INTÉRPRETE DE COMANDOS EN C++ PARA LINUX

Alex Yasmani Huaracha Bellido 2023 - 119027
Jheral Jhosue Maquera Laque 2023 - 119037

INTRODUCCION

El proyecto consiste en una mini-shell desarrollada en C++ para Linux, capaz de interpretar y ejecutar comandos del sistema.

Su objetivo es aplicar conceptos de sistemas operativos como la gestión de procesos, la entrada y salida estándar y la concurrencia.

Además, se implementaron funciones adicionales como tuberías y ejecución paralela mediante hilos POSIX, simulando el comportamiento básico de una shell real.

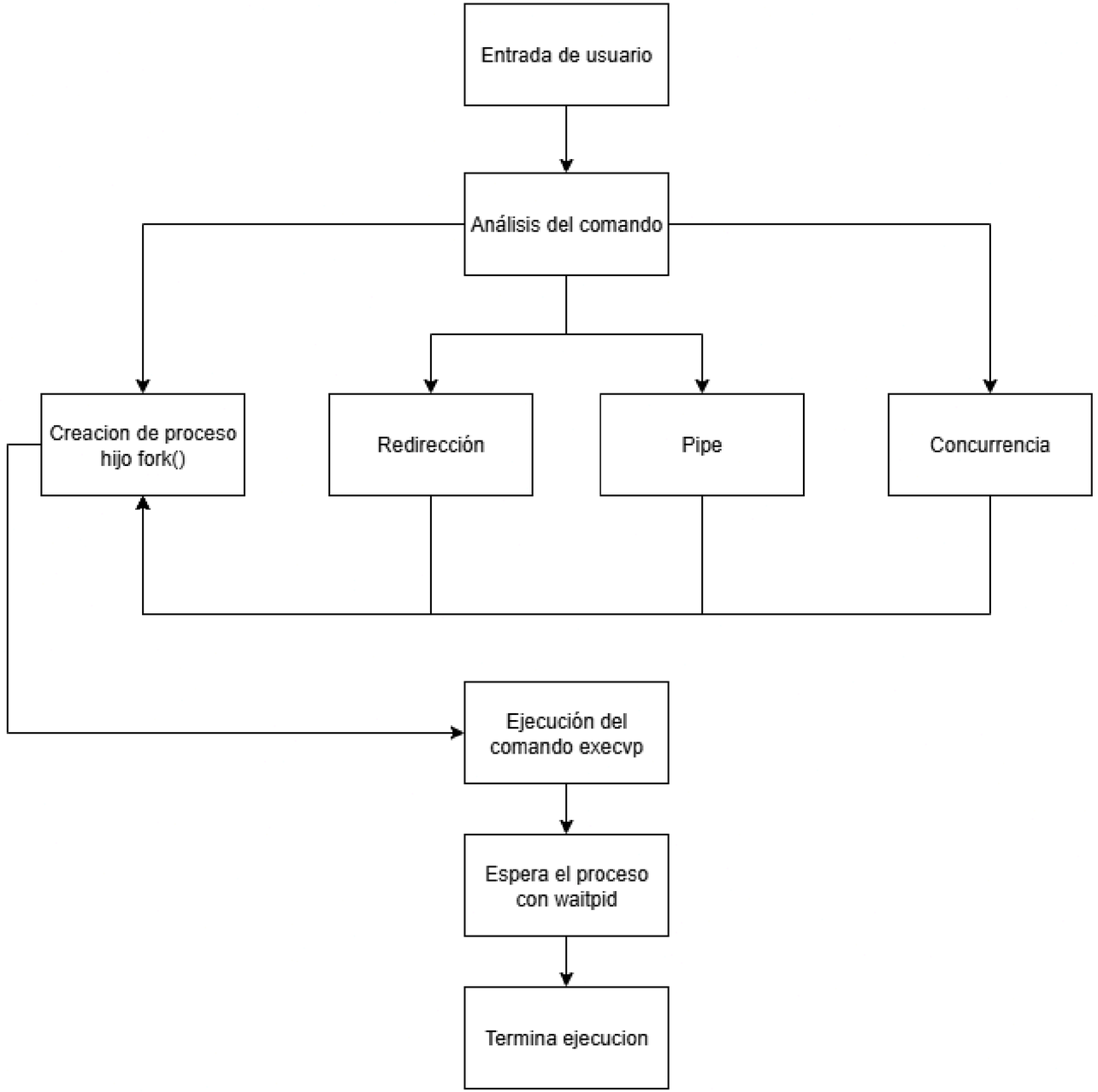
LLAMADAS AL SISTEMA (POSIX) UTILIZADAS

Llamada	Función / Descripción técnica
fork()	Crea un nuevo proceso hijo duplicando el proceso actual. Ambos procesos continúan desde el mismo punto de ejecución, pero con diferentes identificadores de proceso (PID).
execv() / execvp()	Reemplazan el código y el contexto del proceso actual por otro programa ejecutable. No retorna si la ejecución es exitosa. Se usa para lanzar comandos del sistema.
waitpid()	Suspende la ejecución del proceso padre hasta que finalice un proceso hijo específico, permitiendo sincronización entre procesos.
dup2()	Duplica un descriptor de archivo sobre otro (por ejemplo, redirigir la salida estándar STDOUT_FILENO hacia un archivo o tubería).
pipe()	Crea un canal unidireccional que permite la comunicación entre dos procesos (uno escribe y el otro lee). Base de las **tuberías (
pthread_create() / pthread_join()	Crea y sincroniza hilos de ejecución dentro del mismo proceso, permitiendo la ejecución paralela de múltiples comandos.
open() / close()	Abren o cierran archivos en bajo nivel (sin usar fstream). open() devuelve un descriptor numérico; close() libera el recurso.

ARQUITECTURA

Módulo	Funcionalidad principal
main.cpp	Lee la entrada del usuario y envía los comandos al ejecutador
ejecutador.cpp	Analiza el comando, detecta tipo (pipe, redirección, paralelo)
rutas.cpp	Verifica y construye rutas absolutas o relativas
redireccionamiento.cpp	Implementa dup2() para redirigir E/S
tuberia.cpp	Gestiona la comunicación entre procesos mediante pipe()
paralelo.cpp	Implementa concurrencia con pthread_create()
ejecutador.h	Contiene declaraciones y funciones compartidas entre módulos

FUNCIONAMIENTO



ESPECIFICACIONES FUNCIONALES

PROMPT Y COMANDO SALIR

OBJETIVO

Mostrar cómo la MiniShell interactúa con el usuario a través del prompt, y cómo se utiliza el comando salir para terminar la sesión de manera controlada.
