Consola de Comandos

Nicolás Ricciardi
Enzo Levancini
Matias Cruces
Sebastian Rosas

Sistemas Operativos - Segundo Semestre 2024

1 Introducción

El presente informe detalla la implementación de una *shell* en C, que permite la ejecución de comandos en procesos concurrentes. Esta shell incluye características adicionales como el manejo de tuberías (*pipes*), la persistencia de comandos favoritos y la funcionalidad de recordatorios. El desarrollo se enmarca en el Proyecto 1 de la asignatura de Sistemas Operativos, cuyo objetivo es fortalecer las competencias de los estudiantes en el manejo de procesos en Unix.

2 Objetivos

El objetivo principal es implementar una shell que permita:

- Ejecutar comandos en procesos concurrentes.
- Manejar tuberías para la comunicación entre procesos.
- Administrar una lista de comandos favoritos de forma persistente.
- Implementar un sistema de recordatorios con temporizadores.

3 Funcionalidades Implementadas

3.1 Ejecución de Comandos Concurrentes

La shell proporciona un *prompt* en el que el usuario puede ingresar comandos. Estos comandos se ejecutan en un proceso hijo, utilizando la llamada al sistema `fork()` para crear el proceso y `execvp()` para ejecutar el comando deseado. El proceso padre espera la finalización del proceso hijo antes de continuar. Esto permite ejecutar comandos con múltiples argumentos y garantizar la ejecución en primer plano.

3.2 Soporte para Pipes

La shell implementada permite que varios comandos se conecten mediante *pipes*, de forma que la salida de un comando se utilice como entrada de otro. Para ello, se hace uso de la llamada al sistema `pipe()` y se divide la entrada del usuario en comandos individuales, ejecutándolos en procesos concurrentes conectados por tuberías. Un ejemplo de comando soportado es:

```
ps -aux | sort -nr -k 4 | head -20
```

3.3 Comandos Favoritos

Una característica distintiva de esta shell es la implementación de un sistema de comandos favoritos. Los comandos que el usuario ejecuta se almacenan en una lista de favoritos, evitando duplicados. Esta lista puede gestionarse mediante varios comandos personalizados:

- `favs crear`: Crea un archivo de favoritos en una ruta especificada.
- `favs mostrar`: Muestra la lista de comandos favoritos junto con su identificador numérico.
- `favs eliminar`: Elimina uno o más comandos favoritos según su identificador.
- `favs ejecutar`: Ejecuta un comando favorito seleccionado.
- `favs cargar`: Carga comandos favoritos desde un archivo.
- `favs guardar`: Guarda los comandos favoritos en un archivo.

Esto permite mantener una lista persistente de los comandos más utilizados, que el usuario puede gestionar fácilmente durante la ejecución de la shell.

3.4 Sistema de Recordatorios

La shell también incluye un comando personalizado para establecer recordatorios temporizados. El comando `set recordatorio` permite al usuario definir una tarea recordatoria que se activará después de un periodo de tiempo especificado. Por ejemplo:

```
set recordatorio 10 "Hacer una pausa activa"
```

Esto configurará un recordatorio que, después de 10 segundos, desplegará un mensaje en la shell. Esta funcionalidad se implementa mediante el uso de la llamada al sistema `sleep()` en un proceso hijo.

4 Estructura del Código

El código se organiza en funciones que implementan las diversas características de la shell. A continuación se describen algunas de las funciones más importantes:

- `ejecutar_comando()`: Ejecuta un comando en un proceso hijo utilizando `execvp()`.
- `manejar_pipes()`: Maneja la ejecución de comandos conectados por pipes, dividiendo la entrada en múltiples procesos.
- `agregar_favorito()`: Añade un comando a la lista de favoritos si no está ya presente.
- `manejar_favs()`: Implementa las funcionalidades relacionadas con la gestión de comandos favoritos.
- `set_recordatorio()`: Configura un recordatorio para mostrar un mensaje tras un tiempo definido.

5 Pruebas Realizadas

Se llevaron a cabo diversas pruebas para verificar el correcto funcionamiento de la shell, incluyendo:

- Ejecución de comandos básicos como `ls`, `pwd`, y `cat`.
- Uso de pipes para conectar comandos, como `grep`, `sort`, y `head`.
- Gestión de favoritos, probando la adición, eliminación y ejecución de comandos favoritos.
- Configuración de recordatorios para tareas temporizadas.

6 Conclusiones

El proyecto de la shell en C logró cumplir con los objetivos planteados, proporcionando una interfaz funcional para la ejecución de comandos y la administración de procesos. Además, las funcionalidades personalizadas como la persistencia de comandos favoritos y el sistema de recordatorios añaden un valor adicional a la experiencia del usuario.

7 Repositorio

El código fuente del proyecto y las instrucciones de compilación y ejecución se encuentran disponibles en el siguiente repositorio: https://github.com/sebargith/Consola-de-comandos.