LABORATORIO DE ESTRUCTURA DE DATOS Y ALGORITMOS

CURSO 2009

SISTEMA DE ARCHIVOS Y CARPETAS

1. Introducción 3
2. Administración de Directorios y Archivos 3
3. Implementación del Simulador 4
4. Intérprete de comandos 4
4.1. Manipulación de Directorios 5
4.1.1. PWD 5
4.1.2. CD Directorio 5
4.1.3. MKDIR Directorio 7
4.1.4. RMDIR Directorio 7
4.1.5. BOUND cota 8
4.1.6. UNBOUND 8
4.1.7. DIR 8
4.1.8. RECDIR 9
4.1.9. COPY 10
4.1.10. PASTE 10
4.2. Manipulación de Archivos 12
4.2.1. CREATE NombreArchivo 12
4.2.2. FILESIZE NombreArchivo 12
4.2.3. LINES NombreArchivo 12
4.2.4. IL NombreArchivo Linea Posicion 12
4.2.5. BL NombreArchivo Posicion Cantidad 13
4.2.6. TYPE NombreArchivo Posicion Cantidad 13
4.2.7. COPYLINES NombreArchivo Poscion Cantidad 14
4.2.8. PASTELINES NombreArchivo Poscion 14
4.2.9. CLIPBOARD 15
4.2.10. DELETE NombreArchivo 15
4.2.11. UNDELETE 15
4.2.12. RECENT 16
4.3. Otros Comandos 17
4.3.1. EXIT 17
5. Ejemplos 17

1. Introducción

Se desea construir un simulador del Sistema de Archivos y Carpetas de un Sistema Operativo, el cual debe implementar un conjunto de comandos básicos para su manejo. Este tendrá una sintaxis similar a la utilizada por la consola de Windows (DOS).

2. Administración de Directorios y Archivos

La estructura de directorios (conocidos como *carpetas* en la jerga de *Windows*) deberá contar con un directorio *RAIZ* (carpeta base), a partir del cual se podrán crear nuevos directorios y archivos. A su vez, estos nuevos directorios podrán contener también nuevos archivos y directorios, permitiendo múltiples niveles en la estructura. En nuestro simulador los archivos y la estructura de directorios se manejarán únicamente a nivel de memoria y no a nivel de disco.

Los archivos que administrará el sistema serán de texto. Un texto se define como una secuencia de líneas de largo acotado, definido por la constante LINEA_MAX. Un *archivo* es identificado por un *nombre*, cuyo largo no puede exceder 15 caracteres, y una *extensión,* de entre 1 y 3 caracteres como máximo. Los caracteres válidos tanto para el nombre como para la extensión serán de tipo alfanumérico {a,b,c,...,z,A,B,C,…,Z,0...9} (diferenciando mayúsculas de minúsculas) y se utilizará un punto para separar el nombre de la extensión. El tamaño de un archivo está determinado por la cantidad de caracteres que contiene.

Los directorios no podrán contar con extensión y serán identificados solamente con un *nombre*, cuyo largo no podrá exceder de 15 caracteres alfanuméricos. La excepción a esta regla es el *directorio RAIZ* (nivel superior de la estructura de directorios) que se denota con el símbolo "/". El tamaño de un directorio en un momento dado está determinado por la suma de los tamaños de los archivos que se encuentran en dicho directorio y en sus subdirectorios. Los directorios pueden tener o no un tamaño máximo (cota) para los mismos. En caso de tener cota, el tamaño de un directorio nunca podrá exceder dicha cota.

Nomenclatura utilizada:

- Un *subdirectorio* es un *directorio* que pertenece, en un nivel inferior, al *directorio* dado.
- Un *subdirectorio hijo* es un *directorio* que pertenece, en un nivel inmediatamente inferior, al *directorio* dado.
- Un *directorio* puede contener, tanto *subdirectorios* (generando una estructura arborescente) como *archivos*.
- En todo momento se está posicionado en un *directorio* específico dentro de la estructura, este es denominado *directorio actual*.
- Un *archivo* es *vacio* cuando no contiene caracteres.
- Un *camino* es una secuencia de *directorios*, donde para cada par consecutivo de la secuencia existe una relación de padre a hijo, entre el primer *directorio* y el segundo.
- Una *ruta* está definida como un *camino* entre dos *directorios*. Esta puede servir para referirse tanto a *archivos* como a *directorios*.
- Una *ruta absoluta* es un *camino* desde la *RAIZ* hasta el *directorio* que se desee referir.
- Los ancestros de un *directorio* son todos los *directorios* que forman parte de la *ruta absoluta* al *directorio*.

