CI3815 - Organización del Computador

Proyecto # 1 (15%)

Objetivo

Familiarizarse con la representación de datos compuestos y programación estructurada en el ambiente de programación MARS y con el lenguaje ensamblador MIPS.

Definiciones

En el marco de este proyecto, los siguientes términos tendrán las siguientes definiciones:

Archivo: Contiene información de utilidad para un usuario. Su contenido será exclusivamente texto. (no existirán archivos con imágenes o sonidos). Un archivo tiene asociado un nombre, su contenido y su tamaño en bytes.

Directorio: Estructura que almacena los nombres de los archivos y la información de su ubicación en disco. Para efectos del proyecto se trabajará con un único directorio.

Intérprete de comandos: Es un programa que recibe comandos de un usuario y va ejecutándolos uno a uno, pudiendo emitir respuestas según los comandos que sean utilizados.

Cifrado: Es una técnica de seguridad de datos en la que la información es transformada antes de ser almacenada. Asimismo, se debe aplicar un proceso inverso al mostrar la información al usuario.

En este proyecto Ud no tiene que elaborar funciones de cifrado. Éstas serán provistas por el grupo profesoral. Para esto, Ud. debe incluir un archivo llamado funciones.s en su proyecto de la siguiente manera:

ProyectoGrupoXX.s	funciones.s

`.include “funciones.s”`	`.data`
	`funciones_cifrado: .word cifA, cifB`
	`funciones_descifrado: .word desA, desB`
	
	`.text`
	`cifA:`

	cifB: desA: desB:

Donde:

- funciones_cifrado: es un arreglo con las direcciones de varias funciones para cifrar
- funciones_descifrado: es un arreglo con las direcciones de varias funciones para descifrar
- Las funciones de cifrado:descifrado recibirán un carácter y retornarán el resultado del cifrado/descifrado. Ejemplo:

    funCifrado(IN CaracterEntrada; byte; OUT CaracterSalida: byte)

Definición del problema

Los recursos del computador son administrados por un grupo de programas con los cuales los usuarios se comunican para indicarle qué operación se requiere realizar. Los manejadores están definidos en función de las operaciones que le son permitidas realizar al usuario. En este proyecto se requiere que Ud. implemente un Manejador de archivos y un intérprete de comandos.

Actividades

Actividad 1: TAD_Directorio

1. **dir_init(OUT value: entero)**

Descripción	Inicializa la estrututas que componen la representación del TAD.
Pre:	True
Post:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Parámetros:	
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

2. **dir_cp(IN nombre1:String ; nombre2:String ; OUT code: entero)**

Descripción	Copia el contenido del archivo *nombre1* al archivo *nombre2*.
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD se actualizan reflejando la creación del archivo *nombre2* en el directorio actual. El nuevo archivo debe tener como datos una copia de los datos del archivo *nombre1*. El archivo *nombre1* no sufre cambios.
Parámetros:	
nombre1:	nombre del archivo que contiene la información que le serán copiados al archivo *nombre2*.
nombre2:	nombre del archivo a ser creado con la información asociada del archivo *nombre1*.
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

3. **dir_rm(IN nombre1:String ; OUT code: entero)**

Descripción	Borra un archivo
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD se actualizan reflejando la eliminación del archivo *nombre1* del directorio actual.
Parámetros:	
nombre1:	nombre del archivo a ser borrado.
Retorno:	
code:	Valor negativo que representa el código del error o cero si se ejecutó exitosamente.

4. **dir_ren(IN nombre1:String ; IN nombre2:String ; OUT code: entero)**

Descripción	Cambia el nombre de un archivo
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD se actualizan reflejando el cambio de nombre del archivo *nombre1* por *nombre2*.
Parámetros:	
nombre1:	nombre del archivo a ser renombrado.
nombre2:	nombre nuevo.

Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

5. **dir_ls(OUT code: entero)**

Descripción	Imprime por la consola la información de los archivos en el directorio actual. La información se refiere al nombre, el tamaño del archivo y si está cifrado o no.
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD permanecen igual.
Parámetros:	
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

6. **dir_cat(IN nombre:String ; OUT code: entero)**

Descripción	Muestra por consola el contenido de un archivo.
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD permanecen igual.
Parámetros:	
nombre:	nombre del archivo a ser visto.
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

7. **dir_cif(IN nombre:String ; IN func: entero; OUT code: entero)**

Descripción	Se cifra el archivo con nombre y se coloca en un nuevo archivo con el mismo nombre pero agregándole el sufijo “.cif”. Por ejemplo: archi.txt cambia a archi.txt.cif El archivo original desaparece.
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD permanecen igual.
Parámetros:	
nombre:	nombre del archivo a ser cifrado.
func:	Dirección de la función que se debe usar para cifrar.

Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

8. **dir_dcif(IN nombre:String ; IN func: entero; OUT code: entero)**

Descripción	Se descifra el contenido del archivo *nombre* y se coloca en un nuevo archivo con mismo *nombre* pero eliminándole el sufijo ".cif", en caso de tenerlo: note que luego de cifrado un archivo puede perder la extensión mediante la operación renombrar. Por ejemplo: archi.txt.cif cambia a archi.txt El archivo original desaparece
Pre:	Las estructuras que componen la representación del TAD se encuentran inicializadas.
Post:	Las estructuras que componen la representación del TAD permanecen igual. El contenido del archivo *nombre* es descifrado empleando el algoritmo de la clave del cesar. La extensión del archivo es modificada, eliminándosele al nombre inicial, la terminación cif. El archivo cifrado desaparece. (archi.txt.cif -> archi.txt)
Parámetros:	
nombre:	nombre del archivo a ser descifrado.
func:	Dirección de la función que se debe usar para descrifrar.
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó exitosamente.

9. **dir_make(IN nombre:String ; IN contenido: String; OUT code: entero)**

Descripción	Crea un archivo con el contenido especificado.
Pre:	
Post:	Las estructuras que componen la representación del TAD archivo se encuentran inicializadas y contendrán los datos de un nuevo archivo en el sistema.
Parámetros:	
nombre:	nombre del archivo a ser creado.
contenido	Contenido del archivo a crear. Puede asumir que no tendrá más de 100 caracteres.
Retorno:	
code:	Valor negativo que representa el código del error ocurrido o cero si se ejecutó

	exitosamente.

10. **perror(IN code:entero; OUT void)**

Descripción	Imprime por consola la descripción asociada al código de error especificado.
Pre:	true
Post:	true
Parámetros:	
Code	valor negativo que identifica un error ocurrido. *Si* el valor es positivo esta función no debe imprimir nada.
Retorno:	void

Inicialización:

Al inicializar, su programa deberá abrir un archivo de nombre init.txt. Este archivo en su primera línea contendrá el número inicial de archivos con el que su proyecto debe inicializarse, en la siguiente línea vendrá el nombre del primer archivo, luego el número de líneas que el archivo contendrá y luego, una a una, las líneas de ese archivo. Luego vendrá el nombre del siguiente archivo, el número de líneas y sus datos, hasta alcanzar el número de archivos indicado inicialmente.

Ejemplo:

```
2
arc.txt
3
hola soy el archivo de pruebas
y tengo tres lineas
de contenido
otro.txt
4
linea1
linea2
linea3
linea4
```

Actividad 2: Intérprete de Comandos

Usted debe elaborar un programa que lea una línea de la consola, analice lo que el usuario escribe en esa línea y lo interprete para ejecutar aquellos comandos que el usuario desee ejecutar. Luego de ejecutar el comando y mostrar su resultado, el intérprete esperará por el próximo comando del usuario.

Comando a implementar

Comando	**Función según TAD**
cp nombre1 nombre2	dir_cp(nombre1, nombre2)
mv nombre1 nombre2	dir_ren(nombre1, nombre2)
rm nombre	dir_rm(nombre)

ls nombre	dir_ls()
ct nombre	dir_cat(nombre)
ci f1 nombre	dir_cif(nombre, func_f**1**) func_f1 se refiere a la dirección de la función de cifrado colocada en la posición **1** del arreglo *funciones_cifrado.*
dc f1 nombre	dir_dcif(nombre, func_f**1**) func_f1 se refiere a la dirección de la función de descifrado colocada en la posición **1** del arreglo *funciones_descifrado.*
mk nombre	dir_make(nombre, contenido)
ar nombre1	Abre un archivo de nombre nombre1 y empieza a procesar cada línea del archivo como si fuera un comando y muestra los resultados por la consola de MARS
ar nombre1 nombre2	Abre un archivo de nombre *nombre1* y empieza a procesar cada línea del archivo como si fuera un comando los resultados los va almacenando en el archivo de salida *nombre2*

Notas

- Ud debe generar una tabla de códigos de error y sus mensajes correspondientes e incluirla en el informe. En la documentación del código debe incluir los valores de error que retorna cada función.
- Sólo la función perror pueden imprimir por consola mensajes, las otras operaciones se limitarán a devolver un código de error apropiado.
- Cualquier error que pudiera ocurrir durante la ejecución de su programa debe ser tratado y manejado de una forma elegante, cómoda y útil para el usuario del programa.

Recomendaciones

1. Estructure bien su código.
2. Trabaje en forma ordenada e incremental
3. Pruebe que cada una de sus funciones funciona correctamente
4. **Respete** la especificación dada en el presente enunciado: No cambie el nombre de las rutinas NI las definiciones aquí establecidas.
5. Tenga presente que es mejor tener más funciones pequeñas que menos funciones largas.

Entrega

El proyecto debe ser entregado vía aula virtual en el enlace correspondiente. El día jueves de la semana 7 hasta las 11:55pm.

No habrá prórroga y debe constar de:

1. Un informe, de no más de 4 páginas explicando el diseño de su implementación. Las estructuras de datos que UD haya definido, explique sus ventajas y desventajas con respecto a las otras estructuras consideradas.
2. El código **apropiadamente** documentado de su implementación de las actividades del proyecto.