Proyecto 1

Profesora
Cecilia Hernández

Ayudante
Jesús Gomez

Integrantes
Jaime Ansorena Carrasco - 2020401497
Diego Oporto Valenzuela - 2020430403
Dayan Sáez Cubillos - 2020444854

Descripción del código

Se desarrolla una shell llamada "`mishell`" que puede ejecutar comandos ingresados por el usuario. La shell puede ejecutar comandos simples con uso de pipes y además tiene la capacidad de ejecutar un daemon en segundo plano que registra información en el sistema. A continuación, se proporciona una breve explicación de cada función en el código:

- `char* read_input()`

  Esta función se utiliza para leer una línea de entrada del usuario. Utiliza `getline()` para leer la entrada del usuario desde la entrada estándar (`stdin`) y devuelve la línea leída como una cadena (`char*`).

- `void parse_input(char* input, char* commands[][100], int* num_commands)`

  Esta función se encarga de analizar y separar la entrada del usuario en comandos y argumentos. Toma la línea de entrada como argumento (`input`) y divide la línea en comandos separados por el carácter '|' (pipe). Cada comando y sus argumentos se almacenan en una matriz llamada `commands`. El número total de comandos se almacena en `num_commands`.

- `void execute_commands(char* commands[][100], int num_commands)`

  Esta función se utiliza para ejecutar los comandos que se han analizado previamente en `parse_input()`. Utiliza pipes para redirigir la salida estándar de un comando como entrada estándar para el siguiente. Cada comando se ejecuta en un proceso hijo utilizando `fork()` y `execvp()`. Los procesos hijos son esperados con `wait()` por el proceso padre para asegurarse de que se completen antes de continuar con el siguiente comando.

- `void start_daemon(int t, int p)`

  Esta función se utiliza para iniciar un demonio que se ejecuta en segundo plano. Primero, crea un nuevo proceso utilizando fork. El proceso hijo se convierte en un daemon utilizando setsid, lo que lo desvincula del terminal actual. Luego, abre el registro del sistema (`syslog()`) y entra en un bucle donde recopila información del sistema y la registra en el sistema de registro. El demonio se ejecuta durante `p` segundos, registra información cada `t` segundos, y se detiene después de ese período. Para matar al demonio basta con ejecutar el comando `kill` junto con el PID del demonio desde otra terminal.

- `int main()`

  Ejecuta un bucle infinito que muestra un indicador de shell (`"mishell:$"`) y espera la entrada del usuario. Luego, llama a `read_input` para obtener la entrada del usuario y verifica si el usuario ha ingresado un comando específico. Puede ejecutar comandos ingresados por el usuario, iniciar un daemon con el comando `daemon <t> <p>` o salir de la shell con el comando `exit`. Dependiendo de la entrada del usuario, se llaman a las funciones apropiadas.