INSTITUTO TECNOLÓGICO DE COSTA RICA

Arquitectura de Computadoras

Grupo 20

Tarea 02

Jordan Andrés Villalobos Campos 2013113269

Pamela Sánchez 2013019495

Nombre del profesor

Jaime Gutiérrez Alfaro.

ll semestre año 2013

Introducción

El proyecto programado a desarrollar es la implementación de una interfaz de línea de comandos, en otras palabras la implementación de un programa donde le sea posible al usuario de dicho programa escribir comandos en este caso en la terminal de Linux.

Una vez ingresado el comando que se quiere llevar a cabo , al presionar la tecla "Enter" el programa validará el comando y los argumentos ingresados, esto para verificar que sean validos y ejecutarlos.

Entre los comandos de los que podrá hacer uso el usuario con este programa están: Borrar un archivo, Mostrar un archivo, Comparar dos archivos , renombrar y copiar un archivo, además del comando salir para salir de este programa.

Cada uno de ellos recibirá distintos argumentos que determinarán su funcionamiento, sin embargo todos con excepción del comando salir, tendrán como argumento opcional y en común --ayuda .

Además en el caso del algunos de los comandos existirá un argumento más y opcional igual que en el caso de --ayuda, este argumento sería el --forzado que ejecutará los comandos inmediatamente.

Este proyecto está diseñado para ser desarrollado e implementado en el lenguaje de programación de bajo nivel Ensamblador de Linux y a la vez ser el segundo proyecto programado de la materia Arquitectura de Computadoras del Instituto Tecnológico de Costa Rica. El proyecto fue plateado para realizarse en parejas con un lapso de entrega de 2 semanas.

El objetivo de este trabajo es que podamos desarrollar de una manera más amplia las destrezas en programación en los lenguajes de bajo nivel, en este caso ensamblador en particular. Además que para llevar a cabo de manera exitosa el trabajo se deberá realizar una búsqueda e investigación acerca de algunas interrupciones y acerca de unos comandos existentes en ensamblador que podrían ser de alguna utilidad en el trabajo.

La idea es poder interpretar la lógica de estos para poder incorporarla y/o adaptarla al código fuente del proyecto .

Análisis del Problema

Este proyecto como se mencionó anteriormente consiste en la implementación de una interfaz de línea de comandos, un programa donde se pueden escribir comandos en un "prompt" en el lenguaje de programación Ensamblador .

La aplicación a desarrollar debe de ser capaz de poder reconocer e interpretar los comandos introducidos por el usuario. Para que esta aplicación pudiera reconocer los comandos ingresados se verificaron cada uno de los caracteres del comando, se reviso y comparo byte por byte con los distintos caracteres que debería incluir algún comando. Si algún carácter es mal introducido (palabra mal escrita) o no existe el comando se le notificara mediante un error esto al usuario.

Ahora bien de la misma manera , por medio de estas comparaciones se verifico si existía la presencia de los parámetros opcionales [- - ayuda] y [- - forzado] y así por medio de saltos llegar a las rutinas indicadas.

Por ejemplo todas estas comparaciones verificaran si solo se introdujo el comando sin el nombre del archivo , esto lanzara un error. Si el usuario introduce el parámetro - - ayuda, lanzara un mensaje en pantalla con la especificación de la funcionalidad de dicho comando por medio del printf y mensajes declarados en la sección .data.

En el caso de que no se introduzca el nombre del archivo al cual se le desea aplicar el comando o en el caso de comparar en el que el usuario deberá de introducir 2 nombres de archivos y no se introduzca alguno de ellos el programa saltara a una sección del código donde se despliega un mensaje donde se le informa al usuario que uno de los parámetros esta faltante.

Por último se verifico si el parámetro - - forzado era introducido , si es así simplemente se ejecuta inmediatamente el comando de no ser así, saltara a una sección del código donde se le pregunta al usuario de igual manera por medio del printf si está seguro que desea realizar dicha acción y se le proporcionan las opciones 'si' o 'no' , si este escoge no volverá a la introducción de comandos , de otra manera ejecutara el comando ingresado.

Para que los comandos se llevaran a cabo efectivamente se investigo acerca de unas interrupciones que serian de utilidad para algunos de estos comandos a implementar.

Por ejemplo para borrar se encontró la interrupción de "Unlink" (la interrupción 10) la cual se mueve a eax y en ebx estaría el nombre del archivo, luego de esto se hizo uso de la interrupción al sistema 80h (se hizo uso de esta en todos los comandos) y una vez hecho esto el código efectivamente eliminaba del sistema el archivo .txt que se indicó.

