Arquitectura de Computadoras

Parcial de práctica 2021

Visión general

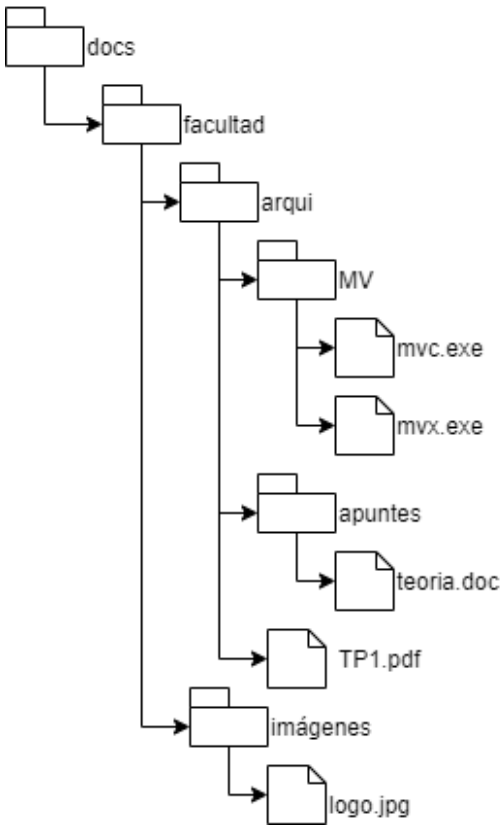
Común a todos los exámenes.

Un sistema de archivos es representado por un árbol. Cada nodo del árbol puede ser un archivo o un directorio (carpeta). Los nodos directorios pueden tener hijos que pueden ser archivos o directorios.

Dentro de cada nodo del árbol se encuentran los metadatos del archivo/directorio (Nombre, fecha de creación, fecha de modificación , tamaño, etc).

Cada archivo o directorio está asociado a un usuario llamado **owner** (el dueño del archivo) y a un grupo (**group**) al que pertenece el archivo.

Además el sistema de archivos tiene una (o varias) estructura(s) de datos donde se organizan los usuarios y los grupos. Cada usuario pertenece a 1 o más grupos. La información es coherente y simétrica, si un usuario pertenece a un grupo, ese grupo contendrá al usuario. No puede haber 2 usuarios con el mismo nombres, ni 2 grupos con el mismo nombre.

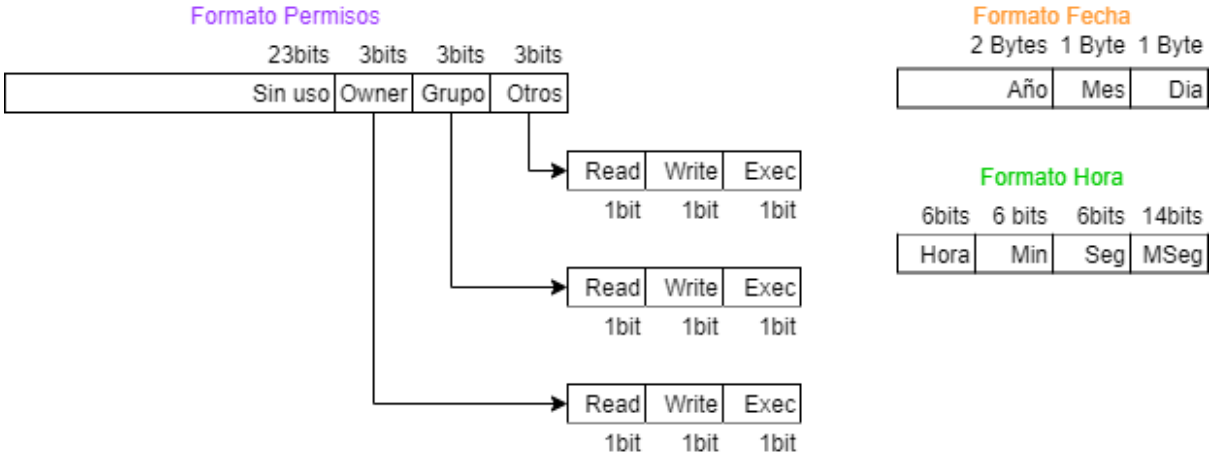


Nodo Arbol
Id Nombre Archivo/Directorio Puntero a Metadatos Listahijos[] Puntero a nodo árbol Group_ID Owner_ID FechaCreacion HoraCreacion FechaModificacion HoraModificacion Permisos Tamaño (en bloques) ListaBloques[]

Nodo Usuario
Id Nombre Grupos[]

Nodo Grupos
Id Nombre Usuarios[]

Tipos de Datos



Formato de fecha: 1 Celda de 4 bytes, en los cuales son, de más significativos a menos significativos: 2 bytes para el año, 1 byte para el mes (1 a 12) y 1 byte para el día (1 a 30).

Formato de hora: 1 celda de 32 bits, en los cuales son, de más significativo a menos significativos: 6 bits para la hora (0 a 23), 6 bits para los minutos (de 0 a 59), 6 bits los segundos (0 a 59) y 14 bits para milésimas de segundos (0 a 999).

Formato de Permisos:
Representan el acceso que puede tener un usuario a un archivo o directorio. Es un atributo propio del archivo o directorio (carpeta). Consta de 1 celda donde solo se utilizan los 9 bits menos significativos.

De más significativo a menos significativo son: 3 bits para el **owner**, 3 para **grupo** y 3 bits para **otros** usuarios.