Ejemplo:

```
/
├──directorio1
│      ├──archivo11.txt
│      ├──archivo12.cpp
│      ├──archivo13.txt
│      └──directorio11
│            ├──archivo111.h
│            └──archivo112.txt
└──directorio2
       └──directorio21
```

- "directorio1" es un *directorio*, contiene 3 *archivos* ("archivo11.txt", "archivo12.cpp" y "archivo13.txt") y 1 *subdirectorio* ("directorio11")
- a su vez "directorio1" y "directorio11" son *subdirectorios* del directorio RAIZ ("/")
- para referirnos al *archivo* "archivo111.h", si estamos situados en "directorio1" (o sea, éste es el directorio actual), podemos hacerlo mediante la ruta absoluta: "/directorio1/directorio11/archivo111.h" o relativa: "directorio11/archivo111.h"

3. Implementación del Simulador

El ejecutable del simulador tendrá como nombre **SIMEXP**. El mismo debe manejar los archivos y la estructura de directorios en memoria, es decir sin grabar nada a disco.

Al inicio de la ejecución de **SIMEXP** debe aparecer en la pantalla el siguiente mensaje:

```
Bienvenidos a SIMEXP
version 1.0
TecnoInf – Eda

>
```

El símbolo **>** (símbolo de mayor) es el llamado *prompt* del sistema; el mismo indica que **SIMEXP** está listo para aceptar comandos del usuario. Los comandos son los que se presentan en la siguiente sección.

4. Intérprete de comandos

El programa **SIMEXP** lee comandos desde la entrada y los ejecuta. La sintaxis general de los comandos es la siguiente:

```
NombreComando Parámetros …
```

Un comando está formado por un nombre seguido de una lista de parámetros eventualmente vacía. El nombre del comando está separado de los parámetros por **un (1)** espacio. Asimismo cada parámetro se separa del siguiente por **un (1)** espacio.

Cuando la ejecución de un comando finaliza correctamente el sistema responde con el mensaje "OK".

Téngase en cuenta que:

i. Al comenzar, el *directorio actual* es el *directorio RAIZ*, que es el único componente de la estructura (no hay *archivos* ni otros *directorios*). Inicialmente, este *directorio* no tiene cota.
ii. Los nombres de los comandos están dados en mayúsculas.
iii. Se puede asumir que **TODOS** los comandos se ingresan con una sintaxis correcta (cantidad de parámetros, largo de los parámetros, etc.)

iv. Se puede asumir que **TODOS** los identificadores de *archivos* (nombre y extensión) y todos los identificadores de *directorios* (nombre) que se pasan como parámetro a los distintos comandos son correctos de acuerdo a la definición dada para los mismos,
v. Para cada comando se detalla lo siguiente, cuando es necesario:
    - **Asumir:** lo que se puede asumir (no es necesario chequearlo) para un comando,
    - **Controlar:** lo que un comando debe controlar para que tenga efecto sobre el Sistema de Archivos y Carpetas. En caso de que alguna de estas condiciones no se cumpla, el comando despliega en pantalla "ERROR".

En cualquier caso en que la ejecución de un comando no sea satisfactoria, el estado del sistema permanecerá inalterado.

4.1. Manipulación de Directorios

A continuación se describen los comandos que permiten manipular *directorios* en el Sistema de Archivos y Carpetas.

4.1.1. PWD

Este comando muestra el camino desde la *RAIZ* al *directorio actual* siguiendo el formato: "/.../.../dirActual".