Para el comando mostrar por otro lado se hizo uso de una combinación de ciertas interrupciones distintas. Primero se debía de abrir el archivo cuyo contenido se deseaba mostrar en pantalla, para esto se hizo uso de la interrupción "Open" (interrupción 5) la cual en su forma de acomodarse en los registros es igual a la anterior, solo que con la excepción de que se guarda en la pila el file descriptor retornado por la misma.

Una vez abierto el archivo se usó la interrupción "Read" (interrupción 3) esto para guardar todo el contenido leído en un buffer declarado anteriormente para esta funcionalidad. Ahora esto debía de ser impreso en pantalla para esto se uso la interrupción "Write" (interrupción 4) con salida estándar en pantalla.

Por último se saca el file descriptor de la pila para cerrar el archivo , con la interrupción "Close" (interrupción 6) , esta última parte se debe a que si no se cierra el archivo podría causar errores en los mismos.

Para el comando renombrar se necesitaba tener el nombre del archivo .txt original para poder operar con este y el nombre que se deseaba fuera el nuevo de este mismo, para esto se hizo uso de la interrupción "Rename" (interrupción 38) esto en eax , y en ebx junto con ecx se mueven el nombre del archivo antiguo y el nuevo nombre en ese orden justamente , hecho esto efectivamente el sistema cambiaba el nombre de dicho archivo por el nuevo indicado por el usuario.

Para el comando copiar que recibe como parámetros el archivo que se desea copiar y el nombre que se le dará a la copia de este, entonces también se debieron de

combinar distintas interrupciones para poder lograr el copiado del archivo. Primero al igual que en el comando mostrar se abrió el archivo el cual se ingreso como primer parámetro con la interrupción "Open" y se guardo en la pila el file Descriptor , luego de esto se creó un nuevo archivo con la interrupción "Creat" con el nombre que se ingreso como segundo parámetro. Y de igual forma se guarda su file descriptor.

Después se lee el contenido del archivo original y se guarda en un buffer para luego ser escrito dentro del nuevo archivo con las interrupciones "Read" y "Write" estas hacen posibles el copiado. Por último lo que quedo por hacer fue cerrar ambos archivos y el comando quedo listo.

Para el comando salir se siguieron las misma lógica que todos los demas comandos, se comparo la introducción de este comando byte por byte y si se introdujo bien este ira a fin que tiene rutinas de mantenimiento de la pila , deja la pila como estaba en un principio y termina el programa

Conclusiones y Recomendaciones

De este último proyecto programado del curso se puede concluir que este tipo de proyectos realizados en el lenguaje ensamblador se necesita de paciencia y mucha concentración ya que este lenguaje es bastante complicado y requiere de su tiempo.

En términos del proyecto en sí , fue un proyecto pues con su grado de dificultad, pero se considero como una prueba de aprendizaje, este proyecto en especifico hay que dedicarle mucho tiempo pues por lo menos en la solución proporcionada por nuestro grupo de trabajo hubo que manejar muchas interrupciones y tener el adecuado control sobre los archivos.

Para este proyecto se usaron funciones del lenguaje de programación C y consideramos que son de utilidad , de cualquier manera se hubiera podido resolver sin estas pero esto queda a gusto de los programadores el uso de estas.

Como grupo de trabajo , si alguien quisiera intentar realizar este proyecto desde un principio entre las recomendaciones que podríamos brindar estarían el hecho de que se planea realizar en grupos ya sea dúos o tríos que preferiblemente trabajen en el proyecto de manera grupal y no individual repartiéndose el trabajo. Esto porque para programar en ensamblador se necesita entender bien el trabajo de todos y si se trabajan de manera apartada puede que a sus compañeros de trabajo les resulte difícil de entender su código o perderse en el mismo.

También aconsejamos llevar al día una bitácora , esto porque a la hora de que el código comienza a dar problemas y a producir errores es sumamente útil poder volver en el tiempo y ver que decisiones de diseño o implementación se tomaron que pudieron producir ese error y que dirección nueva se debe seguir ahora, para esto es la bitácora les ayudará a tener mayor control y manejo sobre los errores que se encuentren en el desarrollo del proyecto.

Una de las últimas recomendaciones que sería propia de proporcionar es, es ir siempre documentando el código que se va escribiendo, de no hacerlo el hilo de las ideas se perderán , haciendo el trabajo más difícil para todo el grupo.

Referencias

Prompt. (2013, 28 de octubre). Wikipedia, La enciclopedia libre. Fecha de consulta: 12:20, noviembre 19, 2013 desde http://es.wikipedia.org/w/index.php?title=Prompt&oldid=70454663.