A su vez cada permiso se subdivide en 1 bit para acceso de lectura, 1 bit para acceso escritura y 1 bit para acceso de ejecución.
Cuando el bit tiene valor 1 significa que el permiso está otorgado, cuando está en 0 el permiso está denegado.

Se recomienda el uso de sistema de numeración octal para trabajar con permisos.
Ejemplos:

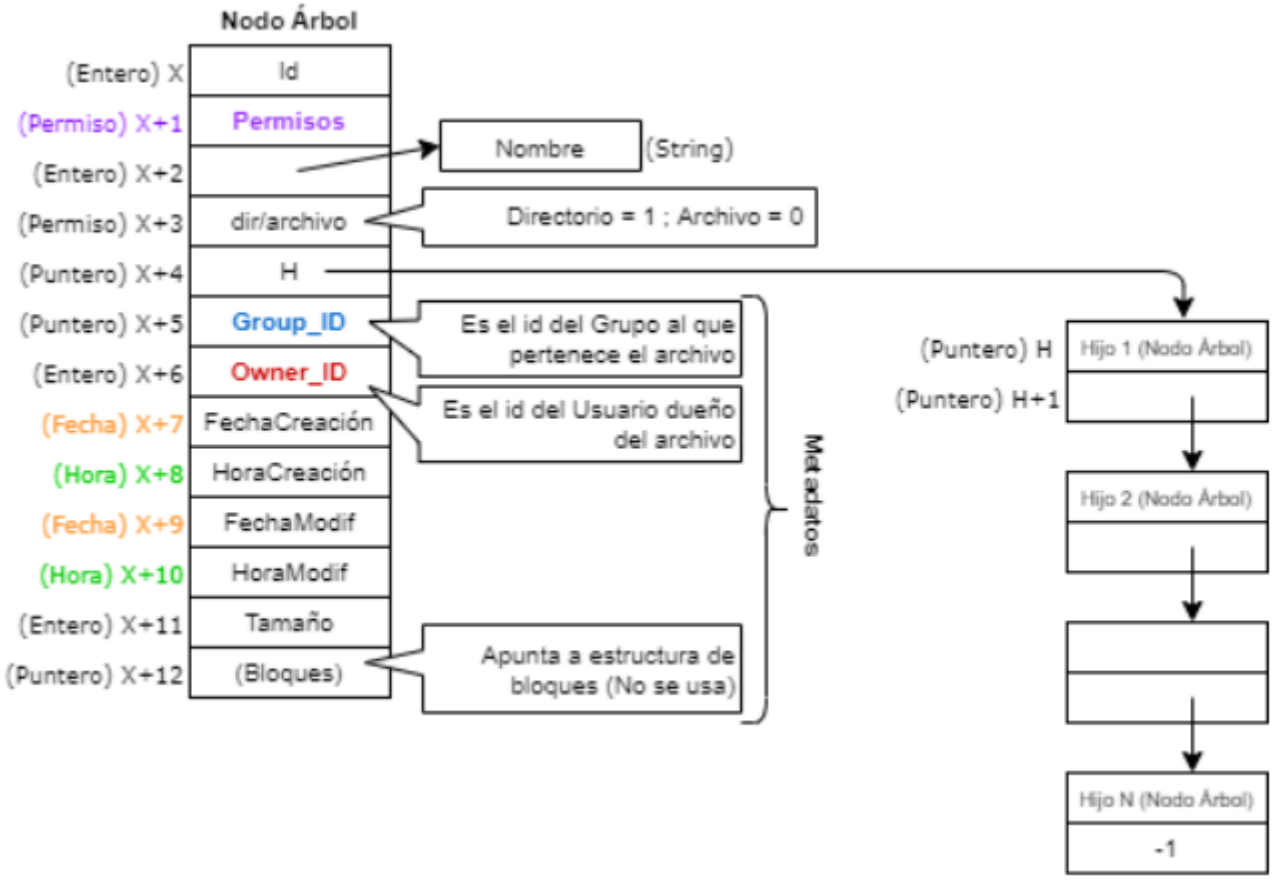
- Si en permisos hay un @777 (todos los bits en 1) quiere decir que tanto el owner, como el miembro del grupo, como cualquier usuario tiene permiso de lectura escritura y ejecución.
- Si en permisos hay un @751 (binario 111 101 001) quiere decir que el owner puede leer, escribir y ejecutar; el miembro del grupo puede leer y ejecutar pero no escribir; y por último el resto de los usuarios pueden solo ejecutar.

Para validar si un usuario tiene permisos se debe seguir el siguiente criterio:

1) Verificar si es el owner del archivo, y luego mirar solo los permisos del owner (@700)
2) Verificar si es miembro del grupo, y luego mirar solo los permisos del grupo (@070)
3) Si no entró ninguno de los casos anterior, mirar los permisos de otros (@007)

A Partir de acá los parciales tienen diferentes combinaciones de árboles y estructura de usuarios y grupos.