Ejemplo: considerar la siguiente estructura de directorios y archivos, con "/directorio2/directorio21" como *directorio actual*.

```
/
├──directorio1
│      ├──archivo11.txt
│      ├──archivo12.cpp
│      ├──archivo13.txt
│      └──directorio11
│            ├──archivo111.h
│            └──archivo112.txt
└──directorio2
       └──directorio21
```

La ejecución del comando **PWD** debe desplegar:

```
/directorio2/directorio21
OK.
```

4.1.2. CD Directorio

Este comando **es el único** que *permite desplazarnos en la estructura de directorios*, definiendo así al nuevo *directorio actual*. El *directorio* **Directorio** indica el *subdirectorio hijo* del *directorio actual* en el cual el sistema deberá posicionarse. Si se utiliza el parámetro *".."* en vez de un nombre de *directorio* el sistema deberá posicionarse en el padre del *directorio actual*. Para regresar al *directorio RAIZ* desde cualquier otro *directorio* se utiliza el parámetro *"/"*.

Asumir:

- No se podrá bajar varios niveles en la estructura con la ejecución de un único comando **CD** (solamente se podrá mover a un *subdirectorio hijo* del *directorio actual*).
- No se podrá subir varios niveles en la estructura con la ejecución de un único comando **CD** (solamente se podrá mover al *directorio padre* del *directorio actual*).

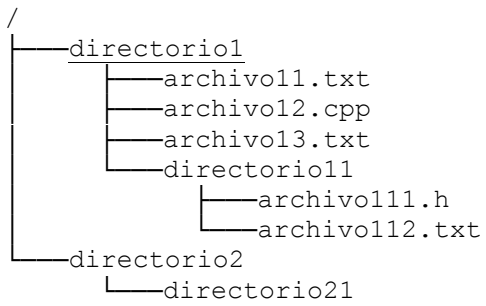
Controlar:

- No se puede ejecutar el comando **CD** con el parámetro *".."* si el *directorio actual* es el *directorio RAIZ.*
- El *directorio* **Directorio** debe ser un *subdirectorio hijo* del *directorio actual.*

Ejemplo: considerando la estructura del ejemplo anterior, si nos encontramos en el *directorio* "/directorio2/directorio21" (*directorio actual*) y queremos movernos al *directorio* "/directorio1":

CD /directorio1 ↵

De esta manera, el *directorio actual* pasará a ser "/directorio1". Este es un ejemplo de *ruta absoluta*, dado que el *directorio* destino está dado desde la *RAIZ*.

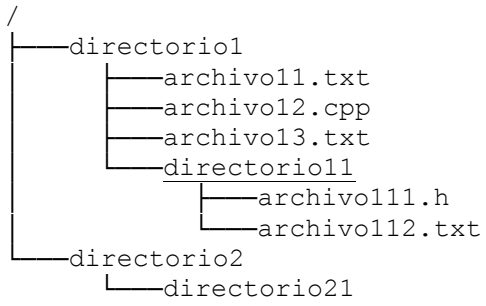


Se debe además poder navegar en la estructura utilizando *rutas relativas*.

Ejemplo: Dado que el *directorio actual* es "/directorio1", podemos hacer que este sea "/directorio1/directorio11" ejecutando el siguiente comando:

CD directorio11 ↵

De esta manera, el *directorio actual* pasará a ser "/directorio1/directorio11".



4.1.3. MKDIR Directorio

Crea un nuevo *directorio* **Directorio** que será *subdirectorio hijo* del *directorio actual*. El *directorio* creado es vacio y no acotado, es decir que su tamaño máximo no está explícitamente restringido.

Controlar:

- En un *directorio* no podrán existir dos *subdirectorios* con el mismo nombre.

Por ejemplo, dada la estructura:

```
/
├──directorio1
│      ├──archivo11.txt
│      ├──archivo12.cpp
│      ├──archivo13.txt
│      └──directorio11
│            ├──archivo111.h
│            └──archivo112.txt
└──directorio2
       └──directorio21
```

Si el sistema se encuentra posicionado en "/directorio1/directorio11", la ejecución del siguiente comando crea un *directorio hijo* de "/directorio1/directorio11":

MKDIR directorio111 ↵

```
/
├──directorio1
│      ├──archivo11.txt
│      ├──archivo12.cpp
│      ├──archivo13.txt
│      └──directorio11
│            ├──archivo111.h
│            ├──archivo112.txt
│            └──directorio111
└──directorio2
       └──directorio21
```

4.1.4. RMDIR Directorio

Elimina el *directorio* **Directorio** que es un *subdirectorio hijo* del *directorio actual*. Todos los *archivos* y *directorios* del directorio **Directorio** serán eliminados.