Contenido sugerido:

FUNCIONAMIENTO

- *PROMPT INTERACTIVO: MINISHELL>*
- *COMANDO DE SALIDA: SALIR*
- *BANNER INICIAL: ASCII COLORIDO DE BIENVENIDA*

EJEMPLO DE FUNCIONAMIENTO

EJECUCIÓN DE COMANDOS BÁSICOS Y MANEJO DE RUTAS

OBJETIVO:

Permitir que la MiniShell ejecute comandos del sistema, ya sea por ruta relativa (como ls, cat, pwd) o ruta absoluta (como /bin/ls o /usr/bin/whoami).

```
minishell> /bin/ls -l /home/yereeh
total 56
drwxr-xr-x 2 yereeh yereeh  4096 oct 11 12:00 Descargas
drwxr-xr-x 2 yereeh yereeh  4096 sep 17 01:15 Documentos
drwxr-xr-x 4 yereeh yereeh  4096 oct 11 16:39 Escritorio
-rwxrwxr-x 1 yereeh yereeh 16496 sep 17 09:54 hilos
drwxr-xr-x 3 yereeh yereeh  4096 sep 24 10:18 Imágenes
drwxr-xr-x 2 yereeh yereeh  4096 sep 17 01:15 Música
drwxr-xr-x 2 yereeh yereeh  4096 sep 17 01:15 Plantillas
drwxr-xr-x 2 yereeh yereeh  4096 sep 17 01:15 Público
drwx------ 6 yereeh yereeh  4096 sep 23 21:00 snap
drwxr-xr-x 2 yereeh yereeh  4096 sep 17 01:15 Vídeos
```

Funcionamiento:

>Se ingresa al comando

- ls -l → ruta relativa.
- /bin/ls -l → ruta absoluta.

>Se crea un proceso hijo con fork().

>Si el comando es “/”, se valida la ruta con stat().

- Si no, se busca en /bin/ (ruta por defecto).
- Luego se ejecuta con execv().

> Se espera al hijo con waitpid() antes de mostrar el siguiente promp

IMPLEMENTACIÓN DE REDIRECCIÓN DE SALIDA (>)

OBJETIVO:

Permitir que la salida de un comando se almacene en un archivo, en lugar de mostrarse en la consola.

```
minishell> ls -l > salida.txt
salida.txt
1  total 52
2  -rw-r--r-- 1 yereeh yereeh    39 oct 13 16:42 a
3  drwxrwxr-x 2 yereeh yereeh  4096 oct 11 12:03 docs
4  drwxrwxr-x 2 yereeh yereeh  4096 oct 13 17:11 include
5  -rwxrwxr-x 1 yereeh yereeh 36616 oct 13 17:11 minishell
6  -rw-rw-r-- 1 yereeh yereeh     0 oct 11 12:03 README.md
7  -rw-r--r-- 1 yereeh yereeh     0 oct 13 22:26 salida.txt
8  drwxrwxr-x 2 yereeh yereeh  4096 oct 13 17:12 src
```

Funcionamiento:

- El comando ingresado contiene el símbolo > (por ejemplo: ls -l > salida.txt).
- Se detecta la posición del operador > en los argumentos.
- Se abre (o crea) el archivo destino con: **>open(archivo, O_WRONLY | O_CREAT | O_TRUNC, 0644)**;
- Se redirige la salida estándar al archivo mediante: **> dup2(fd, STDOUT_FILENO);**
- Se cierra el descriptor con close(fd) y se ejecuta el comando normalmente.