Árbol A1 // Parcial 1,3,4,7,10

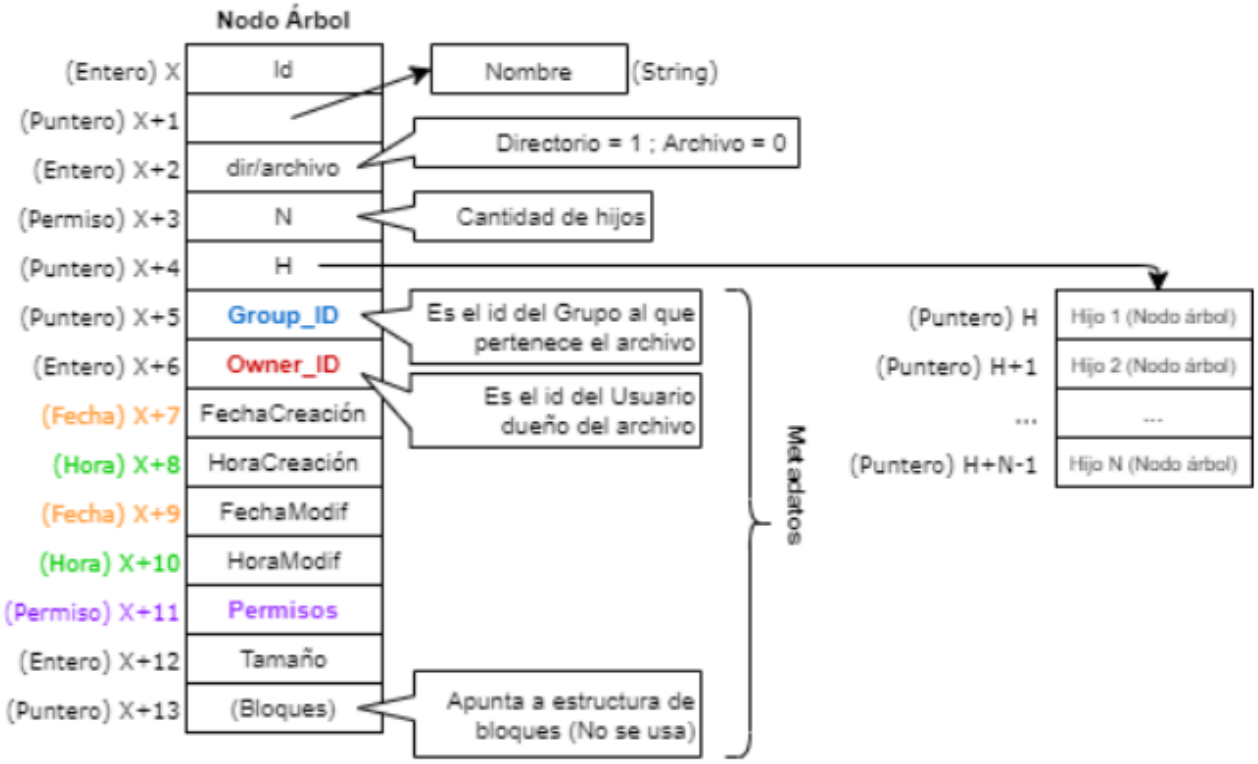


Los archivos y directorios (carpetas) se representan con un nodo de un árbol en una dirección X de memoria. Se diferencia archivos de directorios por el campo en X+2 que si tiene un **1** el nodo corresponde a un **directorio** y si tiene un **0** corresponde a un **archivo**.

Cada nodo del árbol también contiene los metadatos de los archivos/directorios. Ahí mismo se encuentra el **id del usuario** (entero positivo) que es dueño del archivo (OWNER_ID) en X+6 y el **id del grupo** (entero positivo) al que pertenece el archivo (GROUP_ID) en X+5.
Los campos tamaño (X+11) y bloques (X+12) **solo son válidos para los archivos**.

Si el nodo **es un directorio**, en X+4 **apunta a una lista de hijos**. Cada nodo de la lista tiene un puntero a nodo de árbol y un puntero al siguiente. **Si** el directorio **está vacío** en X+4 habrá un -1 (null).

