Bitácora Segundo Proyecto Programado

Día Domingo 10 de Noviembre

* Se comenzó a planear como comprobar los comandos que son ingresados y se decidió verificar byte por byte estos comandos, haciendo uso de funciones del lenguaje de programación C , como por ejemplo el printf para imprimir el promp, errores de sistema, entre otros mensajes.

* Se implemento la impresión del promp y luego la lectura de consola , pero surgió el primer problema , a la hora de correr el programa en la terminal primero hacia la lectura de consola y luego de esto imprimía el promp y continuaba imprimiendo el promp antes de cada de mensaje.

* Intentamos borrar el código temporalmente y probar solo el primer printf que es el que imprimía el promp y mostro una violación de segmento (Core Generado).

* Se solucionó dicho problema, usando una llamada a la subrutina tradicional "Display Text" y con solo haber remplazado este primer printf los demás funcionaron de manera correcta.

* Se realizó un buffer para introducir el nombre del archivo que se desea usar, por medio de un ciclo que introdujera byte por byte en el buffer y se lleva un contador para el largo del nombre del archivo.

*Al momento de correrlo en el gdb mostraba que solo guardaba un byte de todo el nombre del archivo, este inconveniente se solucionó incrementando la posición actual en el buffer (los índices) y al hacer esto ya se guardaba correctamente el nombre completo del archivo.

* Se trabajó en la implementación de las rutinas de ayuda y forzado, usando la misma lógica de ir verificando byte por byte los caracteres , en el caso de ayuda si hay una letra incorrecta o no se termino de escribir completamente el parámetro opcional --ayuda se desplegará un mensaje de error de "comando invalido" y en el caso de forzado si hay algún error de este mismo tipo se imprimirá un mensaje de error indicando que se introdujo un tercer parámetro invalido.

Día Martes 12 de Noviembre

* Se investigaron las interrupciones para poder borrar , abrir , renombrar , leer, escribir , crear y cerrar archivos. Esto para poder integrarlo a la lógica del programa . Estos que encontramos son los siguientes:

Borrar : interrupción 10 "Unlink"

Renombrar : interrupción 38 "Rename"

Abrir : interrupción 5 "Open"

Leer : interrupción 3 "Read"

Escribir : interrupción 4 "Write"

Cerrar : interrupción 6 "Close"

Crear : interrupción 8 "Creat"

* Se trabajó en la implementación de borrar e hicimos un buffer para que este contenga la opción digitada por el usuario a la hora de que se le pregunta al usuario si esta seguro de que desea borrar el archivo.

*Se presentó el problema de que si se ponen varios caracteres imprime un mensaje de error muchas veces seguidas, entre más caracteres se pongan mas veces se imprime el mensaje.

* Después de realizar varias pruebas y de analizar el código se logra solucionar el problema haciendo el buffer más grande para que así leerse más caracteres y el problema se solucionó.

* Se empieza a trabajar con el código de Mostrar tratando de usar una lógica similar a la utilizada en Borrar . Pero a la hora de probarlo solo imprimia 4 caracteres del contenido del archivo . Nos dimos cuenta de que el registro que contenia el largo a imprimir se había modificado por un 4 y esta era la razon que solo imprimiera esa cantidad de bytes.

* El problema se solucionó haciendo uso de la pila poder guardar este valor del largo correcto que se estaba perdiendo en el código.

Día Jueves 14 de Noviembre

* Se investigó acerca del funcionamiento de la interrupción "Rename" (38) para poder comenzar con la implementación del comando Renombrar . Al parecer se necesitaba para este comando 2 buffers, uno con el nombre del archivo original el cual se quiere renombrar u otro con el nombre nuevo de este mismo archivo.

* Así que se tuvo que implementar este nuevo buffer y hacer 2 ciclos de nombre archivo (estos ingresan los valores o caracteres byte por byte de cada uno de los nombres a su respectivo buffer.

* Se encuentra un problema ,a la hora de introducir el comando renombrar en consola con los parámetros respectivos, no cambiaba el nombre del archivo , simplemente no hacía nada.

