Instituto Tecnológico de Costa Rica

Proyecto Programado #2

Interprete de comandos

Curso:

Arquitectura de Computadores

Estudiantes:

Rafael Vargas Quintero

Carné# 2013241357

Gonzalo Céspedes Cocozza

Carné# 201211643

Profesor:

Jaime Gutiérrez Alfaro

Semestre II

Año:

2013

**Introducción**

En este documento más que todo se enfocará en la explicación de cómo llegamos o que hicimos para poder llevar a cabo el proyecto.

El proyecto trata de hacer un intérprete de comandos y tiene que realizar los siguientes comandos:

Borrar

Renombrar

Copiar

Comparar

Mostrar

Salir

Estos métodos se explicaran aspectos que hicieron que llegáramos a la solución del problema de los métodos mas adelante en el documento.

**Análisis del problema.**

**¿Cómo resolvimos el problema?**

Para poder explicar bien como resolvimos el problema vamos a explicar cada uno de lo métodos, de esta manera el lector puede tener más claro o puede entender mejor.

**Borrar:**

Este método lo que hace es simplemente borrar un archivo donde se encuentre el programa.

La sintaxis para poder ingresar el comando sería la siguiente:

**cdi>>Borrar[--ayuda] nombre-archivo [--forzado]**

Aquí se recibe como parámetro con el nombre de un archivo y lo elimina del sistema.

También se usa --forzado para evitar la pregunta de seguridad.

**Resolución:**

Lo que primero hicimos en el algoritmo fue llamar a la función sys\_unlink con la línea (mov eax, 10), luego movemos el nombre del archivo a borrar, después se llama al vector de interrupciones y finalmente se salta a una etiqueta llamada "Limpiar" esta etiqueta son un par de ciclos que se encargan de limpiar los buffers de la entradas del usuario.

**Renombrar:**

Este método lo que hace es cambiarle el nombre a un archivo.

La sintaxis para llevar a cabo el comando es:

**cdi>>Renombrar [--ayuda] nombre-original nombre-nuevo [--forzado]**

Recibe como parámetros el nombre del archivo original y el nombre que desea ponerle.

Usar --forzado para evitar la pregunta de seguridad.

**Resolución:**

Utiliza el mismo algoritmo para almacenar los parámetros, una vez teniendo los nombres se llama al vector de interrupciones.

El algoritmo renombrar:

Lo primero es mover el eax, 38 que este es el numero de la interrupción, al ebx y ecx, se mueve el nombre original y el nuevo respectivamente, se llama al vector de interrupciones y se limpian los buffers con la etiqueta Limpiar.

**Copiar:**

Este comando copia la información de un archivo y la inserte en un archivo nuevo.

La sintaxis para el comando:

**cdi>>Copiar [--ayuda] nombre-original nombre-nuevo**

Los argumentos del programa serian el nombre del archivo que será copiado y el nombre que tendrá el nuevo archivo por crear.

**Resolución:**

Se utiliza el mismo algoritmo de ayuda.

Utiliza el mismo algoritmo de almacenar que comparar (Este algoritmo se encarga de almacenar los nombres insertados por el usuario en la línea de comandos).

Como se tiene que leer el archivo se tiene que abrir primero.

Se mueve al eax, 5 (el número de la interrupción), al ebx el nombre del archivo al ecx, la manera en que se quiere abrir (0=leer, 2= leer y escribir).

Al edx si se desea que si el archivo no existe se cree, una vez que se llama al vector de interrupciones el eax tendrá el file descriptor del archivo.

Después de abrir los dos, se tiene que leer uno y escribir en el otro, hay que repetirlo hasta llegar al end of file del archivo fuente.

**Mostrar:**

Muestra en la salida estándar el contenido de un archivo

Sintaxis para comando:

**cdi>>Mostrar [--ayuda] nombre-archivo**

Recibe como parámetro el nombre de un archivo, cuyo contenido será desplegado en la pantalla (salida estándar).

**Resolución:**

Utiliza el mismo algoritmo de ayudar que las funciones anteriores.

El algoritmo de almacenar solo guarda un nombre.