Árbol A2// Parcial 2,5,11

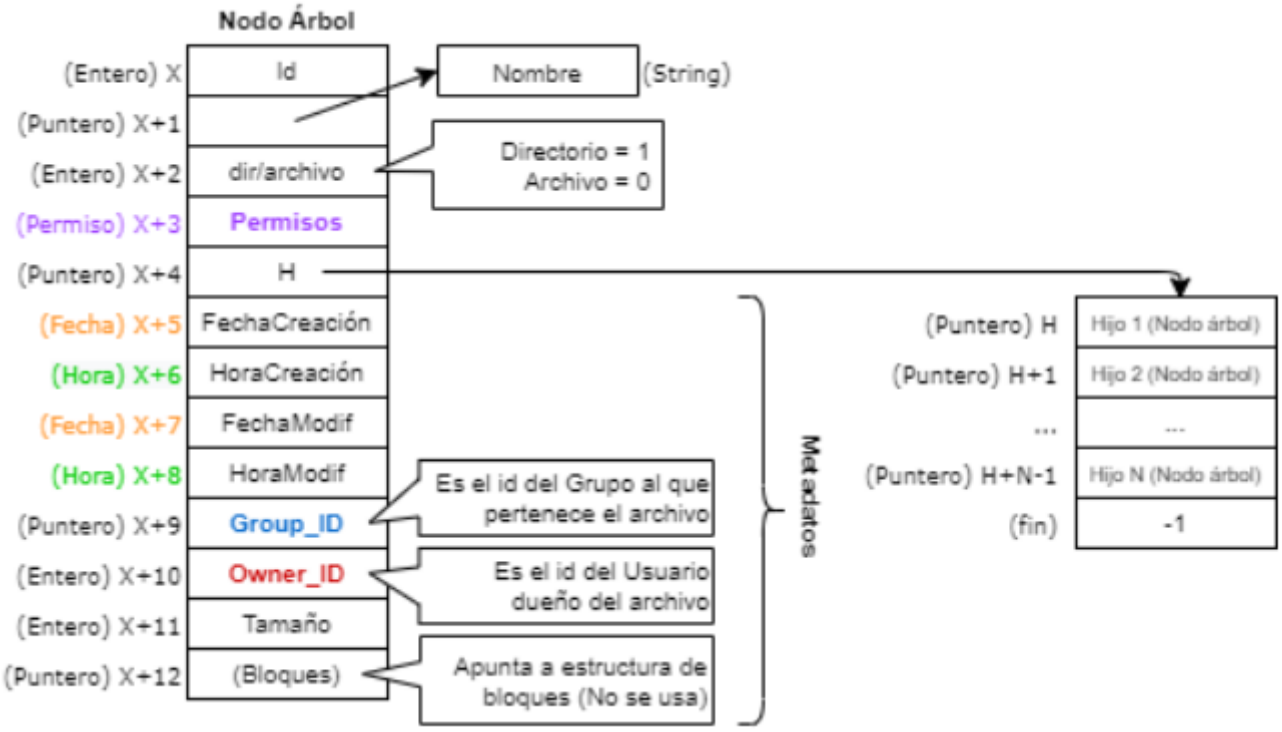


Los archivos y directorios (carpetas) se representan con un nodo de un árbol en una dirección X de memoria. Se diferencia archivos de directorios por el campo en X+2 que si tiene un **1** el nodo corresponde a un **directorio** y si tiene un **0** corresponde a un **archivo**.

Cada nodo del árbol también contiene los metadatos de los archivos/directorios. Ahí mismo se encuentra el **id del usuario** (entero positivo) que es dueño del archivo (OWNER_ID) en X+6 y el **id del grupo** (entero positivo) al que pertenece el archivo (GROUP_ID) en X+5.
Los campos tamaño (X+12) y bloques (X+13) **solo son válidos para los archivos**.

Si el nodo **es un directorio**, en X+4 **apunta a una lista de hijos** que comienza en la celda H. La lista es un arreglo estático de N elementos donde en cada celda tiene un puntero a un nodo del árbol. El valor N con la cantidad de hijos se encuentra en X+4.
Si el directorio **está vacío**, X+3 tendrá valor 0.

Árbol A3 // Parcial 6,8,9



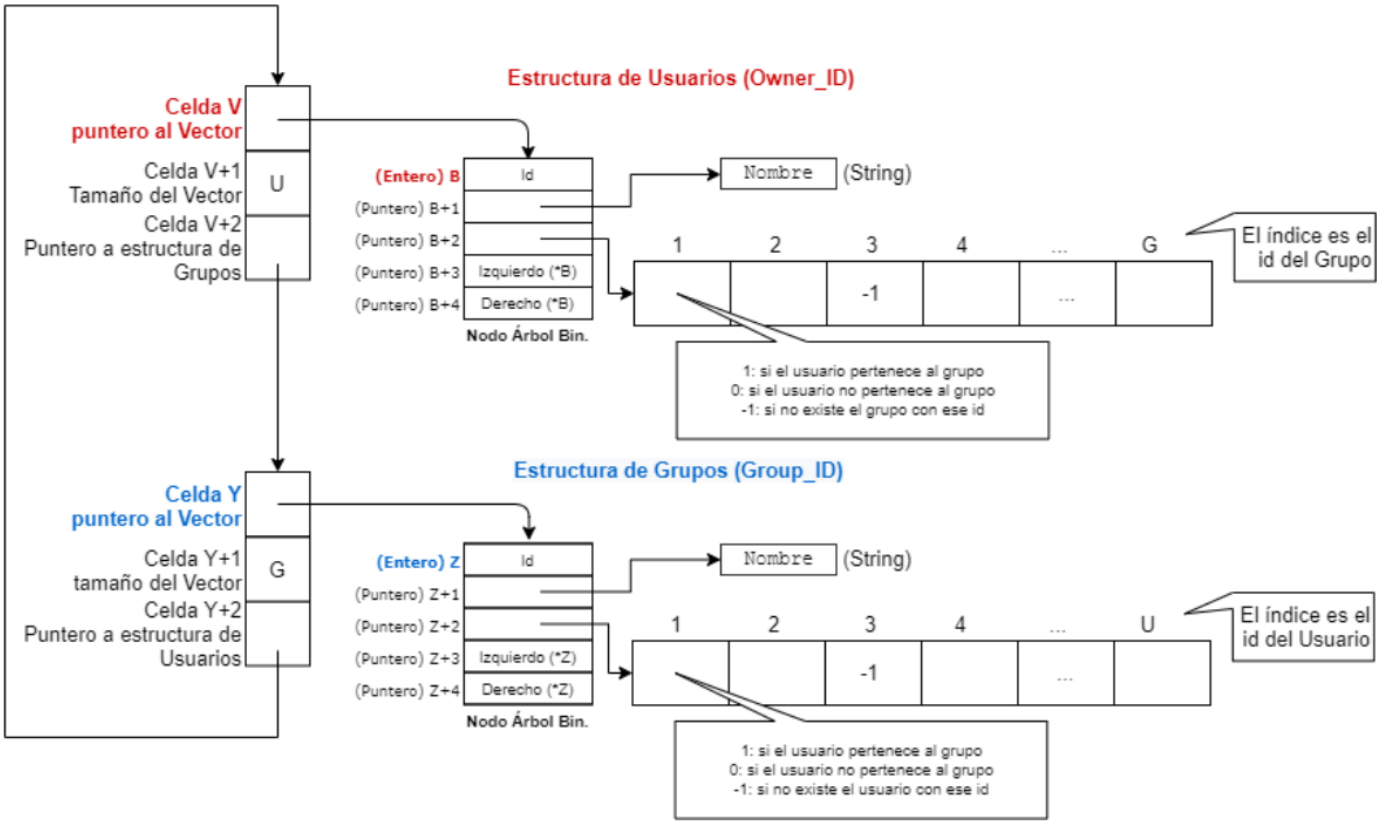
Los archivos y directorios (carpetas) se representan con un nodo de un árbol en una dirección X de memoria. Se diferencia archivos de directorios por el campo en X+2 que si tiene un **1** el nodo corresponde a un **directorio** y si tiene un **0** corresponde a un **archivo**.

