Hangman Game in Linux

Master of Computer Applications

Under the department

University Institute of Computing

Of

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Session 2024 – 2026

Submitted By: Komal

UID: 24MCA20118

Section: 24MCA-2

Submitted To: Mr. Prashant Patil

Class: MCA(General)

Group: B

Teacher's Signature

Table of Contents

INTRODUCTION

Games like Hangman engage users in a fun, educational way, combining entertainment with vocabulary-building. This project focuses on building a Hangman game on a Linux platform, aiming to provide users with an interactive guessing experience that challenges them to complete words under certain constraints.

1.1 Problem Statement

Classic games like Hangman offer a blend of challenge and learning, but they are often confined to specific formats or platforms. Additionally, implementing such games on Linux presents unique challenges, such as ensuring compatibility with various distributions and providing a smooth, responsive user experience in a terminal-based environment. This project aims to develop a Hangman game for Linux, designed to work seamlessly in a terminal, where users can enjoy guessing words while keeping track of their attempts. The project will enhance the game's interactivity by automating hints, tracking progress, and updating the display based on user guesses.

1.2 Objectives

- To design and implement a Hangman game that allows users to guess words letter-by-letter, with a limited number of attempts.
- To build a responsive and clear user interface in the Linux terminal, displaying game progress and hint mechanisms.
- To track and display the number of incorrect guesses and remaining attempts, enhancing user engagement.
- To create a user-friendly, interactive environment for playing the game within a Linux terminal.

1.3 Scope

This project focuses on creating a Hangman game specifically for the Linux terminal environment. It will include core functionalities like word selection, tracking incorrect guesses, and updating the display after each input. Future enhancements could include options for difficulty levels, customizable word lists, and interactive scoring. Additionally, features such as integrating visual representations of the Hangman or multiplayer capabilities could be added to make the game more engaging for a wider audience.

LITRATURE REVIEW

2.1 Overview of Game Development in Linux Platforms

Developing games on Linux has evolved significantly, with the platform now hosting a variety of interactive games, including text-based and graphical options. Traditionally, Linux was not seen as a primary gaming platform due to limited support; however, advancements in open-source libraries and toolkits have made it possible to create more complex games. As Patel (2019) highlights, Linux provides developers with a versatile environment where open-source tools can be leveraged to build engaging games.

2.2 Automation in Game Mechanics

Automation plays a vital role in enhancing user engagement in games by minimizing repetitive tasks and simplifying game flow. In the case of Hangman, automated mechanisms such as tracking user guesses and updating the word's display after each input streamline the gameplay and improve user experience. According to Johnson (2020), automation in game mechanics enables more dynamic and responsive gaming experiences, increasing replayability and user satisfaction.

2.3 User Experience and Interaction in Terminal-Based Game

User experience is crucial in terminal-based games, where graphical limitations require a focus on clear and intuitive design. Studies show that text-based games can be highly engaging if designed thoughtfully, with emphasis on responsive feedback and logical flow (Smith, 2018). For Hangman on Linux, elements such as a clean display of attempts, incorrect guesses, and remaining letters enhance the interactive experience, as well as user satisfaction in a simple terminal setup (Jones, 2021).

2.4 Database Management and Game State Tracking

Effective data management is essential for games that track user progress, such as storing the number of guesses and updating game states. Research by Williams (2020) indicates that managing game states efficiently in code optimizes performance, especially in resource-constrained environments like terminal games. In the Hangman game, structuring the data for each game session ensures smooth gameplay, maintaining continuity between rounds and allowing players to easily follow their progress.

METHODOLOGY

This chapter outlines the steps taken to develop the Hangman game on a Linux platform, detailing the processes involved in design, development, testing, and deployment.

3.1 Requirements Analysis

The project began with defining the key functionalities essential for an engaging Hangman game experience. These functionalities included:

- **Word Selection and Display**: A mechanism for selecting words randomly from a predefined list and displaying blanks for each letter.
- **Guess Tracking**: Tracking correct and incorrect guesses, updating the word display after each input.
- **Attempt Limits**: Setting a maximum number of incorrect guesses before ending the game.
- **User Feedback**: Displaying real-time feedback on remaining attempts and revealing letters for correct guesses.

3.2 Game Design

The game was designed with a focus on simplicity and user engagement:

- **Architecture**: The game is built for the Linux terminal, using Python for a command-line interface that displays the game's progress and accepts user input.
- **Data Structure**: The game uses lists and dictionaries to store the word selection, player guesses, and track incorrect attempts. The main structure includes:
    - **Word Pool**: A list containing words that are randomly selected at the start of each game.
    - **Display State**: A dynamic list updated as users guess letters.
    - **Game State**: A dictionary storing player progress, remaining attempts, and guessed letters.

3.3 Development Process

The development process followed a structured approach:

1. **Frontend Development**: The game interface is designed to display the current state of the word, showing blanks and revealing correct guesses, and providing user feedback on incorrect guesses and attempts left.