Controlar:

- El directorio **Directorio** debe ser un *subdirectorio hijo* del *directorio actual.*

Por ejemplo, para la estructura de directorios del comando anterior y estando posicionado en el *directorio RAIZ*, la ejecución del comando:

RMDIR directorio1 ↵

```
/
└──directorio2
       └──directorio21
```

4.1.5. BOUND cota

Actualiza la cota (tamaño máximo) del *directorio actual* con el valor **cota.**

Asumir:

- El valor **cota** es un numero natural.

Controlar:

- El valor **cota** es mayor o igual que el tamaño del *directorio actual.*

4.1.6. UNBOUND

Indica que el *directorio actual* pasa a ser no acotado. Si el *directorio* no tenia cota, la ejecución del comando no tiene efecto.

4.1.7. DIR

Muestra el contenido del *directorio actual* (tanto *subdirectorios* como *archivos*). En primer lugar se listan los *archivos*, en orden alfabético, y luego los *directorios*, también en orden alfabético. El orden será determinado en forma lexicográfica sobre el nombre y la extensión del *archivo* o sobre el nombre en caso de *directorio*.

El formato para mostrar un *archivo* de 200 caracteres de nombre "test.txt" es el siguiente:

```
<arch nombre= "test.txt" tam=200/>
```

El formato para mostrar un *directorio* vacio de nombre "archivos", con una cota de 500 caracteres y un tamaño de 0 caracteres se muestra a continuación:

```
<dir nombre="archivos" tam=0 cota=500>
</dir>
```

Los valores encerrados entre los símbolos '<' y '>' se denominan *etiquetas*. Dentro de cada etiqueta se debe preservar el formato especificado. Es decir, los **únicos** formatos de etiqueta validos son:

- `<dir nombre="nomdir" tam=% cota=%>`
- `</dir>`
- `<arch nombre="nomarch.ext" tam=%/>`

donde el símbolo % representa un natural o la palabra **IND** en caso que el directorio no tenga cota.

Se muestra una etiqueta por línea. Si el *directorio* no está vacío, se muestra su contenido entre las etiquetas <dir ..> y </dir>.

Por ejemplo, sea la siguiente estructura, en la cual se está posicionado en el *directorio RAIZ*:

```
/ (202:240)
├──archivo1.txt (150)
├──directorio1 (52:IND)
│      ├──directorio11 (52:55)
│      │      ├──archivo111.txt (4)
│      │      ├──archivo112.txt (18)
│      │      ├──archivo113.cpp (0)
│      │      └──directorio111 (30:30)
│      │            ├──archivo1111.txt (20)
│      │            └──archivo1112.h (10)
│      └──directorio12 (0:10)
└──directorio2 (0:0)
       └──directorio21 (0:110)
```

en la cual:

- Para cada *directorio* (a continuación y entre paréntesis) se muestra su tamaño y cota (separados por ":").
- Para cada *archivo* (a continuación y entre paréntesis) se muestra su tamaño.
- Si el *directorio* no tiene cota aparece la palabra **IND.**

Es importante tener en cuenta que aunque un directorio no tenga cota, su tamaño puede quedar acotado por la cota de alguno de sus ancestros. Este es el caso del directorio "directorio21"que tiene como cota el valor 110 pero no puede almacenar archivos dado que su padre tiene como cota el valor 0. Una situación similar ocurre con el directorio "directorio1" que no tiene cota pero esta implícitamente acotado, dado que su padre tiene como cota el valor 240.

La ejecución del comando:

DIR ↵

genera la siguiente salida por pantalla, mostrando el contenido del *directorio actual* (la *RAIZ* en este caso).

```
<dir nombre="/" tam=202 cota=240>
   <arch nombre="archivo1.txt" tam=150/>
   <dir nombre="directorio1" tam=52 cota=IND>
   </dir>
   <dir nombre="directorio2" tam=0 cota=0>
   </dir>
</dir>
OK.
```

Notas:

- El formato de salida **debe ser exactamente igual** al del ejemplo anterior.
- Cada línea ubicada entre las etiquetas <dir …> y </dir> debe estar indentada 3 espacios con respecto a la posición en pantalla de dichas etiquetas.
- Los *subdirectorios* "directorio1" y "directorio2" aparecen vacios porque el comando **DIR solamente** muestra la información del contenido del *directorio actual.*

4.1.8. RECDIR

Muestra la estructura de directorios a partir del *directorio actual*, organizada de la siguiente manera: primero se listan los *archivos* del *directorio actual* y luego el contenido de cada uno de los *subdirectorios* siguiendo el mismo procedimiento (recursivamente). Tanto el listado de *archivos* como el de *directorios* deben realizarse en orden lexicográfico.

