Oliver Iraheta 12151028 29/03/2025

Informe Técnico Proyecto Sistemas Operativos 1

Descripción y funcionamiento del proyecto

- dwimsh (Do What I Mean Shell) es un shell interactivo diseñado para sistemas operativos basados en Unix. Su propósito principal es asistir a los usuarios corrigiendo errores en la invocación de comandos y sugiriendo opciones posibles para ejecutar el comando correcto.
- Cuando un usuario introduce un comando en dwimsh, el sistema verifica si dicho comando existe en el directorio /usr/bin. Si el comando es válido, se ejecuta directamente.
- Si el comando ingresado no es válido, dwimsh busca comandos similares en /usr/bin utilizando la distancia de Levenshtein para determinar la similitud entre palabras. Luego, presenta sugerencias al usuario para que elija la corrección más adecuada antes de ejecutarlo.
- El shell maneja únicamente comandos sin redirección, pipes o ejecución en segundo plano.

Descripción de rutinas y procedimientos

- **Tokenización de la entrada:** La función tokenize divide la línea ingresada por el usuario en palabras separadas (tokens) para facilitar su análisis y ejecución.
- **Obtención de comandos disponibles:** getCommands obtiene una lista de comandos del directorio /usr/bin, lo que permite verificar si un comando ingresado es válido o si se necesita sugerir alternativas.
- **Cálculo de similitud entre comandos:** levenshteinDistance implementa el algoritmo de distancia de Levenshtein para medir qué tan similar es un comando ingresado a los disponibles en el sistema.
- **Búsqueda de comandos similares:** findClosestCommands compara el comando ingresado con la lista de comandos disponibles y devuelve una lista ordenada de los más parecidos.
- **Ejecución de comandos:** executeCommand utiliza fork y execvp para ejecutar comandos en un proceso hijo, permitiendo que el shell continúe funcionando sin interrupciones.
- **Manejo del ciclo principal:** main ejecuta el ciclo principal del shell, capturando la entrada del usuario, procesándola, y decidiendo si ejecutar el comando directamente, sugerir correcciones o finalizar el programa.

Comentarios de implementación

- **Decisiones de diseño y optimizaciones:** Se priorizó la eficiencia en la búsqueda de comandos utilizando estructuras de datos adecuadas. Se implementó la distancia de Levenshtein para mejorar la precisión de las sugerencias.
- **Posibles mejoras:** Se podría incluir compatibilidad con redirección de entrada/salida, manejo de pipes y soporte para scripts. Además, mejorar la velocidad de búsqueda mediante estructuras más avanzadas, como árboles de prefijos.