

Sistemas Operativos

[Práctica 2- MINISHELL]

Brais Cabo FelpeteSergio Pérez Sampedro

Tabla de contenido

[Autores 2](#__RefHeading___Toc600_2814082609)

[Descripción del Código 3](#__RefHeading___Toc602_2814082609)

[Diseño del Código 3](#__RefHeading___Toc604_2814082609)

[Principales Funciones 3](#__RefHeading___Toc606_2814082609)

[Casos de Prueba 4](#__RefHeading___Toc608_2814082609)

[Comentarios Personales 6](#__RefHeading___Toc610_2814082609)

# Autores

**Brais Cabo Felpete**

**Sergio Pérez Sampedro**

# Descripción del Código

**<< MÁXIMO 6 PÁGINAS >>**

## Diseño del Código

**<< EXPLICACIÓN DEL DISEÑO DEL CÓDIGO:**

* **ALGORITMOS UTILIZADOS**
* **ESTRATÉGIA DE EJECUCIÓN DE MANDATOS Y GESTIÓN DE TUBERRÍAS**
* **IMPLEMENTACIÓN DE BACKGROUND**
* **IMPLEMENTACIÓN DE SEÑALES**
* **IMPLEMENTACIÓN DE JOBS+FG**
* **ESTRUCTURAS DE DATOS ESPECÍFICAS**
* **…**

**>>**

**<< NO INCLUIR CÓDIGO FUENTE >>**

**<< SE PUEDE UTILIZAR PSEUDOCÓDIGO O DIAGRAMAS DE APOYO A LA EXPLICACIÓN >>**

## Principales Funciones

**<< DESCRIPCIÓN DE LAS PRINCIPALES FUNCIONES IMPLEMENTADAS INCLUYENDO LA FUNCIÓN PRINCIPAL, MAIN>>**

**<< UTILIZAR EL FORMATO DE LA SIGUIENTE TABLA PARA CADA FUNCIÓN >>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre Función** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 |  |  |  |
|  | Argumento 2 |  |  |  |
|  | … |  |  |  |
| **Variables Locales** | Variable 1 |  |  |  |
|  | Variable 2 |  |  |  |
|  | Variable 3 |  |  |  |
|  | … |  |  |  |
| **Valor Devuelto** |  |  |  |  |
| **Descripción de la Función** |  |  |  |  |

## Casos de Prueba

**Para comprobar si la MINISHELL funcionaba como se esperaba, hemos ido probando los distintos apartados según lo implementábamos. Primero, probamos que al escribirle cosas (sin que ejecutara ningún comando) no desapareciera el prompt. Una vez habíamos logrado esto, implementamos la opción para que ejecutara un mandato con N argumentos. Para probar esta funcionalidad, ejecutábamos mandatos del estilo (ls, find etc) en la MINISHELL y en la SHELL normal, para comprobar que coincidía el output. Posteriormente, probamos la redirección a entrada y salida estandar. Primero probamos la redirección por entrada estándar. Para testear esta funcionalidad usamos el mandato head. Después probamos salida estandar y error, para ello utilizamos el mandato cat, solo redirigiendo salida, solo redirigiendo error y redirigiendo ambas. Es decir todas las combinaciones posibles.**

**Después, implementamos la funcionalidad de poder ejecutar 2 mandatos y enlazados por pipes, con la posibilidad de redirecciones de entrada y salida. Para probarlo, ejecutamos los comandos ls | wc y head | wc con todas las posibilidades de redirección de entrada y salida estándar que hemos mencionado antes. Los resultados obtenidos los íbamos comparando con los que obteníamos de la shell del sistema.**

**Para probar la funcionalidad de ejecutar más de 2 comandos enlazados por pipes y con redirecciones de entrada y salida. Usamos el mismo método que para probar la funcionalidad de dos mandatos pero introduciendo un | wc al final.**

**Después implementamos el mandato cd. Para probarlo, usamos rutas absolutas y relativas. Además al implementar un prompt que te mostraba la ruta actuál, podíamos ir viendo rápidamente si la ruta se cambiaba o no. También usabamos ls para ver si el directorio de trabajo se había cambiado realmente.**

**Posteriormente, implementamos la funcionalidad de jobs y fg. Para probar el jobs, creábamos sleeps en segundo plano e ibamos viendo si se mostraban los que estaban en ejecución. También probamos a mandar al background mandatos con sleep y conectado por un pipe con un find. Para ver si el sleep tenía bloqueado o no al find. A la hora de probar el fg, lo que hemos hecho es ir pasando sleeps al fg a ver si se ejecutan bien. También probamos a mandar a fg mandatos que no estaban al final del array de procesos en segundo plano, con esto pudimos comprobar que cuando un mandato del medio del array era traido a primer plano, se eliminaba del array de jobs correctamente.**

**A la hora de probar la reprogramación de la señal producida por el ctrl + c, lo primero que hicimos era producir la señal en la minishell para ver si se cerraba. Después, probamos si terminaba correctamente los procesos en primer plano. Finalmente, comprobamos que no terminaba aquellos que estaban en backgroud. Para esto íbamos mirando el jobs y viendo si esos procesos seguían ejecutándose.**

**A la hora de probar el mandato exit, lo primero que hicimos era ver si se cerraba la minishell. Después, a la hora de hacer el exit, debugeamos el programa para ver si mataba a todos los hijos correctamente o si por el contrario se quedaban zombies.**

**El último mandato que implementamos fue el umask. Para probar este mandato, fuimos comparando los permisos de archivos creadas con la shell del sistema y la minishell con distintas máscaras. Para estas pruebas nos resultó muy útil el mandato ls -la y el mandato touch para ir creando los archivos.**

# Comentarios Personales

**<< MÁXIMO 2 PÁGINAS >>**

**<< PROBLEMAS ENCONTRADOS>>**

**<< CRÍTICAS CONSTRUCTIVAS >>**

**<< PROPUESTA DE MEJORAS >>**

**<< EVALUACIÓN DEL TIEMPO DEDICADO >>**