Siguiendo con el ejemplo anterior y suponiendo que el *directorio actual* es la *RAIZ*, la ejecución del comando:

RECDIR ↵

genera la siguiente salida por pantalla, mostrando el contenido del *directorio actual* (la *RAIZ* en este caso).

```
<dir nombre="/" tam=202 cota=240>
   <arch nombre="archivo1.txt" tam=150/>
   <dir nombre="directorio1" tam=52 cota=IND>
      <dir nombre="directorio11" tam=52 cota=55>
         <arch nombre="archivo111.txt" tam=4/>
         <arch nombre="archivo112.txt" tam=18/>
         <arch nombre="archivo113.cpp" tam=0/>
         <dir nombre="directorio111" tam=30 cota=30>
            <arch nombre="archivo1111.txt" tam=20/>
            <arch nombre="archivo1112.h" tam=10/>
         </dir>
      </dir>
      <dir nombre="directorio12" tam=0 cota=10>
      </dir>
   </dir>
   <dir nombre="directorio2" tam=0 cota=0>
      <dir nombre="directorio21" tam=0 cota=110>
      </dir>
   </dir>
</dir>
OK.
```

Nota:

- El formato de salida **debe ser exactamente igual** al del ejemplo anterior.
- Cada línea ubicada entre las etiquetas <dir …> y </dir> debe estar indentada 3 espacios con respecto a la posición en pantalla de dichas etiquetas.

4.1.9. COPY

Copia y almacena de manera adecuada, los *archivos* del *directorio actual*. Se guarda únicamente la información del último comando **COPY** ejecutado.

4.1.10. PASTE

Pega, en el *directorio actual*, la estructura de directorios que se copió anteriormente mediante la ejecución del comando **COPY**. Si no se había ejecutado nunca el comando **COPY** entonces el comando **PASTE** no tiene efecto.

Si un *archivo* a pegar tiene el mismo nombre y extensión que un *archivo* en la estructura de directorios donde se ejecuta el comando **PASTE**, este se sobrescribe con el *archivo* a pegar.

Controlar:

- Si no se sobrescriben *archivos*, la suma del tamaño del *directorio* a pegar mas el tamaño del *directorio* donde se está posicionado el ejecutar el comando **PASTE** no debe exceder la cota de este último. Si se sobrescriben *archivos* se debe restar a la suma anterior la suma de los tamaños de los *archivos* sobrescritos.
- Si no se sobrescriben *archivos*, para cada *ancestro* del *directorio actual*, la suma del tamaño del *directorio* a pegar mas el tamaño del *ancestro* no debe exceder la cota de este último. Si se sobrescriben *archivos*, se debe restar a la suma anterior la suma de los tamaños de los *archivos* sobrescritos.

Por ejemplo, sea la siguiente estructura, en la cual se está posicionado en el *directorio* "directorio11" y se ejecuta el comando **COPY:**

```
/ (4:240)
├──directorio1 (4:IND)
│      ├──directorio11 (52:55)
│      │      ├──archivo111.txt (4)
│      │      └──directorio111 (0:10)
│      └──directorio12 (0:10)
└──directorio2 (0:280)
       ├──directorio11 (0:10)
       │      └──archivo111.txt (0)
       └──directorio21 (0:110)
```

Si en algún momento posterior se cambia al *directorio* "directorio2" y se ejecuta el comando:

PASTE ↵

la estructura de directorios es:

```
/ (8:240)
├──directorio1 (4:IND)
│      ├──directorio11 (52:55)
│      │      ├──archivo111.txt (4)
│      │      └──directorio111 (0:10)
│      └──directorio12 (0:10)
└──directorio2 (4:280)
       ├──directorio11 (10:10)
       │      └──archivo111.txt (0)
       ├──archivo111.txt (4)
       └──directorio21 (0:110)
```

