design and use of computer technology, focusing particularly on the interfaces between people (users) and computers. HCI seeks to create efficient, effective, and satisfying user experiences by improving the usability and accessibility of technology [1], [2]. Over the decades, the evolution of HCI has seen a shift from text-based interfaces to graphical user interfaces (GUIs) and, more recently, to natural user interfaces (NUIs) that utilize touch, gesture, and voice. Despite these advances, Command Line Interfaces (CLIs) remain a fundamental tool in many technical domains, including system administration, software development, and data analysis due to their powerful, flexible, and scriptable nature.

One area that remains challenging for many users, especially those less familiar with technology, is the Command Line Interface (CLI) in Linux [3]. Despite its powerful capabilities, the CLI's steep learning curve can be a barrier to entry for many [4]. This paper explores an innovative approach to mitigating this challenge by enhancing HCI for Linux CLI applications through the integration of Telegram bot piping.

ISSN Print : 1979-7141
ISSN Online : 2541-1942

Telegram bots have emerged as versatile tools capable of interacting with users in natural language, offering a more intuitive and accessible interface. By leveraging these bots, we propose a system where users can interact with Linux CLI applications via Telegram, effectively bridging the gap between complex command-line operations and user-friendly interfaces. This method not only simplifies the interaction process but also broadens the accessibility of Linux systems to a wider audience [5], [6], [7].

The integration of Telegram bots with Linux CLI applications involves creating a pipeline where commands and their outputs are exchanged between the CLI and the Telegram bot. This setup allows users to issue commands in a conversational manner, receive real-time feedback, and even automate repetitive tasks without needing extensive knowledge of CLI syntax and operations.

This paper delves into the design, implementation, and evaluation of this system. We discuss the technical challenges encountered, such as ensuring secure communication between the Telegram bot and the Linux system, handling asynchronous command execution, and providing meaningful feedback to the user. Additionally, we present case studies and user feedback to assess the effectiveness and usability of our approach.

Ultimately, this research aims to demonstrate that integrating Telegram bots with Linux CLI applications can significantly enhance HCI, making powerful command-line tools more accessible and reducing the learning curve for new users. By leveraging the widespread familiarity and ease of use associated with messaging apps, we offer a novel solution to improve the user experience in interacting with Linux systems.

2. RELATED WORKS

2.1. Weblinux: a scalable in-browser and client-side Linux and IDE [8]

"WebLinux" is a web application that offers a complete Linux OS and integrated development environment (IDE) within a browser, including a terminal, code editor, and file browser. It operates entirely on the client side using a JavaScript-emulated processor, eliminating the need for Virtual Machines or Linux servers. This makes it highly scalable and easy to deploy for a large number of users, particularly in educational contexts like MOOCs. WebLinux addresses the challenge of providing accessible Linux command line and GCC compiler environments to beginners, functioning offline and ensuring consistent performance without relying on volatile third-party services. It builds on the Jor1k emulator and is suitable for large-scale educational use, as demonstrated in courses offered by institutions like IMT on the FUN-MOOC platform.

2.2. Bot-Based Emergency Software Applications for Natural Disaster Situations [9]

This paper investigates the feasibility of using bots to enhance emergency software applications for victims and volunteers during natural disasters. Instead of creating separate specialized apps, the study explores integrating bots into existing everyday software. Three bot-based applications are evaluated: Jayma (sending disaster information to contacts), Ayni (managing volunteer tasks), and Rimay (registering volunteers and managing campaigns). The architecture leverages existing public domain components and executes on commodity hardware. Results highlight resource utilization patterns.

2.3. Coding a Telegram Quiz Bot to Aid Learners in Environmental Chemistry [10]

This paper explores the development and implementation of a Telegram-based Quiz Bot designed to assist students in learning environmental chemistry amid the COVID-19 pandemic. With traditional face-to-face lectures disrupted, the bot provided a remote learning solution, incorporating gamification to enhance engagement and content mastery. The bot allowed students to compete in answering quiz questions, identifying knowledge gaps, and reinforcing their understanding of key concepts. The study, conducted with students from the National University of Singapore, found the Quiz Bot to be well-received, demonstrating the potential of using social media and technology to support and improve online education in higher-level chemistry courses.

