Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi

Name: Nauman Ali Murad
Registration: 2022479
CS331 - Operating System
Taught by: Ms. Mah Rukh

Title: SysCallWatch - AI Powered System Call Tracer for Linux based OS

Table of Contents

1. Introduction
2. Features
3. Technical Stack
4. Project Architecture
5. Implementation Details
6. Graphical User Interface (GUI)
7. Testing and Validation
8. Future Work
9. Conclusion

Introduction

SysCallWatch is a Python-based tool for monitoring and analyzing system calls in Linux. It provides an intuitive GUI for generating system call logs, analyzing them using the Gemini Large Language Model (LLM), and generating clear and simple summaries. The tool was developed as a semester project for the **CS331 - Operating Systems** course at GIK Institute.

Features

1. **System Call Tracing**:
    - Capture and log system calls invoked by a target program.
2. **Log File Analysis**:
    - Use the Gemini LLM to analyze logs and generate plain-text reports.
3. **GUI for Ease of Use**:
    - Intuitive interface for tracing, reporting, and exploring system calls.
4. **Common System Calls**:
    - Provides a list of commonly used system calls for quick reference.
5. **Cross-Platform Compatibility**:
    - The GUI is implemented in Python's `tkinter` library, making it portable.

Technical Stack

1. **Programming Languages**:
    - Python (for GUI and API communication)
    - C (for system call tracing)
2. **APIs**:
    - Gemini API for natural language report generation.
3. **Libraries and Frameworks**:
    - `tkinter` for GUI development.
    - `subprocess` and `os` for system call handling.

    - `requests` and `json` for API communication.
4. **Log Analysis**:
    - System call log files analyzed using Gemini API.

Project Architecture

1. **Frontend**:
    - Developed in Python using `tkinter` for buttons, text inputs, and output display.
2. **Backend**:
    - C program using `ptrace` system call to monitor the target program.
    - Python handles the logs and integrates with the Gemini API.
3. **Integration**:
    - Python code communicates with the C program and the Gemini API.

Implementation Details

1. System Call Tracing (C Program)

- **Mechanism**:
    - The `ptrace` system call monitors child processes.
    - The target program is executed in the child process, and every system call made by the child is captured by the parent process.
- **Log File**:
    - Logs all system calls, categorizing them as known or unknown.
- **Summary Statistics**:
    - Provides details on the total number of system calls, known and unknown.

2. GUI (Python)

- **Main Features**:
    - Input field for specifying the target program.
    - Buttons for generating logs, sending logs to Gemini, and viewing reports.
    - A terminal-like interface for displaying logs and outputs.
- **System Call List**:
    - Displays common system calls with a copy-to-clipboard feature.

3. Gemini API Integration

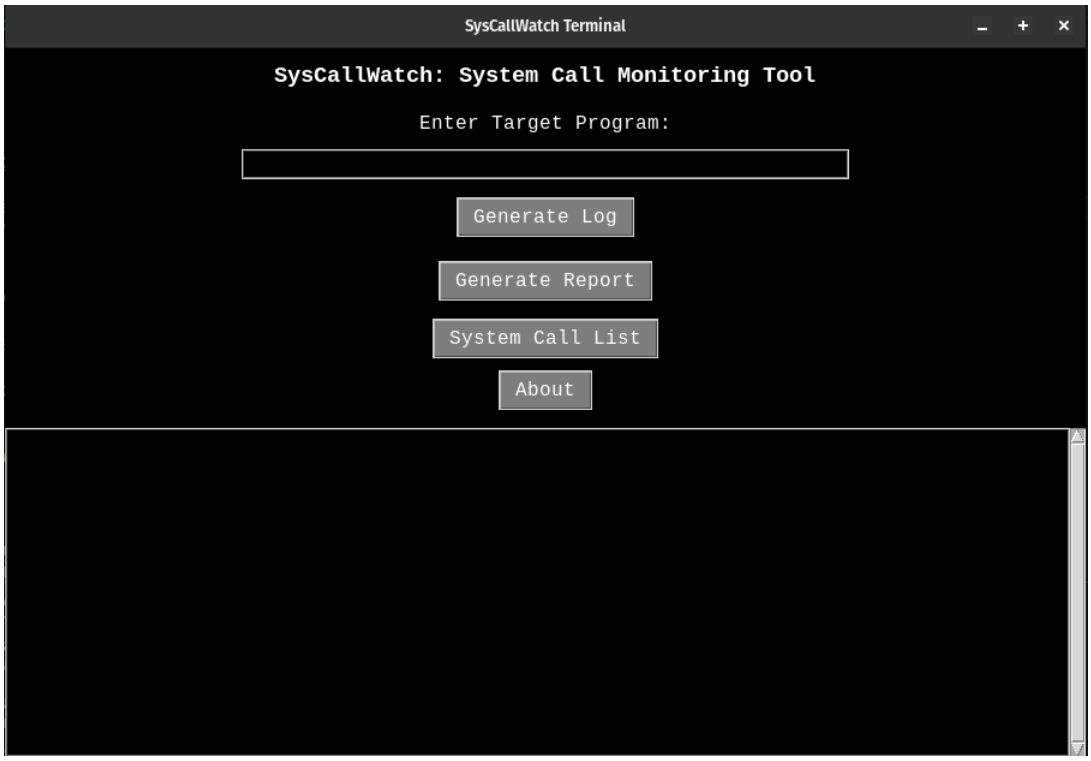
- **Purpose**:
    - To generate plain-text summaries of the system call logs.
- **Data Flow**:
    - Log file content is sent to the Gemini API.