Cada nodo del árbol también contiene los metadatos de los archivos/directorios. Ahí mismo se encuentra el **id del usuario** (entero positivo) que es dueño del archivo (OWNER_ID) en X+10 y el **id del grupo** (entero positivo) al que pertenece el archivo (GROUP_ID) en X+9.
Los campos tamaño (X+11) y bloques (X+12) **solo son válidos para los archivos**.

Si el nodo **es un directorio**, en X+4 **apunta a una lista de hijos** que comienza en la celda H. La lista es un arreglo donde en cada celda tiene un puntero a un nodo del árbol, el fin de la lista se indica con -1 como valor.
Si el directorio **está vacío**, X+4 tendrá una dirección H y en esa celda H estará el valor -1.

Usuarios y grupos:

UG1 // Parcial 1,8,9,10



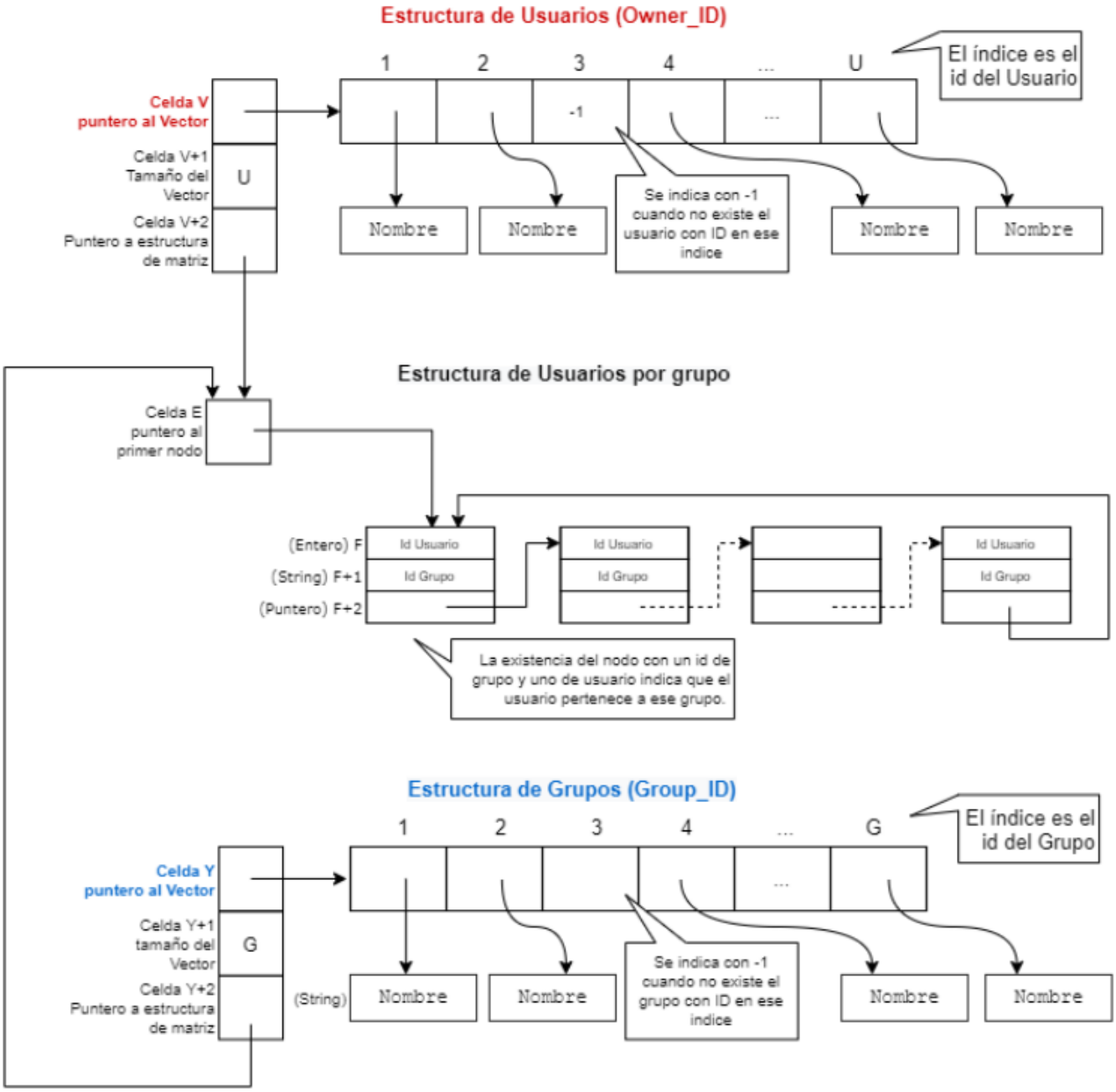
La estructura de **usuarios** se compone de 3 celdas con:

- V apunta al nodo raíz de un árbol binario de búsqueda ordenado por el id del usuario. Donde cada nodo se compone de 5 celdas:
  - B contiene el id del usuario.
  - B+1 contiene un puntero a un string con el nombre del usuario.
  - B+2 apunta a un vector donde el índice de la celda corresponde al id de un grupo y el contenido puede ser 1 si el usuario pertenece al grupo, 0 si el usuario no pertenece o -1 si el grupo no existe.
- V+2 contiene la cantidad de usuarios.
- V+3 contiene un puntero a la estructura de grupos Y

La estructura de **grupos**, es idéntica a la de usuarios: El árbol binario está ordenado por el id del grupo, y el índice del vector de usuarios es el id de usuario conteniendo 1, 0 o -1 en los casos que el usuario pertenezca al grupo, no pertenezca o no exista.

