Project Report

CS-311 Operating Systems

Team Members:

Marium Imran (2023303)

Momina Najeeb (2023334)

Wafa Abbass (2023748)

Submitted To:

Ms. Safia Baloch

Submission Date:

21-12-25

Introduction to the Problem

1. Traditional command-line interfaces require manual typing to interact with the operating system.
2. Manual interaction may be inconvenient or inaccessible for some users.
3. Voice-based interaction provides an intuitive and efficient alternative to traditional input methods.
4. This project aims to bridge voice interaction with operating system command execution.
5. The system converts spoken commands into executable operating system commands.
6. The project demonstrates practical operating system concepts through a real-world application.
7. In addition to voice input, the system provides audio feedback using text-to-speech to enhance user interaction.

Unique Functionality of the Project

1. The system allows execution of operating system commands using voice input.
2. Natural language spoken by the user is converted into text using a speech recognition API.
3. The recognized text is mapped only to predefined and secure system commands.
4. Arbitrary command execution is restricted to ensure system safety.
5. Each command execution displays process ID, execution time, and exit status.
6. The project integrates usability with operating system process management concepts.

7. The system verbally confirms important system actions using text-to-speech, such as startup greeting and command execution confirmation.

System Flow and Function Call Description

1. Voice input is captured through the microphone.
2. Audio input is converted into text using a speech-to-text API.
3. Converted text is passed to a command parsing function.
4. The parser matches the text with predefined command patterns.
5. If a valid match is found, the corresponding system command is selected.
6. The command is executed as a separate operating system process.
7. The operating system returns command output, exit code, and execution time.
8. The results are displayed to the user.
9. The system provides verbal feedback using a text-to-speech engine to confirm command execution.

Function Calls Used in the Program

1. The **main()** function controls the overall program execution loop, initializes the text-to-speech engine, and coordinates interaction between all modules.
2. The **listen_and_recognize()** function captures audio input and converts spoken commands into text using the SpeechRecognition library.
3. The **parse_text_to_command()** function processes natural language input and maps it to predefined operating system commands.

4. The **run_command()** function creates and manages operating system processes using the subprocess module, retrieves process metadata, and handles standard output and error streams.
5. The **speak()** function provides audio feedback to the user using the **pyttsx3** text-to-speech library.

Libraries Used in the Project

• **SpeechRecognition:** Used for capturing audio input and converting speech into text.
• **subprocess:** Used to create, execute, and manage operating system processes.
• **pyttsx3:** Used to provide text-to-speech audio feedback to the user.
• **time:** Used to measure command execution time.

System Flow

Voice Input → Speech to Text Conversion → Command Parsing → Process Creation (OS) → Kernel Execution → Output Display → Audio Feedback

---

Commands to Execute the Program

Prerequisites

- Python 3.x installed on the system
- A working microphone for voice input
- Internet connection (required for speech recognition API)
- Supported Operating System: Linux

Setup Instructions

1. Clone the GitHub repository to your local machine:
   git clone (*https://github.com/mariumgk/Voice-driven-Remote-Shell.git*)
2. Navigate to the project directory:
   cd Voice-driven-Remote-Shell-main
3. Install the required Python dependencies:
   pip install SpeechRecognition pyaudio
   pip install pyttsx3

Executing the Program

1. Run the main program file using the following command:
   python voice_shell.py
2. Once the program starts, speak supported voice commands to execute operating system tasks.
3. The system will display: Process ID (PID), Execution time, Exit status, and Command output.
4. To exit the program, type: exit

Supported Voice Commands

File and Directory Commands

• **"list files"**, **"show files"**, **"ls"**
Lists all files and directories in the current working directory.

• **"current folder"**, **"where am I"**, **"pwd"**
Displays the present working directory.

• **"clear screen"**, **"clear"**
Clears the terminal screen.

• **“show file <filename>”**
Displays the contents of the specified file.

System Information Commands

• **“show date”**, **“current date”**, **“system time”**
Displays the current system date and time.

• **“who am I”**, **“my username”**, **“current user”**
Displays the username of the currently logged-in user.

• **“cpu information”**, **“cpu details”**, **“processor info”**
Displays detailed information about the system processor.

• **“kernel version”**, **“system information”**, **“os version”**
Displays kernel and operating system version information.

Resource Monitoring Commands

• **“show processes”**, **“list processes”**, **“running processes”**
Displays a list of currently running system processes.

• **“disk usage”**, **“show disk space”**, **“storage usage”**
Displays disk space usage of the file system.

• **“memory usage”**, **“show memory”**, **“ram usage”**
Displays system memory usage.