- The API response is parsed and displayed in the GUI.

Graphical User Interface (GUI)

The following sections describe the GUI elements. You can add screenshots alongside these descriptions.

1. **Main Window**:
    - Input field for the target program.
    - Buttons: *Generate Log*, *Generate Report*, *System Call List*, and *About*.



Generate Log: Enter the name of the target program and click "Generate Log" to start tracing system calls.

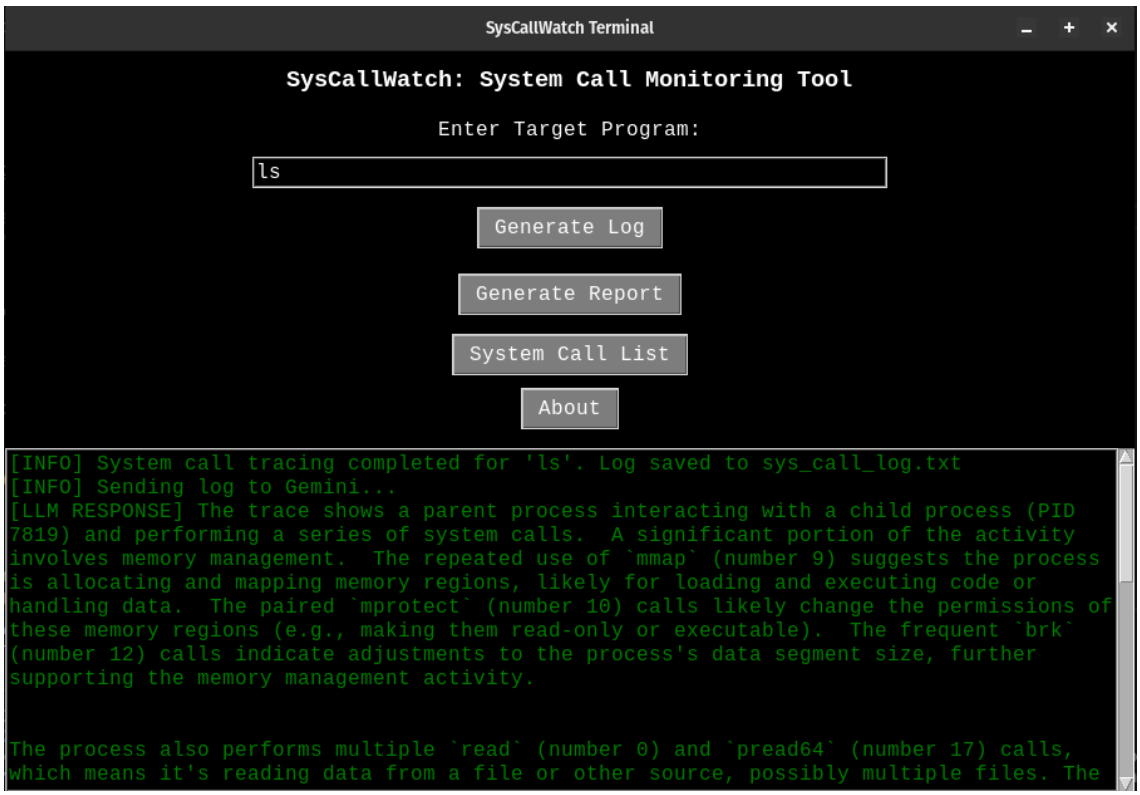
Generate Report: Once the log is generated, click "Generate Report" to analyze the log using the Gemini API and get a detailed summary.

System Call List: Click "System Call List" to view a list of commonly used system calls, and easily copy any of them to your clipboard.

2. **System Call List Popup**:
    - Displays commonly used system calls with a copy-to-clipboard feature.

3. **About Section**:
    - Provides project details, credits, and GitHub repository link.

4. **Terminal Output Area**:
    - Displays log generation, API responses, and status messages.



Testing and Validation

Test Cases

- **Log Generation**:
    - Verified the accuracy of logs generated for known programs.
- **API Response**:
    - Ensured that Gemini API responds with appropriate summaries.
- **GUI Functionality**:
    - Validated input handling, button interactions, and output display.

Future Work

1. **Dynamic System Call Tracking**:
    - Add real-time monitoring of system calls.
2. **Error Handling**:
    - Improve robustness against API failures and invalid inputs.
3. **Enhanced Log Analysis**:

    - Incorporate more advanced analysis using other AI models.
4. **Cross-Platform Compatibility**:
    - Expand support to macOS and Windows (limited by system call differences).

Conclusion

SysCallWatch successfully demonstrates the integration of system-level programming with modern AI tools. It provides a user-friendly interface for analyzing complex system-level behaviors, making it accessible to both beginners and advanced users.

For more details and contributions, visit the GitHub Repository.