Tanto en **usuarios** como en **grupos** si la celda que apunta al nombre tiene -1 (NULL), el usuario o grupo está eliminado (es decir que ese id está disponible para ser utilizado).
Entonces, para eliminar un usuario o un grupo, no se debe eliminar el nodo, solo marcarlo con NULL sobre el nombre.

UG2 // Parcial 2,3,4



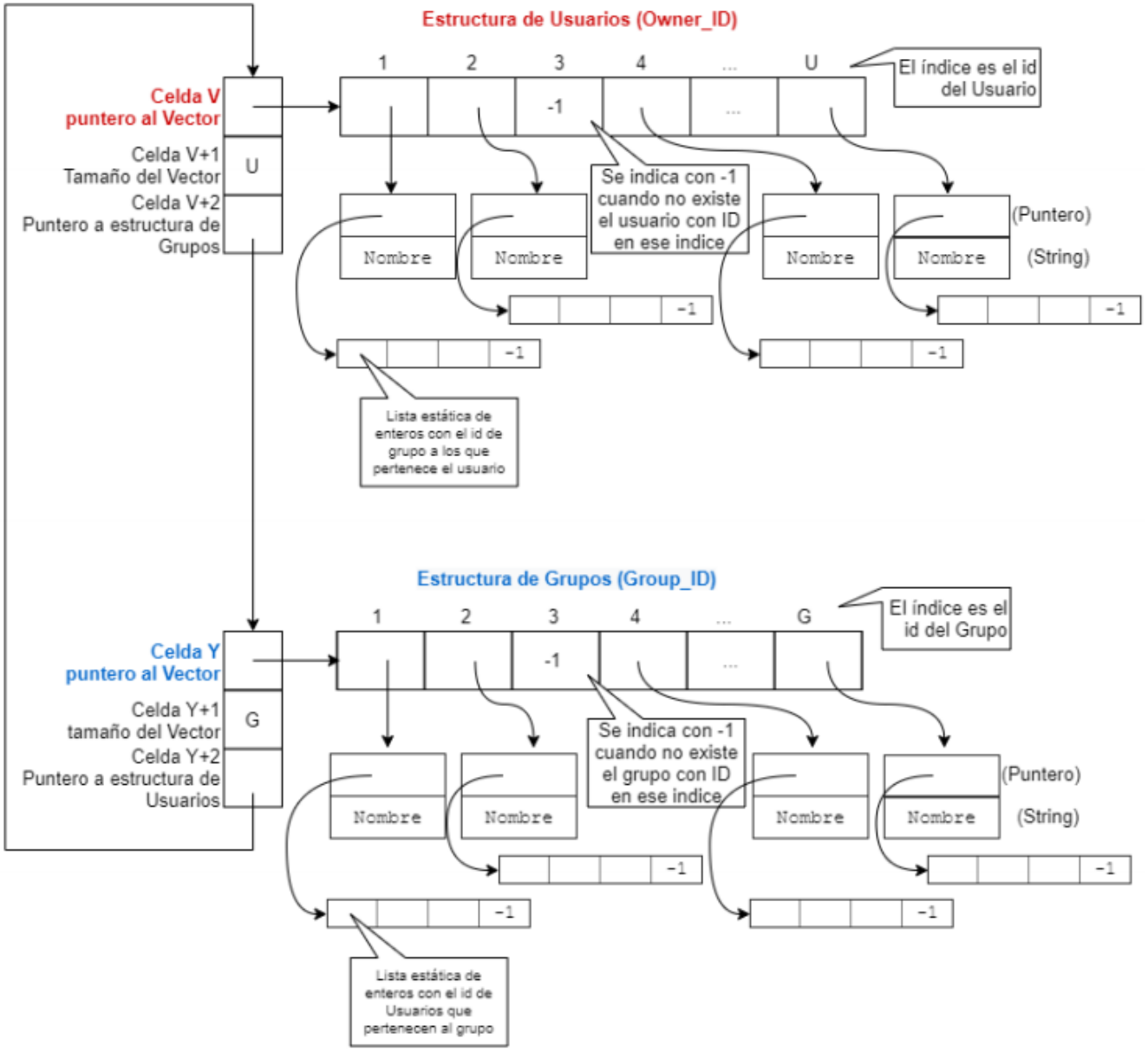
La estructura de **usuarios** tiene **3 celdas** comenzando en la **celda V**:

- V apunta a una lista estática (vector), donde cada celda tiene el puntero al string con el nombre de usuario y la posición relativa al comienzo de la celda corresponde al id de usuario.
- V+1 Contiene la cantidad de celdas del vector de usuario
- V+2 apunta a la estructura de usuarios x grupo

La estructura de **grupos**, es idéntica a la de usuarios, **3 celdas** comenzando en la **celda Y** solo que contiene los nombres de los grupos.

La estructura que **vincula usuarios y grupos** consta de **una celda E** que apunta un nodo que tiene id usuario, id grupo y siguiente nodo. Es una lista circular, de modo que no hay un último nodo.
Si existe el nodo, quiere decir que, el **usuario pertenece al grupo**.