ISSN Print : 1979-7141
ISSN Online : 2541-1942

3. METHOD

In this section, we describe the proposed method for enhancing Human-Computer Interaction (HCI) for Linux CLI applications using Telegram Bot piping. The method involves creating a Telegram bot that interfaces with a Linux server to process CLI commands and return the output to the user via Telegram. This approach leverages the interactive and user-friendly nature of Telegram bots to simplify the use of CLI applications, making them more accessible and efficient.

3.1. Telegram Bot

Telegram is a widely-used messaging platform that supports bots—automated accounts that can interact with users and perform predefined tasks. Telegram bots can handle various types of user interactions, such as sending and receiving messages, processing commands, and delivering notifications. Bots are created using the Telegram Bot API, which allows developers to programmatically control bot behaviors and integrate them with other services. Some capabilities of Telegram Bot are as folows:

1. Message Handling: Bots can send and receive text messages, photos, videos, documents, and other types of media.
2. Command Processing: Bots can process commands issued by users, triggering specific actions or responses.
3. Interactive Interfaces: Bots can present interactive elements such as inline keyboards, buttons, and menus to enhance user interaction.
4. Real-time Communication: Bots can provide real-time responses, making interactions quick and efficient.
5. Webhooks and Long Polling: Bots can use webhooks to receive updates from Telegram servers instantly or use long polling to check for updates periodically.

3.2. Command-Line-Interface Integration

The integration of Telegram bots with Linux CLI applications involves several components and a specific communication protocol to ensure seamless interaction between the user and the server. Below, we detail the components of the architecture and the communication protocol.

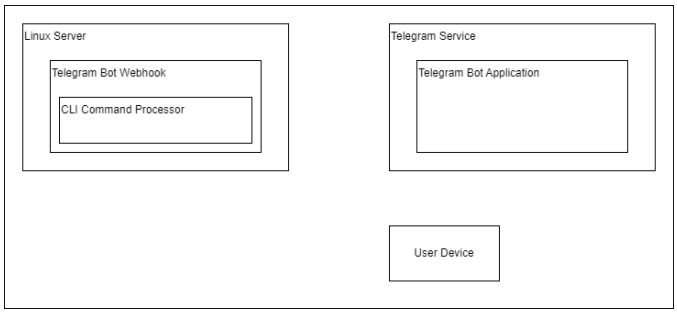

Figure 1. Architecture Diagram

The architecture diagram in Fig. 1. Show the consisting of following components:

ISSN Print : 1979-7141
ISSN Online : 2541-1942

1. User Device: This is the device used by the end-user to interact with the Telegram bot. It could be a smartphone, tablet, or computer with the Telegram app installed. The user sends commands to the Telegram bot via this device and receives the command outputs.
2. Telegram Service: This service acts as the intermediary between the user device and the Telegram bot application. It is responsible for routing the messages and commands from the user to the bot and vice versa. The Telegram service ensures real-time communication and delivery of messages.
3. Telegram Bot Application: Hosted on the Telegram service, this application handles the incoming commands from the user. It processes these commands and interacts with the CLI Command Processor to execute the commands on the server. The Telegram bot application is developed using the Telegram Bot API, allowing it to automate tasks and provide responses based on user inputs.
4. Linux Server: This server hosts the Telegram bot application and executes the CLI commands. It includes two subcomponents as follows.
    a. Telegram Bot Webhook: This component listens for incoming messages from the Telegram service. When a user sends a command via Telegram, it is received by the webhook, which then processes the command and forwards it to the CLI Command Processor.
    b. CLI Command Processor: This component is responsible for executing the CLI commands on the Linux server. It takes the command received from the Telegram bot webhook, processes it, and generates the output. The output is then sent back to the Telegram bot application.

The communication protocol as shown in Fig. 2. describes the steps involved when a user sends a command via the Telegram bot and receives the output. The protocol ensures that commands are correctly processed and responses are delivered efficiently.

