Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Estructuras de Datos y Algoritmos (EDA)
Tarea 3: TreeSO: Un Pequeño Sistema de Archivos

Prof: José M. Saavedra Rondo
Ayudantes: Braulio Torres & Cristóbal Loyola & Francisco Jiménez

Octubre 2022

1. Objetivo

Poner en práctica los conocimientos adquiridos sobre organización da datos basada en árboles para un problema común como la implementación de sistemas de archivos.

2. Descripción

En esta tarea se deberá crear un pequeño sistema de archivos (filesystem), al que llamaremos **treeSO**, que permita organizar carpetas y archivo a través de un conjunto de comandos, muy similar al sistema de archivos Linux. La implementación de treeSO debe utilizar una estrutura árbol (tree). Su sistema de archivos deberá soportar los siguientes comandos:

- **cd**: comando para ubicarse en una carpeta

```
Sintaxis
    cd <carpeta>
Ejemplo
    cd mis_documentos
```

- **ls**: lista el contenido de una carpeta

```
Sintaxis
    ls <carpeta>
Ejemplo
    ls /mis_documentos/tareas
```

- **mkdir**: crear una carpeta.

```
Sintaxis
    mkdir <carpeta>
Ejemplo
    mkdir tareas
```

- **mkfile**: crea un archivo en cierta carpeta

```
Sintaxis
    mkfile <carpeta> <file>
Ejemplo
    mkfile /mis_documentos/tareas   file1
```

- **rm**: borra un archivo o una carpeta. En este último caso, borra la carpeta con todos sus descendientes

```
Sintaxis
    rm <carpeta>
Ejemplo
    rm tareas
```

- **tree**: muestra el contenido actual en formato de árbol.

```
Sintaxis
    tree <carpeta>
Ejemplo
    tree tareas
```

- **find**: busca carpetas o archivos desde cierta carpeta. El resultado es una lista de las carpetas y archivos encontrados.

```
Sintaxis
    find <carpeta> <nombre a buscar>
Ejemplo
    find /mis_documentos file1
```

- **exit**: con este comando salimos de treeSO.

3. Especificaciones

1. El SO treeSO debe estar creado como un árbol genérico.
2. Crear el ADT Item para representar a las carpetas o archivos. La estructuras de Item debe permitir representar:
   - nombre: el nombre de la carpeta o archivo
   - tipo: un entero, 1 si es carpeta o 0 si es archivo.
3. Cada nodo del árbol representa a un Item, entonces podremos saber si se trata de una carpeta o archivo.
4. Un archivo no puede contener ningún nodo hijo. Por lo tanto, un archivo en un nodo terminal.
5. Nombrar al programa principal como "treeSO". Al ejecutar el programa deberá aparecer un indicador del sistema, este será expresado con el símbolo #. Notar que al símbolo del sistema se le debe anteponer el path actual, inicialmente es la raíz (/).

```
$TreeSO
¡Bienvenido a  TreeSO!
¡Autores:  <estudiante1> y <estudiante2>!
/#
```

6. La carpeta raíz será expresa con el símbolo /.

7. Hay dos nombres especiales para referirse a carpetas. Punto (.), se refiere a la carpeta actual y dos puntos (..) se refiere a la carpeta anterior.

```
Ejemplo:
/mis_documentos/tareas# cd ..
/mis_documentos#
```

8. Ahora podremos interactuar con el SO a través de los comandos antes indicados. Por ejemplo:

```
/#mkdir carpeta1
/#mkdir carpeta2
/#mkdir carpeta3
/#ls .
/#carpeta1 carpeta2 carpeta3
/#cd carpeta1
/carpeta1# mkdir carpetaA
/carpeta1# mkdir carpetaB
/carpeta1# cd carpetaA
/carpeta1/carpetaA# mkdir carpetaA1
/carpeta1/carpetaA#tree /
--/
----carpeta1
--------carpetaA
------------carpetaA1
----carpeta2
----carpeta3
#
/carpeta1/carpetaA#cd ..
/carpeta1#
```

El esquema de la Figura 1 muestra el contenido de *treeSO* luego de la ejecución de los comandos anteriores.

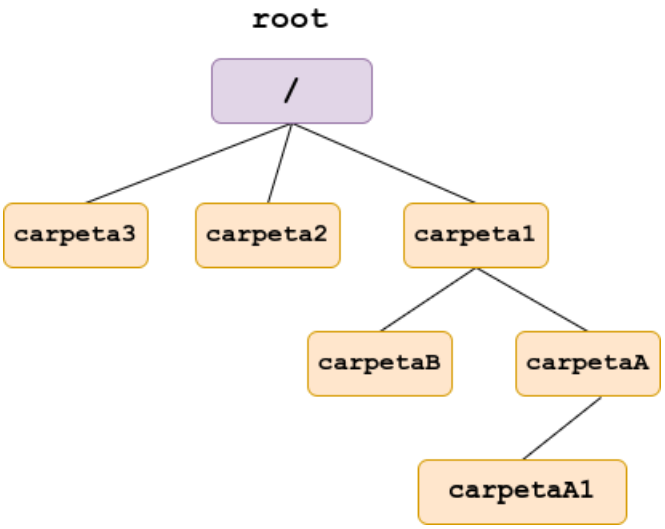

Figura 1: Esquema de treeSO

4. Código Base

Las implementaciones deben ser propias. Solamente pueden ocupar las implementaciones que vienen en el repositorio del curso `https://github.com/jmsaavedrar/eda_cpp/`.

5. Informe

1. **Abstract o Resumen**: es el resumen del trabajo.
2. **Introducción**: aquí se define el problema. (10 %)
3. **Desarrollo**: aquí se describe en detalle el diseño e implementación de los programas necesarios para realizar sus experimentos. (40 %).
4. **Resultados Experimentales y Discusión**: aquí se presentan ejemplos de ejecución de su programa.
5. **Conclusiones**: ideas o hallazgos principales sobre el trabajo. (10 %)

6. Restricciones

1. Pueden trabajar en grupos de 2 estudiantes.
2. Todos los programas deben ser propios, permitiendo solamente utilizar el código disponible en el repositorio del curso `https://github.com/jmsaavedrar/eda_cpp/`.
3. El hallazgo de plagio será penalizado con nota 1.0, para todos los grupos involucrados.
4. Todos las implementaciones deben ser realizadas en C++. El código debe incluir un archivo README con las instrucciones de compilación y ejecución.
5. **La entrega del informe es obligatorio**. Un trabajo sin informe no será calificado, asignando la nota mínima igual a 1.0.

7. Entrega

La entrega se debe realizar por canvas hasta el domingo 28 de octubre, 2022, 23:50 hrs. La entrega debe incluir:

1. Código fuente (en C++), junto a un README con los pasos de compilación.
2. Informe