UG3 // Parcial 5,6,7,11



La estructura de **usuarios** tiene **3 celdas** comenzando en la **celda V**:

- V apunta a una lista estática (vector) donde:
  - La posición relativa al comienzo de la celda corresponde al id de usuario, comenzando en 1.
  - Cada celda apunta a una estructura con:
    - Puntero a un vector que tiene la listas de ids de los grupos a los que pertenece el usuario
    - El string con el nombre del usuario (finaliza con \0)
- V+1 Contiene la cantidad de celdas del vector de usuario
- V+2 apunta a la estructura de grupos Y

La estructura de **Grupos** idéntica a la de usuarios, solo que el vector el índice indica el id del grupo y los vectores a los que apuntan cada celda contienen los ids de los usuarios que pertenecen al grupo.

Ejercicios

Ejercicio 1 Arbol A1 y Usuarios y grupos UG1

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Devolver en una lista los id y nombre de todos los usuarios con permiso de escritura de un archivo determinado. La lista debe ser doblemente enlazada, estar ordenada por id de usuario y **no** tener usuarios repetidos. Se debe especificar su estructura.

La subrutina principal tiene como parámetros:

- un puntero a un archivo (X) del sistema de archivos,
- la dirección de memoria (V) con a la estructura de usuarios del sistema,
- la dirección de memoria donde se encuentra el puntero al primer nodo de la lista.

Y debe responder al siguiente protocolo en C:

```
void list (TArbol* archivo, TUsuarios* usrs, Tlist* lista)
```

Nota: debe contemplar solicitar memoria si es necesario.

Ejercicio 2 Arbol A2 y Usuarios y grupos UG2

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Eliminar un usuario del filesystem, quitándolo de la estructura de usuarios y de todos los grupos a los que pertenece. Si el usuario no existe no se puede eliminar. Si el usuario es owner de algún archivo entonces no puede ser eliminado.

Devuelve:

- en AX = id del usuario si pudo eliminarlo,
- o AX = 0 si no es posible eliminar el usuario.

La subrutina principal recibe como parámetros:

- un puntero a un directorio (X) raíz del sistema de archivos,
- un puntero a un string con el nombre de un usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema.

Y debe responder al siguiente protocolo en C:

```
int delusr (TArbol* raiz, String usrname, TUsuarios* usrs)
```

Nota: debe contemplar liberar la memoria de nodos no utilizados. Tomar en cuenta que todas las direcciones de memoria fueron obtenidas dinámicamente, utilizando llamadas al sistema.

Ejercicio 3 Arbol A1 y Usuarios y grupos UG2

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Devolver en una lista los id y nombre de todos los usuarios con permiso de escritura de un archivo determinado. La lista debe ser doblemente enlazada, estar ordenada por id de usuario y **no** tener usuarios repetidos. Se debe especificar su estructura.

La subrutina principal tiene como parámetros:

- un puntero a un archivo (X) del sistema de archivos,
- la dirección de memoria (V) con a la estructura de usuarios del sistema,
- la dirección de memoria donde se encuentra el puntero al primer nodo de la lista.

Y debe responder al siguiente protocolo en C:

```
void list (TArbol* archivo, TUsuarios* usrs, Tlist* lista)
```

Nota: debe contemplar solicitar memoria si es necesario.

Ejercicio 4 Arbol A1 y Usuarios y grupos UG2

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Eliminar un grupo del filesystem, quitando de la estructura de grupos y poner -1 en el Group_ID de todos los archivos del filesystem que pertenecían al grupo y quitar los permisos de grupo de los mismos. Solo se puede eliminar el grupo si existe, y ningún usuario pertenece al mismo.

Devuelve:

- en AX = id del grupo si pudo ser eliminarlo,
- o AX = 0 si no es posible eliminar el grupo.

La subrutina principal recibe como parámetros:

- un puntero a un directorio (X) raíz del sistema de archivos,
- un puntero a un string con el nombre de un grupo,
- la dirección de memoria (Y) con a la estructura de grupos del sistema.

Y debe responder al siguiente protocolo en C:

```
int delgrp (TArbol* raiz, String grpname, TGrupos* grps)
```

Nota: debe contemplar liberar la memoria de nodos no utilizados.

Ejercicio 5 Arbol A2 y Usuarios y grupos UG3

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Devolver en una lista todos los id y nombre de los archivos, que han sido modificados luego de una fecha y hora dada, y que poseen permiso de ejecución de un usuario determinado, La subrutina debe buscar recursivamente en todo el sistema de archivos. Se debe especificar la estructura de la lista.

La subrutina principal tiene como parámetros:

- un puntero a un directorio (X) raíz del sistema de archivos,
- la fecha,
- la hora,
- el id de usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema,
- la dirección de memoria donde se encuentra el puntero al primer nodo de la lista.

Y debe responder al siguiente protocolo en C:

```
void list (TArbol* raiz, TFecha fecha, THora hora, int idUsuario, TUsuarios* usrs, Tlist* lista)
```

Nota: debe contemplar solicitar memoria si es necesario.

Ejercicio 6 Arbol A3 y Usuarios y grupos UG3

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Devolver en una lista todos los id y nombre de los archivos, que han sido modificados luego de una fecha y hora dada, y que poseen permiso de ejecución de un usuario determinado, La subrutina debe buscar recursivamente en todo el sistema de archivos. Se debe especificar la estructura de la lista.

La subrutina principal tiene como parámetros:

- un puntero a un directorio (X) raíz del sistema de archivos,
- la fecha,
- la hora,
- el id de usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema,
- la dirección de memoria donde se encuentra el puntero al primer nodo de la lista.

