Minishell - Guía Completa

Introducción al proyecto Minishell

El proyecto Minishell consiste en crear tu propio intérprete de comandos al estilo Bash. Debes implementar un entorno donde el usuario pueda ejecutar comandos, manejar procesos, pipes, redirecciones y manejar errores adecuadamente.

1. Conceptos básicos de C

- Punteros: Referencias a direcciones de memoria.
- Listas enlazadas: Estructuras dinámicas para manejar colecciones de elementos.
- Memoria dinámica: Uso de malloc/free para gestionar memoria.
- Estructuras: Uso de struct y typedef para definir tipos complejos.

Ejemplo - Lista enlazada simple

```
typedef struct s_node {
    char *data;
    struct s_node *next;
} t_node;
```

Esta estructura permite almacenar cadenas en una lista dinámica.

2. Programación en Unix/Linux

- fork(): Crear procesos hijos.
- execve(): Ejecutar comandos externos.
- wait(): Esperar procesos hijos.
- pipe(): Crear comunicación entre procesos.
- dup2(): Redirigir entradas/salidas.
- signal(): Manejar señales del sistema (Ctrl+C, Ctrl+\).

Ejemplo - fork() y execve()

```
pid_t pid = fork();
if (pid == 0) {
    char *args[] = {"/bin/ls", NULL};
    execve(args[0], args, envp);
} else {
    wait(NULL);
}
```

3. Parsing de comandos

- Separar comandos y argumentos.
- Manejar comillas (', "").
- Expandir variables de entorno ($HOME).

- Detectar y gestionar redirecciones (>, <, >>) y pipes (|).

Ejemplo - Parsing básico

Entrada: ls -l | grep src
Salida:
- Comando 1: ls -l
- Pipe
- Comando 2: grep src

4. Builtins

Debes implementar comandos internos:
- cd
- echo
- pwd
- export
- unset
- env
- exit

5. Gestión de memoria y debugging

- Usa valgrind para evitar fugas de memoria.
- Libera toda la memoria al terminar el programa.
- Controla los errores de sistema (perror, strerror).

6. Consejos finales

- Respeta la Norminette.
- Divide el proyecto en módulos (parser, executor, builtins).
- Documenta tu código para facilitar el debugging.