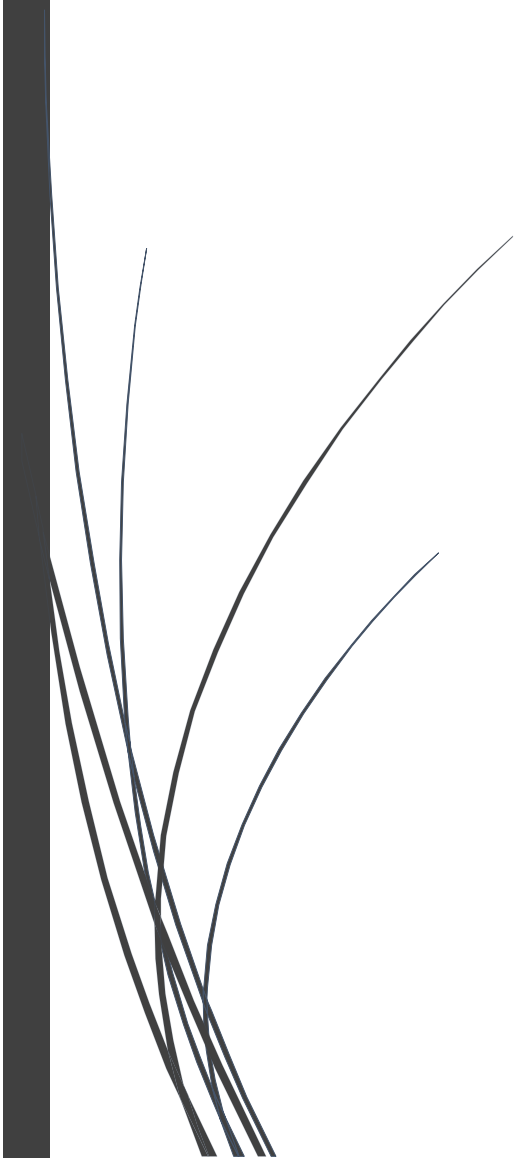
Sistemas Operativos

Práctica 2 MiniShell



Autor: Carlos Santos Morales
GRADO INGENIERIA DE SOFTWARE

Universidad Rey Juan Carlos

Índice de contenidos

Descripción del código 2

MYSHELL.C 2

FUNCIONES USADAS 3

Comentarios personales 4

TIEMPO DEDICADO 4

PROBLEMAS ENCONTRADOS 5

MEJORAS 5

Descripción del código

MYSHELL.C

El desarrollo de esta práctica consistía en simular una pequeña Shell siendo capaz de cumplir los requisitos descritos. La estructura de esta implementación consiste en un bucle infinito mostrando el prompt por pantalla (si no se introdujese ningún comando). En el caso de que se introdujese un comando, el programa creara un hijo mediante la función fork( ) devolviendo el pid del hijo que utlizaré posteriormente. Con esto conseguimos que el que muera tras la ejecución sea el hijo y no el padre y la minishell siga en funcionamiento.