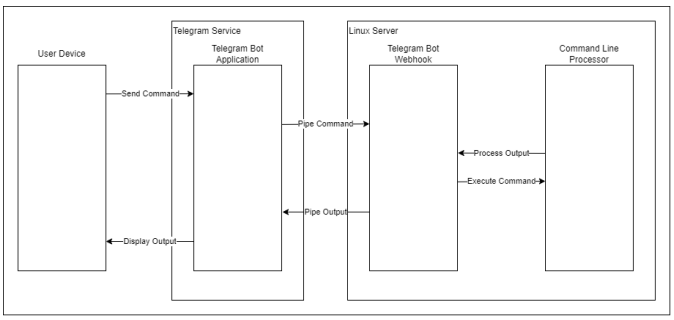

Figure 2. Communication Protocol

1. User Sends Command: The interaction begins with the user sending a command through the Telegram app on their user device.
2. Command Reception by Telegram Service: The Telegram service receives the command from the user's device and forwards it to the Telegram bot application.
3. Command Reception by Telegram Bot Webhook: The Telegram bot webhook on the Linux server receives the command from the Telegram service.
4. Command Processing by CLI Command Processor: The Telegram bot webhook forwards the received command to the CLI Command Processor. The processor executes the command on the Linux server using the appropriate CLI tools and generates the output.
5. Output Reception by Telegram Bot Webhook: The CLI Command Processor sends the output back to the Telegram bot webhook.

ISSN Print : 1979-7141
ISSN Online : 2541-1942

6. Output Transmission to Telegram Service: The Telegram bot application sends the processed output back through the Telegram service.
7. Response Delivery to User Device: The Telegram service routes the command output to the user's device. The user receives the output in their Telegram app, completing the interaction cycle.

This protocol ensures a seamless and interactive experience for users interacting with Linux CLI applications via Telegram. By utilizing Telegram bots, the system makes command-line interfaces more accessible and user-friendly, providing real-time feedback and simplifying complex command executions

4. RESULT AND EVALUATION

In this section, we present the results of the proposed method for enhancing Human-Computer Interaction (HCI) for Linux CLI applications using Telegram Bot piping. We also provide an evaluation based on implementation and user feedback. This section aims to showcase the effectiveness of the system in making CLI applications more accessible and efficient.

4.1. Implementation

The implementation of the Telegram bot integrated with a Linux CLI application involves several steps. These steps detail the process from creating the bot to writing the code for the webhook and piping it to the terminal. First, we create a new bot using the BotFather bot in the Telegram application. We configure the bot by setting its name and username, as illustrated in Fig. 3.

Next, we create a Python script for the Telegram bot webhook that listens for messages sent to our bot. When a user starts the bot or requests help, the bot will respond with "Hi! I am your bot. Send me a command to run in the terminal." Any other message will be interpreted as a Linux command, executed in the terminal, and the result will be displayed by the bot. The code we used is shown in Fig. 4.

Once the code is run on our Linux server, the bot becomes accessible. The command line interface of our Linux server can now be accessed via our Telegram bot, as illustrated in Fig. 5.

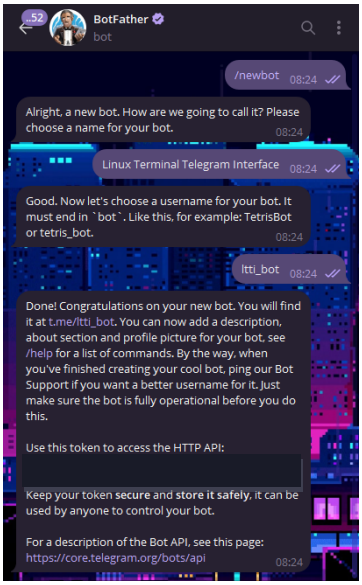

Figure 3. Bot Creation

ISSN Print : 1979-7141
ISSN Online : 2541-1942

```python
import subprocess
import telebot

TELEGRAM_TOKEN = 'XXX'

bot = telebot.TeleBot(TELEGRAM_TOKEN, parse_mode=None)

@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    bot.reply_to(message, "Hi! I am your bot. Send me a command to run in the terminal.")

@bot.message_handler(func=lambda m: True)
def echo_all(message):
    command = message.text
    try:
        result = subprocess.run(command, shell=True, check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
        response = result.stdout if result.stdout else result.stderr
    except subprocess.CalledProcessError as e:
        response = e.stderr if e.stderr else str(e)

    bot.reply_to(message, response)

bot.infinity_polling()
```