Y debe responder al siguiente protocolo en C:

```
void list (TArbol* raiz, TFecha fecha, THora hora, int idUsuario, TUsuarios* usrs, Tlist* lista)
```

Nota: debe contemplar solicitar memoria si es necesario.

Ejercicio 7 Arbol A1 y Usuarios y grupos UG3

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Agregar un grupo nuevo al filesystem y asignar al mismo todos los usuarios con permiso de lectura de un archivo dado. La subrutina debe intentar re-utilizar un id de grupo que no esté siendo utilizado o en su defecto agregar un id nuevo.

Devuelve:

- en AX = id del grupo agregado

La subrutina principal recibe como parámetros:

- un puntero a un archivo (X) del sistema de archivos,
- un puntero a un string con el nombre de un grupo,
- la dirección de memoria (Y) con a la estructura de grupos del sistema.

Y debe responder al siguiente protocolo en C:

```
int addgrp (TArbol* raiz, String grpname, TGrupos* grps)
```

Notas:

1. Para el caso de estructuras dinámicas: contemple solicitar memoria.
2. Para el caso de estructuras estáticas: asuma que hay espacio disponible contiguo en la memoria.
3. Asumir que no existe un grupo con el mismo nombre del grupo nuevo.

Ejercicio 8 Arbol A3 y Usuarios y grupos UG1

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Agregar un grupo nuevo al filesystem y asignar al mismo todos los usuarios con permiso de lectura de un archivo dado. La subrutina debe intentar re-utilizar un id de grupo que no esté siendo utilizado o en su defecto agregar un id nuevo.

Devuelve:

- en AX = id del grupo agregado

La subrutina principal recibe como parámetros:

- un puntero a un archivo (X) del sistema de archivos,
- un puntero a un string con el nombre de un grupo,
- la dirección de memoria (Y) con a la estructura de grupos del sistema.

Y debe responder al siguiente protocolo en C:

```
int addgrp (TArbol* raiz, String grpname, TGrupos* grps)
```

Notas:

1. Para el caso de estructuras dinámicas: contemple solicitar memoria.
2. Para el caso de estructuras estáticas: asuma que hay espacio disponible contiguo en la memoria.
3. Asumir que no existe un grupo con el mismo nombre del grupo nuevo.

Ejercicio 9 Arbol A3 y Usuarios y grupos UG1

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Agregar un usuario nuevo al filesystem y asignarlo a todos los grupos de los archivos (solo archivos) que se encuentran en un directorio dado. La subrutina debe intentar re-utilizar un id de usuario que no esté siendo utilizado o en su defecto agregar un id nuevo.

Devuelve:

- en AX = id del usuario agregado.

La subrutina principal recibe como parámetros:

- un puntero a un directorio (X) del sistema de archivos,
- un puntero a un string con el nombre de un usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema.

Y debe responder al siguiente protocolo en C:

```
int addusr (TArbol* dir, String usrname, TUsuarios* usrs)
```

Notas:

1. Para el caso de estructuras dinámicas: contemple solicitar memoria.
2. Para el caso de estructuras estáticas: asuma que hay espacio disponible contiguo en la memoria.
3. Asumir que no existe un usuario con el mismo nombre del usuario nuevo.

Ejercicio 10 Arbol A1 y Usuarios y grupos UG1

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Agregar un usuario nuevo al filesystem y asignarlo a todos los grupos de los archivos (solo archivos) que se encuentran en un directorio dado. La subrutina debe intentar re-utilizar un id de usuario que no esté siendo utilizado o en su defecto agregar un id nuevo.

Devuelve:

- en AX = id del usuario agregado.

La subrutina principal recibe como parámetros:

- un puntero a un directorio (X) del sistema de archivos,
- un puntero a un string con el nombre de un usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema.

Y debe responder al siguiente protocolo en C:

```
int addusr (TArbol* dir, String usrname, TUsuarios* usrs)
```

Notas:

1. Para el caso de estructuras dinámicas: contemple solicitar memoria.
2. Para el caso de estructuras estáticas: asuma que hay espacio disponible contiguo en la memoria.
3. Asumir que no existe un usuario con el mismo nombre del usuario nuevo.

Ejercicio 11 Arbol A2 y Usuarios y grupos UG3

Escriba la (o las) subrutina(s) necesaria(s) para resolver el siguiente problema:

Agregar un usuario nuevo al filesystem y asignarlo a todos los grupos de los archivos (solo archivos) que se encuentran en un directorio dado. La subrutina debe intentar re-utilizar un id de usuario que no esté siendo utilizado o en su defecto agregar un id nuevo.

Devuelve:

- en AX = id del usuario agregado.

La subrutina principal recibe como parámetros:

- un puntero a un directorio (X) del sistema de archivos,
- un puntero a un string con el nombre de un usuario,
- la dirección de memoria (V) con a la estructura de usuarios del sistema.

Y debe responder al siguiente protocolo en C:

```
int addusr (TArbol* dir, String usrname, TUsuarios* usrs)
```

Notas:

1. Para el caso de estructuras dinámicas: contemple solicitar memoria.
2. Para el caso de estructuras estáticas: asuma que hay espacio disponible contiguo en la memoria.
3. Asumir que no existe un usuario con el mismo nombre del usuario nuevo.

Solución

(Aquí escribe la solución el alumno)

```
;Especificación de las todas las subrutinas a utilizar
            push        <>    ;comentario del argumento
            ...
            call
            ...         ;especificar salida (si corresponde)

;subrutina...
NOMBRE_SUBRUTINA:       push  bp
                        ...
```