MANEJO DE ERRORES/ERRNO

OBJETIVO

Explicar cómo la MiniShell detecta y maneja errores, proporcionando información útil al usuario mediante errno y mensajes claros, sin interrumpir la ejecución.

FUNCIONAMIENTO

CAPTURA DE ERRORES COMUNES:

- ***COMANDO VACÍO***
- ***COMANDO NO ENCONTRADO***
- ***ERRORES EN RUTAS ABSOLUTAS O RELATIVAS***
- ***ERRORES EN TUBERÍAS O REDIRECCIONES***

USO DE ERRNO Y PERROR() PARA MENSAJES DETALLADOS

EJEMPLO DE FUNCIONAMIENTO

```
minishell> p | ls
Error al ejecutar el primer comando: No such file or directory
(errno 2: No such file or directory)

docs  include  minishell  README.md  salida.txt  src
minishell> prueba
Error: no se encontró el comando 'prueba': No such file or directory
(errno 2: No such file or directory)
```

EXTENSIONES DE VALOR AGREGADO

COMUNICACIÓN ENTRE PROCESOS CON TUBERÍAS

OBJETIVO

Permitir que la salida de un comando se envíe directamente como entrada de otro comando, simulando el comportamiento de cmd1 | cmd2.

FUNCIONAMIENTO

- *SE DETECTA EL OPERADOR | EN LOS ARGUMENTOS DEL COMANDO.*
- *SE DIVIDE EL COMANDO EN DOS PARTES:*
  - *CMD1: ANTES DEL |*
  - *CMD2: DESPUÉS DEL |*
- *SE CREA UNA TUBERÍA CON PIPE(FD).*
- *SE CREAN DOS PROCESOS HIJOS CON FORK():*
  - *EL PRIMER HIJO REDIRIGE SU SALIDA (STDOUT) HACIA FD[1].*
  - *EL SEGUNDO HIJO REDIRIGE SU ENTRADA (STDIN) DESDE FD[0].*
- *EL PROCESO PADRE CIERRA AMBOS EXTREMOS Y ESPERA A QUE TERMINEN CON WAITPID().*

EJEMPLO DE FUNCIONAMIENTO

```
minishell> ls /home/yereeh/Escritorio/proyecto_so/mini-shell-linux/src | grep cpp
ejecutador.cpp
main.cpp
paralelo.cpp
redireccion.cpp
rutas.cpp
tuberia.cpp
```

EJECUCIÓN CONCURRENTE DE COMANDOS

OBJETIVO

Permitir la ejecución simultánea de varios comandos usando hilos (threads) con la biblioteca POSIX pthreads.

FUNCIONAMIENTO

- EL USUARIO ESCRIBE UN COMANDO CON LA PALABRA CLAVE PARALLEL.
- SE SEPARAN LOS COMANDOS INTERNOS POR ; O ESPACIOS.
- POR CADA COMANDO, SE CREA UN HILO CON PTHREAD_CREATE().
- CADA HILO EJECUTA SU PROPIO COMANDO MEDIANTE LA FUNCIÓN EJECUTAR_COMANDO().
- EL PROGRAMA PRINCIPAL ESPERA A QUE TODOS LOS HILOS TERMINEN CON PTHREAD_JOIN()
- AL FINAL, SE MUESTRA UN MENSAJE INDICANDO QUE TODOS LOS PROCESOS FINALIZARON.

EJEMPLO DE FUNCIONAMIENTO

```
minishell> parallel ls ; pwd ; date ; whoami
a  docs  include  minishell  README.md  salida.txt  src
/home/yereeh/Escritorio/proyecto_so/mini-shell-linux
yereeh
lun 13 oct 2025 22:33:22 -05
Todos los comandos en paralelo han finalizado.
```

CONCLUSIONES

- *Se implementó una MiniShell funcional en C++ para Linux capaz de ejecutar comandos básicos y rutas absolutas o relativas.*
- *Se añadieron extensiones de valor agregado como:*
  - *Tuberías (|) → comunicación entre procesos.*
  - *Concurrencia (parallel) → ejecución simultánea con hilos POSIX.*
- *El proyecto demuestra el uso efectivo de llamadas al sistema POSIX como fork(), execv(), pipe(), dup2() y pthread_create().*
- *Se logró comprender el manejo de procesos, redirección de E/S y la sincronización de hilos en sistemas operativos tipo Unix.*

GRACIAS