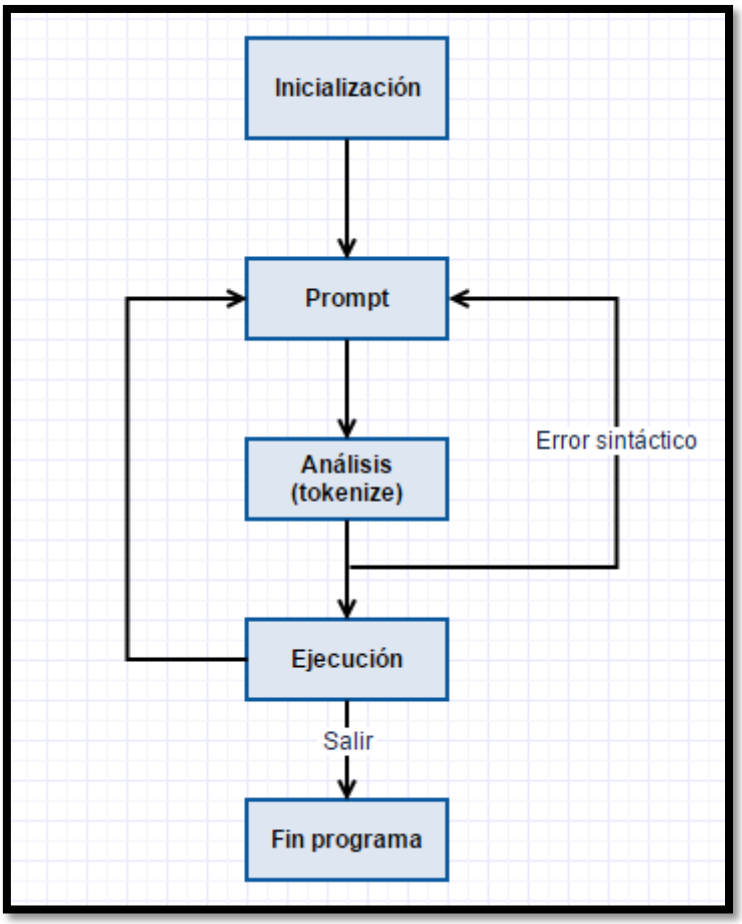
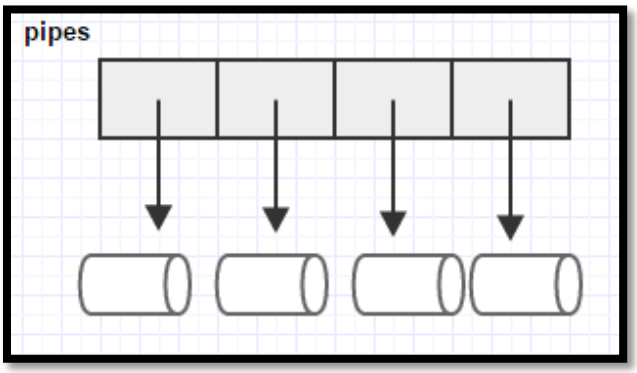


Ilustración 1: *Estructura a grandes rasgos de la minishell.*

FUNCIONES USADAS

Parte del código que merece ser comentado es:

- Matriz de pipes: Creado usando la función malloc (memoria dinámica) con una posición menos que el número de comandos. Dentro de cada posición se crea otro array, también con memoria dinámica, de dos posiciones que se convertirán en pipes con la función pipe( ).



  Una vez creados los pipes hay que diferenciar si está en la primera posición, en la última o en las posiciones centrales para redirigir la salida o la entrada de los comandos. Esto se hace con dup( ) o dup2( ) en mi caso, cerrando la posición 1 ó 2 de la tabla de descriptores y duplicando la posición 1 (escritura) o la 2 (lectura) del pipe. Posterior mente se cerrarán todos los pipes, tanto del hijo como del padre.

- Redirecciones: Si la línea de comandos tiene redirección de salida, de entrada o de error se invoca la función redirecciones(tline * line, int i). Es el mismo procedimiento que con los pipes, invocando a dup2( ) pero previamente abriendo el descriptor de fichero e indicándole el modo de apertura (lectura, escritura, crear fichero y escritura etc).

- Ejecución del comando: La función utilizada para la ejecución del comando es execvp (char *nombre del ejecutable, char *prog[ ]). Esta llamada al sistema la hace el hijo que posteriormente muere.

- Array de pids: Si el comando no es ejecutado en background el padre esperará a cada uno de sus hijos, en cambio, si hay background, el padre mostrará el pid del proceso.

Comentarios personales

TIEMPO DEDICADO

La práctica de minishell, me ha resultado entretenida y bastante laboriosa debido a que necesité bastante tiempo para comprender como deberían funcionar los pipes y qué parte debía ejecutar el hijo y cual no. Para mi esa ha sido la parte con mayor dificultad y la que ha ocupado la mayor parte de mi tiempo dedicado a la práctica.