Figure 4. Webhook Code

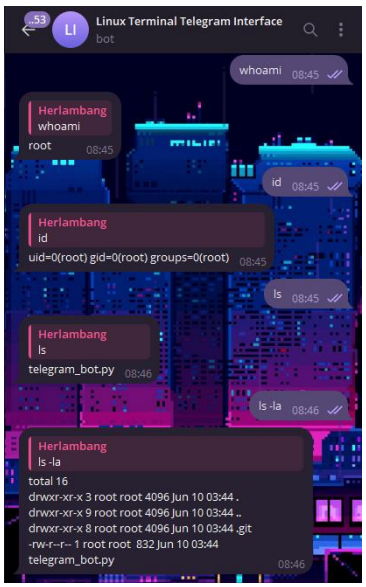

Figure 5. Implementation Result

4.2 User Feedback

User experience will be tested with User Experience Question (UEQ) [11], using a Likert scale survey [12] in the form of a questionnaire given directly to users after they try using the implemented solution. This is part of the Evaluate Design phase. The questionnaire was sent to a total of 20 respondents.

Respondents were selected using a nonprobability sampling method, which is commonly used when conditions do not support the use of probability sampling methods [13]. The technique used is Reliance on Available Subjects, which involves selecting respondents who are available and reachable during the research process [13]. In this study, 20 college students were chosen to be respondents.

The questionnaire uses a Likert scale of 1-5, containing options based on the respondents' opinions ranging from positive to negative, expressed in terms such as "Strongly Agree," "Agree," "Neutral," "Disagree," and "Strongly Disagree." The values for each response are shown in Table I. The questionnaire questions are shown in Table 2.

The questionnaire results are used to calculate the user satisfaction score using Equation 1.

$$satisfaction = \frac{\Sigma w}{x \times y \times z} \times 100\% \qquad (1)$$

ISSN Print : 1979-7141
ISSN Online : 2541-1942

In this equation w represents the total score obtained from respondents, x represents the number of questions on the questionnaire, y represents the number of respondents, and z represents the maximum score for each question as explained in Utomo's research [14].

Table 1. Questionnaire Questions

No.	Question
1	The implementation of the Telegram bot integrated with Linux CLI applications is effective and user-friendly.
2	Using the Telegram bot for interacting with Linux CLI applications enhances the accessibility of command-line tools.
3	The Telegram bot responded promptly to my commands and provided outputs in a timely manner.
4	Interacting with Linux CLI applications through the Telegram bot was intuitive and straightforward.
5	The Telegram bot provided clear instructions on how to interact with the CLI applications.
6	Integrating CLI applications with Telegram bot improves the overall user experience compared to traditional CLI interfaces.
7	I found it convenient to access Linux CLI applications through the Telegram bot.
8	The Telegram bot effectively handled different types of commands and provided accurate outputs.
9	The Telegram bot integration simplified complex command executions on the Linux server.
10	Overall, I would prefer using the Telegram bot for interacting with CLI applications instead of traditional CLI interfaces.

Table 2. Response Value

Response	Value
Strongly Agree	5
Agree	4
Neutral	3
Disagree	2
Strongly Disagree	1

The total assessment score obtained is 953 points. By using Equation 1, the user satisfaction score obtained is 95.3%.

ISSN Print : 1979-7141
ISSN Online : 2541-1942

5. CONCLUTION

This study presents an innovative approach to improve Human-Computer Interaction (HCI) for Linux Command Line Interface (CLI) applications by integrating Telegram bot piping. Despite the power of CLI, its steep learning curve poses a challenge for many users. By leveraging Telegram bots, we aimed to make CLI tools more accessible through natural language interactions. The integration allows users to interact with Linux CLI applications via Telegram, simplifying command executions and providing real-time feedback.

The user feedback indicated high satisfaction with the Telegram bot integrated CLI applications, with users finding it effective, user-friendly, and preferable over traditional CLI interfaces. The overall user satisfaction score of 95.3% reflects strong acceptance of the proposed approach. This research demonstrates the potential of integrating messaging apps with CLI applications to enhance user experience, reduce the learning curve, and broaden the accessibility of Linux systems to a wider audience. Future work could focus on further refining the system and conducting larger-scale user studies to explore additional features and usability improvements.

6. DECLARATION OF COMPETING INTEREST

We declare that we have no conflict of interest.