Notas:

- El comando **PASTE** se pudo ejecutar correctamente (realizó el pegado) porque el *directorio* "directorio2" tiene una cota suficiente como para contener la cantidad de información a pegar (y lo mismo sucede con sus *ancestros*).
- Se actualiza el tamaño del *directorio* donde se ejecutó el comando y lo mismo para los *ancestros.*

4.2. Manipulación de Archivos

A continuación se describen los comandos que permiten manipular *archivos* en el Sistema de Archivos y Carpetas. Estos comandos se ejecutan sobre el *directorio actual.*

4.2.1. CREATE NombreArchivo

Crea un nuevo *archivo vacio* en el *directorio actual.*

Controlar:

- Si existe un *archivo* en el *directorio actual* con el mismo nombre y extensión que el *archivo* **NombreArchivo**, se elimina el *archivo* existente y se crea uno nuevo.

4.2.2. FILESIZE NombreArchivo

Muestra la cantidad de caracteres (incluidas las comillas dobles) que contiene el *archivo* **NombreArchivo**.

Controlar:

- El *archivo* **NombreArchivo** debe existir en el *directorio actual.*

4.2.3. LINES NombreArchivo

Muestra la cantidad de líneas que contiene el *archivo* **NombreArchivo**.

Controlar:

- El *archivo* **NombreArchivo** debe existir en el *directorio actual.*

4.2.4. IL NombreArchivo Linea Posicion

Inserta una nueva línea en la línea, cuyo número es, **Posicion** del *archivo* **NombreArchivo**. Las líneas del *archivo* **NombreArchivo** que se encuentren en una posición mayor o igual a **Posicion** pasan a ocupar una posición más hacia adelante.

Asumir:

- Si el *archivo* **NombreArchivo** tiene **n** líneas, las posiciones parámetro válidas podrán variar entre **1** y **n+1**.
- El largo de la línea a insertar esta acotado por la constante ***LINEA_MAX***.

Controlar:

- Si el *directorio* al cual pertenece el *archivo* **NombreArchivo** tiene cota, la inserción de la línea en el *archivo* **NombreArchivo** no puede hacer que se sobrepase dicha cota.
- La inserción de la línea **Linea** en el *archivo* **NombreArchivo** no puede hacer que se sobrepase la cota de los *ancestros* del *directorio actual*.
- El *archivo* **NombreArchivo** debe existir en el *directorio actual.*

Por ejemplo, considere que el *archivo* "texto.txt" contiene la siguiente información:

"informacion de linea 1"
"linea 2 con mas información"

la ejecución del comando:

IL texto.txt "nueva línea 2" 2 ↵

hace que el contenido del *archivo* "texto.txt" sea:

"informacion de linea 1"
"nueva linea 2"
"linea 2 con mas información"

4.2.5. BL NombreArchivo Posicion Cantidad

Elimina **Cantidad** líneas del *archivo* **NombreArchivo**, a partir de la posición (número de línea) **Posicion**. Las líneas del *archivo* que se encuentren, originalmente, en posiciones posteriores a (**Posicion** + **Cantidad** - 1) se ubicarán, luego de la ejecución del comando, a partir de la posición **Posicion.**

Asumir:
- **Posicion** y **Cantidad** son números positivos.
- si el *archivo* **NombreArchivo** tiene **n** líneas, (**Posicion** + **Cantidad** – 1) ≤ **n**.

Controlar:
- el *archivo* **NombreArchivo** debe existir en el *directorio actual.*

Por ejemplo, considere el *archivo* "texto.txt" con el contenido del final del ejemplo anterior. La ejecución del comando:

BL texto.txt 1 2 ↵

hace que el contenido del *archivo* "texto.txt" sea:

"linea 2 con mas información"

4.2.6. TYPE NombreArchivo Posicion Cantidad

Muestra **Cantidad** líneas, desplegando cada una de ellas en pantalla, a partir de la posición (número de línea) **Posicion** del *archivo* **NombreArchivo**.

