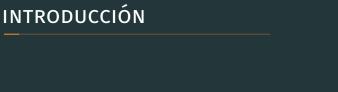
## AYUDANTÍA P1

Raúl Andrés Álvarez Esteban Ricardo Esteban Schilling Broussaingaray IIC2333 [2020-1] - Sistemas Operativos y Redes



## INTRODUCCIÓN

Los objetivos de esta ayudantía son:

- 1. Comprender la estructura de nuestro sistema de archivos.
- 2. Aclarar el funcionamiento esperado de la API crfs.

2



#### **BLOQUES**

- · El disco es de tamaño 2GB con 4 particiones, cada una con bloques de 8KB cada uno.
- · Este tiene 262144 bloques, ordenados de manera secuencial.
- Un puntero a un bloque no es más que un unsigned int de 4 bytes, correspondiente al número de bloque.
- Cuando un bloque es asignado a una función específica, este se asigna completamente.
- El primer bloque de cada partición siempre será nuestro directorio root.

#### **BLOQUE DE DIRECTORIO**

El bloque de directorio corresponde a un directorio en nuestro sistema de archivos. Cada entrada del directorio, es decir, cada archivo o subdirectorio dentro de este está representado como una secuencia de 32 bytes.

- · Un bit para indicar si la entrada es inválida (0x0) o válida (0x01).
- · 23 bits que representan el puntero al bloque índice del archivo.
- 29 bytes que representan el nombre de la entrada en ASCII considerando su extensión, cuando un byte no esté siendo ocupado, este debe ser seteado a 0x00. Los bytes de texto se alinean a la izquierda.

#### **BLOQUE DE BITMAP**

- · Corresponden al bloque posterior al bloque raíz de cada partición.
- Su función es indicar los bloques que están ocupados y los que están libres.
- · Habrá un bit igual a 1 si el bloque correspondiente está ocupado y 0 en caso contrario.

## Ejemplo

Supongamos que el byte número 123 (contando desde 0) del primer bloque de bitmap es 0xA3 (10100011<sub>2</sub>), por lo que inmediatamente sabemos que los bloques 984, 986, 990 y 991 están ocupados.

- Los primeros 2 bloques de cada partición siempre estarán ocupados.
- Los bloques de bitmap siempre deben reflejar el estado actual del disco.

### **BLOQUE ÍNDICE**

- · Es el primer bloque de un archivo.
- Comienza con 4 bytes que representan la cantidad de referencias al archivo.
- · Posee 8 bytes para el tamaño del archivo.
- · Posee 8176 bytes para 2044 punteros que apuntan directamente a bloques de datos.
- Además, se reservan los últimos 4 bytes para un puntero de direccionamiento indirecto.
- El orden de los bloques en el bloque índice dicta el orden de los bloques de datos del archivo.

7

#### **BLOQUES DE DIRECCIONAMIENTO INDIRECTO**

- Similar al bloque indice, solo que no poseen bytes para metadata o puntero final, por lo que este bloque tiene espacio para guardar 2048 punteros.
- El bloque de direccionamiento indirecto simple posee punteros directos a bloques de datos.
- Esto, naturalmente, establece un tamaño máximo de archivo:  $(2045*8KB + 2048*8KB) \approx 32MB$

#### **BLOQUE DE DATOS**

- · Utiliza la totalidad de su espacio para guardar los archivos.
- No pueden ser subasignados, es decir, cuando uno asigna un bloque de datos a un archivo, este se asigna en su totalidad y no pueden haber dos o más archivos compartiendo el mismo bloque.

# CRFS API

- · Cristian Ruz File System.
- La API debe estar implementada en un archivo cr\_API.c con la interfaz llamada cr\_API.h.
- Debe funcionar a partir de un programa main.c que utilice las funciones de su librería.
- · Son libres de subir sus propios archivos para agregarlos al disco.
- Dentro de su API, deben definir un struct llamado crFILE, el que representa un archivo abierto. Este es similar al struct FILE de la librería stdio.h. Son libres en cuanto a los datos que posee esta estructura.
- · La API utiliza únicamente rutas absolutas.

#### **FUNCIONES GENERALES**

- void cr\_mount(char\* diskname) Esta función se encarga de montar el disco, dejando como variable global la ruta al archivo binario correspondiente. Siempre es la primera función que corre en su archivo main.
- void cr\_bitmap(unsigned disk, bool hex) Imprime el bloque de bitmap del disco representado por disk, si disk es igual a 0, se imprimen todos los bitmaps junto a la cantidad de bloques ocupados y la cantidad de bloques libres.

#### **FUNCIONES GENERALES**

- int cr\_exists(unsigned disk, char\* path) Retorna 1
  si path existe en la partición entregada y 0 si no.
- · void cr\_ls(unsigned disk) Similar al comando ls de Unix, imprime los contenidos de la partición entregada.

#### **FUNCIONES DE MANEJO DE ARCHIVOS**

- crFILE\* cr\_open(unsigned disk, char\* path, char mode) Abre un archivo y retorna un puntero a la instancia de crFILE que lo representa. El modo puede ser 'r' para leer archivos existentes o 'w' para escribir nuevos archivos.
- int cr\_read(crFILE\* file\_desc, void\* buffer, int nbytes) Lee los siguientes nbytes del archivo descrito por file\_desc y lo guarda en un buffer. La función debe retornar la cantidad de bytes leídos.
- int cr\_write(crFILE\* file\_desc, void\* buffer, int nbytes) Escribe los nbytes que se encuentren en el buffer al archivo descrito por file\_desc.

#### **FUNCIONES DE MANEJO DE ARCHIVOS**

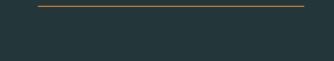
- int cr\_close(crFILE\* file\_desc) Función que cierra un archivo abierto previamente. Cuando un archivo es cerrado, este debe estar actualizado en el disco.
- int cr\_rm(unsigned disk, char\* path) Función que elimina una referencia a un archivo. Es muy importante que se liberen todos los bloques ocupados por el archivo solo si es que no existen mas referencias a este.

#### **FUNCIONES DE MANEJO DE ARCHIVOS**

- int cr\_load(unsigned disk, char\* orig) Toma una carpeta o archivo del computador y carga todos los archivos que contenga a la partición indicada. Ignora cualquier subcarpeta.
- int cr\_unload(unsigned disk, char\* orig, char \*dest) Contrario a la función load, esta función recibe la ruta de un archivo de una partición y lo guarda en la ruta dest del computador.

#### **FUNCIONES DE MANEJO DE LINKS**

- int cr\_hardlink(unsigned disk, char\* orig, char\* dest): función que se encarga de crear un hardlink del archivo referenciado por orig una nueva ruta dest, aumentando la cantidad de referencias al archivo original.
- int cr\_softlink(unsigned disk\_orig, unsigned disk\_dest, char\* orig, char\* dest): Función que se encarga de crear un softlink del archivo referenciado por orig una nueva ruta dest, pudiendo este último estar en una partición distinta al archivo original.



FIN