2. **Backend Development**: The core game logic was implemented in Python, with functions for:

- **Selecting Words**: A function that selects random words from the pool.
- **Processing Guesses**: This function checks if a guessed letter is in the word and updates the display if correct or decrements attempts if incorrect.
- **End Conditions**: Logic to check if the player has guessed the word correctly or if they have run out of attempts.

3. **User Feedback Mechanism:** The game provides real-time feedback after each guess, showing the current state of the word and notifying the user of remaining attempts or incorrect guesses.

3.4 Testing and Validation

Testing was conducted to ensure game functionality and an enjoyable user experience:

- **Unit Testing**: Individual functions were tested, including word selection, guess processing, and display updates.
- **Integration Testing**: The interaction between different components, such as the display and guess processing, was validated to ensure smooth gameplay.
- **User Acceptance Testing (UAT)**: Feedback was collected from initial users to confirm that the game was intuitive and enjoyable.

3.5 Deployment

After testing, the Hangman game was set up for deployment in a Linux environment. The deployment process involved:

- **Setup in Linux Terminal**: Configuring the game to run smoothly in a terminal interface with accessible commands and instructions.
- **Documentation**: Providing a README file with instructions on running the game, including dependencies and usage guidelines.
- **Github Link:** https://github.com/komal483/hangman_game

3.6 Evaluation and Feedback

Following deployment, user feedback was gathered to assess game performance and identify areas for improvement. Continuous updates were planned to incorporate user suggestions, such as adding more word lists or difficulty levels.

RESULTS

The Hangman game was evaluated through extensive testing to ensure it met the intended functionalities and provided a smooth user experience within the Linux terminal. The evaluation focused on game functionality, user interface responsiveness, and areas for potential enhancement.

4.1 Game Functionality and Accuracy

The game was tested using various scenarios to validate core functionalities, such as word selection, guess processing, and tracking attempts.

- **Word Selection**: The game accurately selected random words from a predefined list at the start of each game, maintaining a fair and varied experience for the player.
- **Guess Processing**: The game effectively tracked correct and incorrect guesses, updating the display accordingly. Test cases confirmed that only valid single-letter guesses were accepted, and duplicate guesses were managed without affecting the remaining attempts.
- **Attempt Limits**: The game's limit on incorrect guesses functioned as intended, ending the game once the maximum number of incorrect attempts was reached and displaying the correct word if the player had not guessed it.

4.2 User Interface Responsiveness

The user interface was designed to be clear and interactive, making it easy for users to follow game progress and feedback in the Linux terminal.

- **Response Time**: The game maintained a near-instant response time for most actions, such as input validation and display updates. This quick response helped maintain an immersive and engaging game experience, even in a terminal-based setup.
- **Real-Time Feedback**: The interface displayed real-time feedback for each guess, showing the word's progress and the number of attempts remaining, which kept players informed and motivated.

4.3 Limitations and Areas for Improvement

Despite its functionality, the Hangman game showed a few limitations that could be enhanced in future versions:

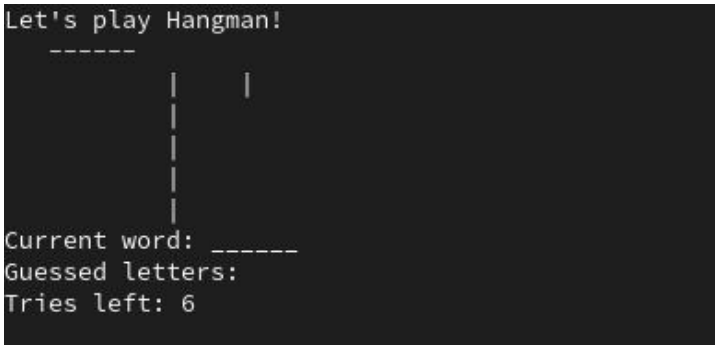
- **Word Pool Expansion**: The current game uses a predefined list of words, which may limit replayability. Adding an option for users to upload their own word lists or generating words dynamically would enhance variety.
- **Difficulty Levels**: The game could be further improved by incorporating difficulty levels that adjust the maximum number of incorrect guesses or choose words based on complexity.
- **Graphical Enhancement**: While the game runs smoothly in the terminal, adding simple ASCII art for each incorrect guess could improve the visual experience.

4.4 Future Work

Based on feedback and testing results, future development could focus on:

- **Adding a Scoring System**: Implementing a scoring mechanism based on accuracy and the number of attempts would add a competitive aspect to the game.
- **Multiplayer Mode**: Adding a two-player mode, where one player enters a word for the other to guess, could increase engagement.
- **Enhanced User Feedback**: Incorporating simple animations or colors for correct and incorrect guesses would improve game feedback and make the experience more dynamic.