Asumir:
- **Posicion** y **Cantidad** son números positivos.
- Si el *archivo* **NombreArchivo** tiene **n** líneas, (**Posicion** + **Cantidad** – 1) ≤ **n**.

Controlar:
- El *archivo* **NombreArchivo** debe existir en el *directorio actual.*

4.2.7. COPYLINES NombreArchivo Poscion Cantidad

Copia **Cantidad** líneas del *archivo* **NombreArchivo**, a partir de la posición (numero de línea) **Posicion**. Esto es, guarda las líneas copiadas en un porta-papeles. Se guarda únicamente la información del último comando **COPYLINES** ejecutado.

Asumir:

- **Posicion** y **Cantidad** son valores positivos.
- Si el *archivo* **NombreArchivo** tiene **n** líneas, (**Posicion** + **Cantidad** – 1) ≤ **n**.

Controlar:

- El *archivo* **NombreArchivo** debe existir en el *directorio actual.*

4.2.8. PASTELINES NombreArchivo Poscion

Pega líneas (desde el porta-papeles), previamente copiadas con la ejecución del comando **COPYLINES**, a partir de la posición (numero de línea) **Poscion** en el *archivo* **NombreArchivo**. Las líneas del *archivo* **NombreArchivo** que se encuentren, originalmente, en una posición mayor o igual a **Posicion**, pasarán a ocupar posiciones más hacia adelante (a partir de la finalización de las líneas pegadas). Si el porta-papeles está vacío el comando no tiene efecto.

Asumir:

- Si el *archivo* **NombreArchivo** tiene **n** líneas, las posiciones parámetro válidas podrán variar entre **1** y **n+1**.

Controlar:

- El *archivo* **NombreArchivo** debe existir en el *directorio actual*.
- Si el *directorio* al cual pertenece el *archivo* **NombreArchivo** tiene cota, el pegado de las líneas del porta-papeles en el *archivo* **NombreArchivo**, no puede hacer que se sobrepase dicha cota.
- El pegado de las líneas del porta-papeles en el *archivo* **NombreArchivo** no puede hacer que se sobrepase la cota de los *ancestros* del *directorio actual*.

Por ejemplo, considere que el archivo "texto.txt" contiene:

""informacion de linea 1"
"linea 2 con mas información"
"linea3"
"linea 4 y ultima"

Se ejecuta el comando

COPYLINES texto.txt 3 2 ↵

hace que se almacenen en el porta-papeles las líneas:

"linea3"
"linea 4 y ultima"

Considere el *archivo* "texto2.txt", el cual contiene:

"informacion de linea 1 de texto2"
"linea 2 con mas informacion de texto2"
"linea3 de texto2"
"linea 4 y ultima de texto2"

La ejecución del comando:

PASTELINES texto2.txt 1 ↵

"linea3"
"linea 4 y ultima"
"informacion de linea 1 de texto2"
"linea 2 con mas informacion de texto2"
"linea3 de texto2"
"linea 4 y ultima de texto2"

4.2.9. CLIPBOARD

Muestra el contenido del porta-papeles, desplegando una línea, en pantalla, por cada línea contenida en el porta-papeles. Si el porta-papeles está vacío no se muestra nada en pantalla.

4.2.10. DELETE NombreArchivo

Elimina el *archivo* **NombreArchivo** del *directorio actual*.

Controlar:
- El *archivo* **NombreArchivo** debe existir en el *directorio actual*.

4.2.11. UNDELETE

Restaura el último *archivo* eliminado en el *directorio* del cual fue borrado. En cualquier caso la ejecución del comando **UNDELETE** elimina del historial el último *archivo* eliminado.

Se debe disponer de un historial (potencialmente infinito) donde se guardan los *archivos* eliminados.

Controlar:
- La *ruta absoluta* del *archivo* a restaurar debe existir en el Sistema de Archivos y Carpetas.
- Si en la *ruta absoluta* del *archivo* a restaurar ya existe un *archivo* con ese mismo nombre y extensión, éste se sobrescribe.
- Si el tamaño (al momento de ejecutar el comando **UNDELETE**) del *directorio* del cual fue borrado el *archivo* no permite su restauración, únicamente se elimina el *archivo* del historial.