1. Una vez con el nombre abro el archivo.

2. Leo el archivo.

3. Verifico si es eof: si lo es procedo a limpiar los buffer y empezar de nuevo el programa.

4. Si no escribo en pantalla lo que leí del archivo.

5. Vuelvo a 2

**Comparar:**

Toma dos archivos y los compara línea por línea.

**cdi>>Comparar [--ayuda] nombre-archivo-1 nombre-archivo-2**

Dados dos nombres de archivo existentes, los compara línea por línea mostrando como salida los números de línea donde los archivos son diferentes, en caso de ser idénticos tanto para nombre-archivo-1 como para nombre-archivo-2, entonces no debería imprimirse nada (pues no hay ninguna línea en la cual difieran).

**Resolución:**

El algoritmo de comparar se basa en la cantidad de líneas que se haya leído de cada archivo. Dependiendo de la cantidad que haya en cada contador de líneas, se pueden cumplir tres casos;

a. Que se hayan leído más líneas del segundo archivo que del primero. (Para que esto pase una línea tiene que ser más grande que la otra)

b. Que se hayan leído más líneas del primer archivo que del segundo. (Para que esto pase una línea tiene que ser más grande que la otra)

c. Que la cantidad de líneas leídas de los dos archivos, sean iguales hasta el momento.

Si se cumple el caso ‘a’ o el caso ‘b’, tengo que igualar la cantidad de líneas leídas dependiendo del caso. Para hacer esto simplemente, leo de un solo archivo hasta que la cantidad de líneas leídas sea igual.

Si la cantidad de líneas es igual, leo de los dos archivos simultáneamente.

Utiliza el mismo algoritmo de ayuda que las funciones anteriores.

El algoritmo que almacena los nombres de los archivos en una posición de memoria específica, almacena dos nombres a diferencia de otros comandos (Borrar, Mostrar), que almacenan solo uno.

1. Una vez con los nombres de los archivos, se procede a abrir los dos archivos. (Para guardar su respectivo File Descriptor)

2. Inicializo los contadores de caracteres por línea.

3. Pregunto si se cumple alguno de los tres posibles casos.

3.1. Si se cumple ‘a’, leo del primer archivo hasta que la cantidad de líneas leídas sea igual.

3.2. Si se cumple ‘b’, leo del segundo archivo hasta que la cantidad de líneas sea igual.

3.3. Si se cumple ‘c’, sigo al paso 4.

4. Leo de los dos archivos.

5. Verifico si es eof de alguno de los dos archivos.

5.1. Si hay eof, termino el programa.

6. Verifico si lo leído del primer archivo es un salto de línea.

6.1. Si lo es, inicializo de nuevo el contador de carácter por línea del primer archivo. Vuelvo al paso 3.

6.2. Si no lo es, aumento en uno el contador de carácter por línea del primer archivo.

7. Verifico si lo leído del segundo archivo es un salto de línea.

6.1. Si lo es, inicializo de nuevo el contador de carácter por línea del segundo archivo. Vuelvo al paso 3.

6.2. Si no lo es, aumento en uno el contador de carácter por línea del segundo archivo.

8. Verifico si los caracteres leídos son iguales

8.1. Si son iguales, vuelvo al paso 3

8.2. Si no son iguales, Imprimo en pantalla la línea en que el archivo es diferente.

9. Vuelvo al paso 3

**Conclusión**

Después de concluido el trabajo, podemos decir que aprendimos cómo funcionan las líneas de comando y ciertas características que tiene cada una, las cuales pueden ser resultado de la manera en la que se diseñó la línea.

Aprendimos a abrir, leer y escribir en los archivos, y a hacer algoritmos que se aprovechen de éstas utilidades para realizar otro tipo de funciones como lo son (Comparar y Copiar).

**Recomendaciones:**

-A la hora de utilizar una interrupción del sistema, si no se sabe cómo usarla, buscar en la documentación por explicaciones o buscar un ejemplo en la web.

-Tener cuidado con los tamaños de las porciones de memoria no inicializadas. Asegurarse de ser el correcto, si no, puede impedir el correcto funcionamiento del código. Es difícil de encontrar el error.

-Todos los problemas tienen más de una solución, si no sirve la solución que se tiene planeada, deséchela y busque otra solución.

-Ser organizado con los permisos con los que se abren los archivos.