*Se hace uso del debbuger gdb para tratar de encontrar la solución nos dimos cuenta de que el error estaba en el código, al parecer al tratar de hacer la lógica de las rutinas que ingresan los nombres de los archivos a su buffer igual al que ya se había implementado en Borrar y Mostrar , se colocó mal un salto.

* Cuando se compará en el buffer comando si hay un Enter (lo que significa que se termino de introducir el nombre completo del archivo en el buffer) si había un Enter saltaba a renombrar el archivo, pero no debía de ser así ya que aún faltaba el segundo parámetro que sería el nuevo nombre.

* Este se logró solucionar de esta manera:

-Cuando se comparaba la existencia de un Enter si estaba el Enter en lugar de saltar a renombrar directamente el archivo , saltará a imprimir un error el cual le indicará al usuario que aún falta la introducción del nuevo nombre deseado para el archivo y también en esa rutina se comprobara si hay un espacio en blanco (" ") si este existe saltará a llenar el segundo buffer con el segundo nombre.

* En este segundo ciclo a la hora de preguntar o comparar si hay un Enter si saltará a renombrar el archivo ya que ambos buffers contienen lo indicado , Después de esta modificación efectivamente el archivo era renombrado exitosamente.

Día Viernes 15 de Noviembre

* En un principio se planeaba hacer que las secciones del código que verificaban la existencia de los parámetros - - ayuda y - - forzado fueran reutilizables, para que así todos los comandos saltaran a esta rutina para no hacer un código muy extenso .

* Entonces para que la rutina supiera a cual mensaje de ayuda debía de saltar se puso como una marca en cada comando y por medio de comparaciones estas rutinas supieran a cual mensaje de cual comando saltar.

* Pero cuando se agregaron más comandos surgieron errores , en las iteraciones del código por alguna razón las marcas en algunos casos se perdían y no mostraban nada (esto fue descubierto gracias al debugger) y el código comenzó a verse como el tan llamado código espagueti.

* Se discutió esta situación y el grupo de trabajo llegó a la conclusión de que la mejor manera para poder llevar un mejor entendimiento y control sobre el código era hacer rutinas de ayuda y forzado para cada uno de los comandos de manera individual.

* Se tuvo que cambiar la implementación ósea borrar las "marcas" y escribir más código , pero una vez que hicimos esto el código nos era muchísimo más entendible y ya funcionaba de la manera deseada.

Día Sábado 16 de Noviembre

* Se comienza a investigar un poco la manera de como poder implementar el comando copiar , ya que no se encontró ninguna interrupción que copiara directamente como en el caso de Rename y Unlink que hacen su funcionalidad especifica.

* Entonces debimos de combinar varias interrupciones distintas, la idea hasta ahora es la siguiente:

1- Abrir el archivo

2- Con la interrupción creat , crear un nuevo archivo

3- Leer el contenido del primer archivo para guardar su contenido en un buffer

4- Escribir el contenido de ese buffer dentro del segundo archivo creado

5- Cerrar ambos archivos.

* Se comienza la implementación del comando copiar en un principio siguiendo la misma lógica de los comandos anteriores.

* Se siguió la lógica que se planeaba anteriormente pero nos encontramos con el problema de cómo manejar los file descriptors para cada archivo porque los estábamos guardando en registros los mismos y se perdían mas adelante en el código.

* Se decidió hacer uso de la pila pero se tuvieron que hacer varios push y pops para evitar la perdida de estos . En una de las corridas de código cuando se invocaba copiar no hacia nada y era por la misma razón de que en un momento del código se saco uno de los file descriptors y no se volvió a guardar para poder cerrar los archivos correctamente.

* Se logro implementar el comando copiar de manera exitosa en el proyecto.

Día Martes 19 de Octubre

*Se implemento el comando salir de la misma manera que se hizo con los demás comandos , este se comparo byte por byte y si se introdujo bien ira directamente a fin que es una rutina de mantenimiento de pila que a su vez termina el programa.