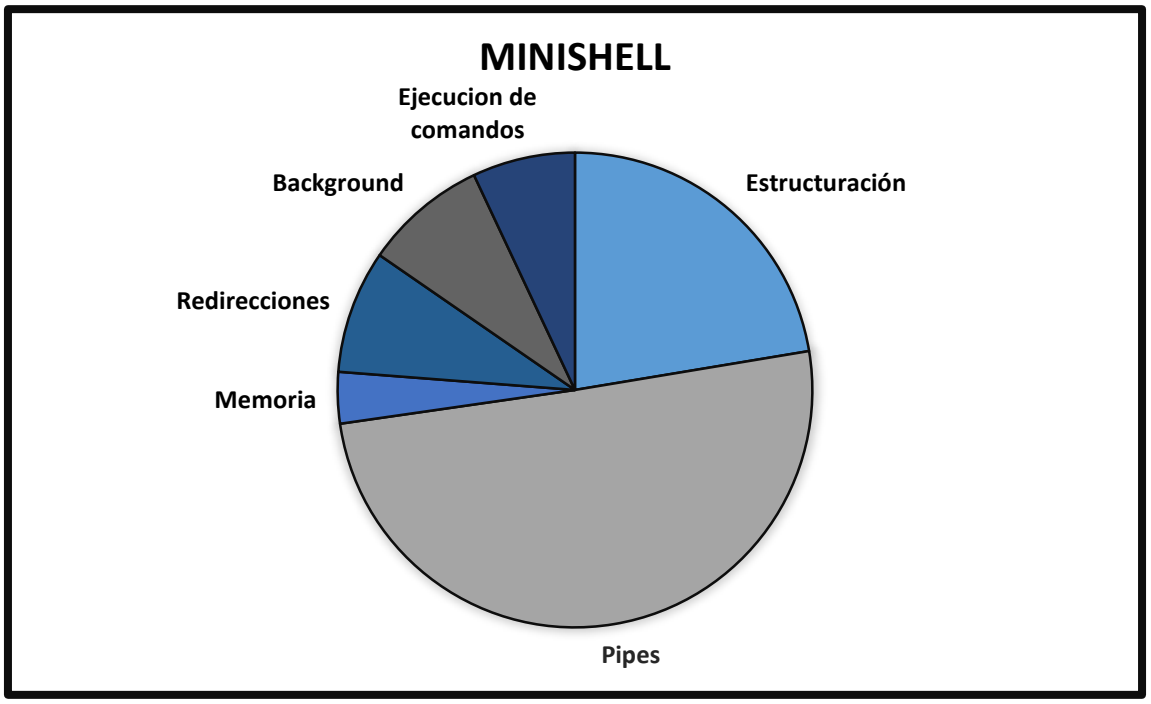


Ilustración 2: Dificultad de la minishell

PROBLEMAS ENCONTRADOS

Algunos de los problemas que he encontrado en esta práctica fueron la creación y el uso de los pipes, ya que tenía que tocar la tabla de descriptores y no me quedaba muy claro que posición del pipe tenía que abrir y cuál no.

Una vez entendido y realizado correctamente la apertura, bien era para lectura o para escritura, al no cerrar todos los pipes no funcionaba correctamente esta parte. Por eso, cuando logré entender que, tanto los hijos como el padre debía cerrar todos y cada uno de los pipes del vector de pipes el problema quedó solucionado y me quedó bastante claro su funcionamiento.

Otro problema que tuve fue el uso de las redirecciones. La ejecución del comando se escribía en el fichero que había creado pero nunca terminaba de escribir en él.

Me di cuenta que las redirecciones las debía hacer los procesos hijos ya que, al ejecutar la llamada al sistema execvp( ) el hijo moriría y con él la escritura en el fichero.

MEJORAS

No tengo muchos comentarios al respecto, la única pequeña mejora que quizá sea de contemplar es la fecha de la entrega de la práctica que, justo estaba en la semana de los exámenes finales.

Aun así, esta práctica me ha parecido muy interesante y me ha servido de aprendizaje de cara al examen final de la asignatura al tener que realizar la gestión de procesos hijos y seguir con el continuo aprendizaje del lenguaje de programación C.