7. REFERENCES

[1] A. A. Alnuaim *et al.*, "Human-Computer Interaction for Recognizing Speech Emotions Using Multilayer Perceptron Classifier," *J Healthc Eng*, vol. 2022, pp. 1–12, Mar. 2022, doi: 10.1155/2022/6005446.

[2] I. S. MacKenzie, *Human-Computer Interaction: An Empirical Research Perspective*. Elsevier Science, 2024. [Online]. Available: https://books.google.co.id/books?id=f1vbEAAAQBAJ

[3] K. A. Galanis, K. C. Nastou, N. C. Papandreou, G. N. Petichakis, D. G. Pigis, and V. A. Iconomidou, "Linear B-Cell Epitope Prediction for In Silico Vaccine Design: A Performance Review of Methods Available via Command-Line Interface," *Int J Mol Sci*, vol. 22, no. 6, p. 3210, Mar. 2021, doi: 10.3390/ijms22063210.

[4] I. Goldstein, "What! No GUI?," *Information Systems Education Journal*, vol. 17, no. 1, p. 40, 2019.

[5] G. C. Lenardo, Herianto, and Y. Irawan, "Pemanfaatan Bot Telegram sebagai Media Informasi Akademik di STMIK Hang Tuah Pekanbaru," *JTIM : Jurnal Teknologi Informasi dan Multimedia*, vol. 1, no. 4, pp. 351–357, Feb. 2020, doi: 10.35746/jtim.v1i4.59.

[6] C. Huda, F. A. Bachtiar, and A. A. Supianto, "Reporting Sleepy Driver into Channel Telegram via Telegram Bot," in *2019 International Conference on Sustainable Information Engineering and Technology (SIET)*, IEEE, Sep. 2019, pp. 251–256. doi: 10.1109/SIET48054.2019.8986000.

[7] A. Alcayde García, E. Salmeron-Manzano, A. Zapata-Sierra, and F. Manzano-Agugliaro, "A TELEGRAM BOT FOR EDUCATION 4.0: ZQUIZUALBOT 4.0," in *16th International Technology, Education and Development Conference*, Mar. 2022, pp. 7288–7295. doi: 10.21125/inted.2022.1842.

[8] R. Sharrock, L. Angrave, and E. Hamonic, "WebLinux," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, New York, NY, USA: ACM, Jun. 2018, pp. 1–2. doi: 10.1145/3231644.3231703.

[9] G. Ovando-Leon, L. Veas-Castillo, V. Gil-Costa, and M. Marin, "Bot-Based Emergency Software Applications for Natural Disaster Situations," *Future Internet*, vol. 14, no. 3, p. 81, Mar. 2022, doi: 10.3390/fi14030081.

[10] J. S. H. Ong, P. R. Mohan, J. Y. Han, J. Y. Chew, and F. M. Fung, "Coding a Telegram Quiz Bot to Aid Learners in Environmental Chemistry," *J Chem Educ*, vol. 98, no. 8, pp. 2699–2703, Aug. 2021, doi: 10.1021/acs.jchemed.1c00201.

ISSN Print : 1979-7141
ISSN Online : 2541-1942

[11] View of Usability Evaluation In Ruang Guru Applications Using User Experience Questionnaire (UEQ).” Available: https://www.ejournal.iocscience.org/index.php/mantik/article/view/727/484
[12] V. H. Pranatawijaya, W. Widiatry, R. Priskila, and P. B. A. A. Putra, “Penerapan Skala Likert dan Skala Dikotomi Pada Kuesioner Online,” *Jurnal Sains dan Informatika*, vol. 5, no. 2, pp. 128–137, Dec. 2019, doi: 10.34128/jsi.v5i2.185.
[13] E. Babbie, *The Practice of Social Research fourteenth edition*. Boston: Cengage Learning , 2014.
[14] R. T. C. Utomo, R. Purwoko, S. U. Sunaringtyas, and Amirrudin, “Rancang Bangun Cyber Exercise Simulasi Penanganan Insiden Siber Menggunakan Enisa Cyber Exercise Cycle,” *Politeknik Siber dan Sandi Negara*, 2023.

ISSN Print : 1979-7141
ISSN Online : 2541-1942