4.2.12. RECENT

Muestra nombre y extensión de, a lo sumo, los últimos **MAX_RECENT** (constante natural dada), *archivos* distintos (con diferente, nombre, extensión o *ruta absoluta*) modificados. Cada uno de estos nombres se imprime en una nueva línea de pantalla junto con su *ruta absoluta*, empezando por el *archivo* más recientemente modificado.

Sólo se consideran modificados los *archivos* pasados como parámetro a cualquiera de los siguientes comandos (si estos, terminaron su ejecución satisfactoriamente): **CREATE, IL, BL, PASTELINES, DELETE, UNDELETE**. Notar que no tendrán en cuenta los *archivos* que se han reemplazados por la ejecución del comando PASTE.

Por ejemplo, considerando que la estructura de *directorios* es la siguiente y que el *directorio actual* es "directorio11":

```
/ (35:240)
├──archivo3.txt (21)
├──directorio1 (14:IND)
│      ├──archivo2.txt (10)
│      ├──directorio11 (4:55)
│      │      ├──archivo111.txt (4)
│      │      └──directorio111 (0:10)
│      └──directorio12 (0:10)
└──directorio2 (0:200)
       └──directorio21 (0:100)
```

Suponiendo que la constante **MAX_RECENT** tiene como valor 2 y que se ejecutan los siguientes comandos:

IL archivo111.txt "otra línea mas" 2 ↵
CD .. ↵
BL archivo2.txt 2 ↵
CD .. ↵
IL archivo3.txt "olasas 1" ↵
BL archivo3.txt 1 ↵
RECENT ↵

se imprime en pantalla:

```
/archivo3.txt
/directorio1/archivo2.txt
OK
```

4.3. Otros Comandos

4.3.1. EXIT

Este comando termina la ejecución del comando **SIMEXP.**

5. Ejemplos

A continuación se mostrará una serie de ejemplos de ejecución de comandos, se asume que la estructura contiene inicialmente solo el *directorio RAIZ* y que no hay *archivos*. El estado de la estructura que se alcanza al ejecutarse completamente un ejemplo se mantiene al inicio del ejemplo siguiente.

El estudiante puede obtener los archivos correspondientes a los ejemplos (ejemplo*n*.txt) con sus respectivas salidas (salida*n*.txt) en un archivo *zip*, desde la sección laboratorio del sitio web del curso. Los ejemplos del archivo *zip* incluyen al principio los comandos necesarios para dejar el Sistema de Archivos y Carpetas en el mismo estado que quedó luego de ejecutar el ejemplo inmediatamente anterior.

EJEMPLO 1

```
> CREATE texto1.txt
OK.
> IL texto1.txt "linea1_1" 1
OK.
> IL texto1.txt "linea1_2" 1
OK.
> IL texto1.txt "linea1_3" 3
OK.
> CLIPBOARD
OK.
> LINES texto1.txt
3
OK.
> FILESIZE texto1.txt
30
OK.
> PASTELINES texto2.txt 1
ERROR.
> CREATE texto2.txt
OK.
> PASTELINES texto2.txt 1
OK.
> FILESIZE texto2.txt
0
OK.
> TYPE texto1.txt 2 2
"linea1_1"
"linea1_3"
OK.
> COPYLINES texto1.txt 2 2
OK.
> CLIPBOARD
"linea1_1"
"linea1_3"
OK.
> IL texto2.txt "linea2_1" 1
OK.
> PASTELINES texto2.txt 1
OK.
> EXIT
OK.
```

EJEMPLO 2

```
> TYPE texto2.txt 1 3
“linea1_1”
“linea1_3”
“linea2_1”
OK.
> BL texto3.txt 1 1
ERROR.
> BL texto2.txt 2 1
OK.
> TYPE texto2.txt 1 2
“linea1_1”
“linea2_1”
OK.
> TYPE texto2.txt 1 2
“linea1_1”
“linea2_1”
OK.
> CREATE texto3.txt
OK.
> COPYLINES texto2.txt 1 2
OK.
> PASTELINES texto3.txt 1
OK.
> BL texto3.txt 2 1
OK.
> IL texto3.txt "bola_3" 1
OK.
> TYPE texto3.txt 1 2
“bola_3”
“linea1_1”
OK.
> EXIT
OK.
```