OUTPUT



```
Guessed letters: centpq
Tries left: 3

Guess a letter: r
Wrong guess!
   ------
         |    |
         |    O
         |   /|
         |   /
         |
Current word: p_t__n
Guessed letters: centpqr
Tries left: 2

Guess a letter: y
Good guess!
   ------
         |    |
         |    O
         |   /|
         |   /
         |
Current word: pyt__n
Guessed letters: centpqry
Tries left: 2

Guess a letter: k
Wrong guess!
   ------
         |    |
         |    O
         |   /|
         |   /           |
Current word: pyt__n
Guessed letters: centpqryk
Tries left: 1

Guess a letter: a
Wrong guess!
Sorry, you ran out of tries. The word was: python
```

```
Guess a letter: r
Good guess!
   ------
         |    |
         |    O
         |   /|
         |   /           |
Current word: devel_per
Guessed letters: pqdfghjevlr
Tries left: 1

Guess a letter: o
Good guess!
Congratulations! You've guessed the word: developer
[vipin@localhost ~]$
```

FUTURE SCOPE

The current Hangman game provides a fun and interactive experience for users on a Linux terminal. However, there are several enhancements and additional features that could make the game more engaging and versatile. Key directions for future development include:

1. Incorporation of Additional Features

- **Difficulty Levels**: Adding easy, medium, and hard difficulty levels by adjusting the number of attempts and complexity of chosen words would allow players to select their preferred challenge.
- **Hint System**: Implementing a hint system that provides users with clues or reveals random letters could enhance gameplay and help less experienced players.

2. Enhanced User Interface

- **Graphical Interface**: Transitioning from a purely command-line interface to a graphical interface using tools like Tkinter or Pygame would improve accessibility and engagement.
- **ASCII Art Display**: Adding ASCII art that visually represents the hangman as incorrect guesses are made could add an entertaining, visual element to the game.

3. Improved User Interface

- **Two-Player Option**: Introducing a two-player mode, where one player inputs a word for the other to guess, would encourage social interaction and increase replayability.
- **Online Multiplayer**: Developing an online multiplayer mode, where players can compete against each other remotely, would make the game more interactive and enjoyable for a wider audience.

4. Scoring and Leaderboards

- **Scoring System**: Implementing a scoring system based on guess accuracy and attempts could introduce a competitive aspect and motivate players to improve.
- **Leaderboard**: Adding a leaderboard that tracks the highest scores could encourage players to achieve new records and share results with others.

5. Improved Word Management

- **Custom Word Lists**: Allowing players to create and use custom word lists would enable tailored game experiences based on different themes or difficulty levels.
- **Dynamic Word Pool**: Using an API to fetch words dynamically could expand the word list, making the game more unpredictable and challenging over time.

6. Integration Of AI based Features

- **Smart Word Recommendations**: Implementing an AI model to adjust word difficulty based on player skill levels could offer a more personalized experience.
- **Hint Generation through NLP**: Using natural language processing to provide context-based hints would add an educational element to the game, helping players learn more as they play.

7. **Mobile Compatibility**

- **Mobile Version**: Developing a mobile version of the game would allow users to play on the go, broadening the game's reach and user base.

CONCLUSION

The goal of this project was to create an engaging and interactive Hangman game that could be played on a Linux terminal. The project aimed to provide an enjoyable user experience while highlighting fundamental programming concepts like control structures, functions, and user interaction in Python. The completed game successfully achieves these objectives and offers the following key insights:

1. **Interactive and Engaging Gameplay**

The Hangman game provides a fun and accessible experience for users. With a straightforward command-line interface, players can enjoy guessing words while honing their vocabulary skills. This simple but engaging game can be a valuable addition to Linux users seeking a quick and enjoyable break from routine tasks.

2. Importance of User Feedback
To improve the user experience, interactive feedback was integrated into the game, offering immediate responses to correct or incorrect guesses. This feature contributes to the game's appeal, as players receive clear visual indicators of their progress, making gameplay more immersive.

3. Learning Tool for Beginners
The Hangman game serves as an excellent tool for beginners to understand programming basics, including string manipulation, conditional logic, and loops. The project demonstrates how fundamental coding principles can be combined to create an enjoyable application, making it ideal for those learning programming on Linux.

4. Potential for Future Enhancements
While the current implementation is effective and user-friendly, the project also opens up various possibilities for future development. Adding features such as a graphical interface, different difficulty levels, or multiplayer options would increase the game's appeal and versatility.

5. Contribution to the Linux Gaming Ecosystem
The Hangman game contributes to the variety of simple games available on Linux, enhancing the user experience on this platform. By offering a Python-based game, it aligns with Linux's open-source ethos and provides users with a customizable and expandable application.

In summary, this Hangman game effectively fulfills its purpose as a fun, interactive, and educational tool for Linux users. Its straightforward design, combined with the potential for future enhancements, makes it a promising addition to Linux-based games. With continued development, this project has the potential to evolve into a more sophisticated and user-friendly application, appealing to a broader audience.