Universidad Nacional
Jorge Basadre Grohmann

PROYECTO DE
MINISHELL

Sistemas Operativos

I N T E G R A N T E S

Contenidos del Proyecto

Jarem Manuel Vargas Centeno	2023-119065
Abdiel Jean Tapia Milaveres	2023-119062

EL PROYECTO

Desarrollar un intérprete de comandos (mini-shell) en C++ que permita ejecutar procesos en Linux, aplicando mecanismos de concurrencia, redirección de flujos, y gestión de memoria.

DISEÑO Y ARQUITECTURA

Módulo	Descripción	Archivos principales
main.cpp	Punto de entrada. Inicializa la shell.	main.cpp
Shell	Bucle del prompt, manejo de señales, integración general.	shell.cpp, shell.hpp
Parser	Tokeniza y detecta operadores (>, <, |, &).	parser.cpp, parser.hpp
Executor	Crea procesos hijos, maneja pipes y redirecciones.	executor.cpp, executor.hpp
Builtins	Comandos internos (cd, pwd, help, etc.).	builtins.cpp, builtins.hpp
Utils	Funciones auxiliares (historial, strings, logging).	utils.cpp, utils.hpp

El sistema se estructura siguiendo una arquitectura modular que separa las responsabilidades en componentes bien definidos:

DETALLES DE IMPLEMENTACIÓN

Llamadas del Sistema POSIX

Función / API POSIX	Uso en el proyecto
`fork()`	Crear procesos hijos para ejecutar comandos externos
`execvp()`	Reemplazar el proceso hijo por el nuevo programa
`wait() / waitpid()`	Sincronizar la finalización del proceso hijo
`dup2()`	Redirección de flujos de entrada y salida
`pipe()`	Comunicación entre procesos mediante pipes
`pthread_create()`	Ejecución paralela en el comando parallel
`sigaction()`	Captura e ignora señales como SIGINT
`getcwd(), chdir()`	Implementación de comandos pwd y cd
`open(), close()`	Manejo de archivos para redirecciones

El sistema también implementa manejo de errores utilizando:

- perror() para mostrar mensajes de error del sistema
- Verificación de códigos de retorno de todas las llamadas del sistema
- Validación de entrada de usuarios y parámetros
- Limpieza adecuada de recursos en caso de fallo

PRUEBAS Y RESULTADOS

CASO DE PRUEBA	COMANDO	RESULTADO ESPERADO
Ejecución básica	ls -l	Lista archivos con detalles
Redirección salida	ls > out.txt	Crea archivo con listado
Redirección entrada	cat < input.txt	Muestra contenido del archivo
Proceso background	sleep 5 &	No bloquea el prompt
Comando interno cd	cd /tmp	Cambia al directorio /tmp
Comando interno pwd	pwd	Muestra directorio actual
Comando parallel	parallel "cd /tmp; pwd"	Ejecuta comandos en paralelo
Comando help	help	Lista comandos disponibles
Comando meminfo	meminfo	Muestra información de memoria

EJECUCIÓN DEL PROYECTO

Pasamos a ejecutar el proyecto...

CONCLUSIONES

1. Gestión de Procesos: Se implementó correctamente la creación, ejecución y sincronización de procesos mediante las llamadas del sistema fork(), execvp() y waitpid().

2. Redirección de E/S: El manejo de redirecciones utilizando dup2() demostró cómo los sistemas operativos gestionan los flujos de entrada y salida.

3. Concurrencia: La implementación del comando parallel con pthread ilustró los principios de programación concurrente y sincronización.

4. Arquitectura Modular: El diseño modular facilitó el desarrollo, testing y mantenimiento del código.