Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2333 - Sistemas Operativos y Redes

# Proyecto 1

7 de octubre de 2019

2º semestre 2019 - Profesor C. Ruz

Alfonso Aguirrebeña - Henry Blair - Pablo Haeussler - Cristobal Poblete

---

## 1. cr_API.c

Módulo que contiene tanto las funciones generales como funciones para el manejo de archivos dentro del disco.

### Funciones Generales

- **void cr_mount(char* diskname)**.
  """
  char* diskname: recibe como argumento un string con el path local del disco que se desea montar
  """
  Función para montar el disco. Establece como variable global la ruta local donde se encuentra el archivo.bin correspondiente al disco

- **void cr_bitmap(unsigned block, bool hex)**.
  """
  unsigned* block: bitmap que se desea mostrar
  bool hex: booleano que determina si mostrar en hexadecimal o binario el bitmap
  """
  Función para imprimir el bitmap. Al llamar a esta funcion se lee el disco montado con la función anterior y se salta el primer bloque el cual corresponde al root. Esta imprime en stderr el estado actual del bloque de bitmap correspondiente a block (bitmap block ∈1, ...,129), ya sea en binario (hex is false) o en hexadecimal (hex is true). Si se ingresa block=0, se imprime el bitmap completo, además una linea ocupied: indicando la cantidad de bloques ocupados y una linea free: indicando la cantidad de bloques libres. Si se ingresa un bitmap fuera del rango, imprime mensaje que muestra el error.

- **int cr_exists(char* path)**.
  ””””
  char* path: recibe como argumento un string con el path de un archivo o carpeta
  ””””
  Funcion que permite identificar si un archivo o carpeta existe en la ruta especificada en path. Retorna 1 si el archivo o carpeta existe y 0 en caso contrario. Realiza esto de manera recursiva con ayuda de la función auxiliar cr_exists_recur mencionada en la seccion de Funciones Auxiliares.

- **void cr_ls(char* path)**
  ””””
  char* path: recibe como argumento el path de un archivo o carpeta
  ””””
  Funcion para listar los elementos de un directorio del disco. Imprime en pantalla los nombres de todos los archivos y directorios contenidos en el directorio indicado por path.

- **int cr_mkdir(char* foldername)**
  ””””
  char* path: recibe como argumento un path en donde se quiere crear un nuevo directorio
  ””””
  Utiliza dos funciones auxiliares; cr_exists_direction y get_pos_on_directory explicadas en la sección de funciones auxiliares. Creará un nuevo directorio en el path entregado en foldername, tomando en cuenta si se debe crear o no un bloque de directorio de continuación en caso de ser necesario.

## Funciones de manejo de archivos

- **crFILE* cr_open(char* path, char mode)**
  """
  char* path: string que corresponde al path del archivo que se desea abrir char mode: modo para abrir archivo, puede ser r.º "w". """
  Función para abrir un archivo. Si mode es 'r', busca el archivo en la ruta path y retorna un crFILE* que lo representa. Si mode es 'w', se verifica que el archivo no exista en la ruta especificada y se retorna un nuevo crFILE* que lo representa. crFile* corresponde a una estructura crfile de archivo que contiene;
  - crfile → mode: mode
  - crfile → overright: bool que indica si el archivo se sobreescribirá o no
  - crfile → block: almacena la direccion de memoria del bloque
  - crfile → pos_data: 1
  - crfile → pos_index: 1
  - crfile → path: guarda el path del archivo abierto
- **int cr_read(crFILE* file_desc, void* buffer, int nbytes)**
  """
  crFILE* file_desc:
  void* buffer:
  int nbytes:
  """
- **int cr_write(crFILE* file_desc, void* buffer, int nbytes)**
  """
  crFILE* file_desc:
  void* buffer:
  int nbytes:
  """
- **int cr_close(crFILE* file_desc)**
  """
  crFILE* file: struct crfile de file """
  Función para cerrar archivos. Cierra el archivo indicado por filedesc. Debe garantizar que cuando esta función retorna, el archivo se encuentra actualizado en disco. Libera la struct crfile.
- **int cr_rm(char* path)**
  """

char* path: recibe path de algun elemento """"
Función para borrar archivos. Elimina el archivo referenciado por la ruta path del directorio correspondiente. Los bloques que estaban siendo usados por el archivo deben quedar libres, esto se genera modificando el bitmap correspondiente al bloque del directorio eliminado.

- **int cr_unload(char* orig, char* dest)**
  """"

  """"

- **int cr_load(char* orig)**
  """"

  """"

## Funciones auxiliares

- **uint32_t get_pointer(FILE* pos)**
  """
  FILE* pos: puntero hacia cierta dirección del archivo
  """
  Retorna el puntero de pos.

- **void write_pointer(FILE* pos, uint32_t pointer)**
  """
  FILE* pos: Direccion de memoria de disk
  uint32_t pointer: datos memoria direccion que se desea guardar
  """
  Escribirá un puntero pointer en los siguientes 4 Byte de pos.

- **void write_name(FILE* disk, char* name)**
  """
  FILE* disk: espacio de memoria en donde se guardara un nombre
  char* name: nombre de archivo que se desea guardar
  """
  Escribirá en nombre name de un elemento en los siguientes 27 Byte de disk

- **int cr_exists_recur(FILE* disk, char** path, int len, uint32_t my_dir, int from)**
  """
  FILE* disk: argumento tipo file del disco
  char** path: array que contiene los elementos del path ej: /home/usuario/carpeta1 → [home, usuario, carpeta1]
  int len: int que representa la cantidad de elementos del array anterior
  uint32_t my_dir: direccion de memoria del elemento del array que se esta revisando
  int from: int que indica que elemento del array se esta revisando
  """
  Recorre recursivamente el path entregado en forma de array, identificando en que elemento va gracias a from y retornara 1 si el elemento actual se encuentra en el directorio anterior, retornara 0 en caso contrario.

- **uint32_t cr_exists_direction_recur(FILE* disk, char** path, int len, uint32_t my_dir, int from)**:
  """
  FILE* disk: argumento tipo file con el disco
  char** path: lista de array que contiene los elementos del path ej: /home/usuario/carpeta1 → [home, usuario, carpeta1]
  int len: int que representa la cantidad de elementos del array anterior
  uint32_t my_dir: direccion de memoria del elemento del array que se esta revisando
  int from: int que indica que elemento del array se esta revisando

"""
Busca de manera recursiva la direccion de memoria del elemento indicado en path, para finalmente retornar esta dirección.

- **uint32_t cr_exists_direction(char* path)**
"""
char* path: string correspondiente a un path
"""
Retorna la dirección de memoria del elemento indicado en el path, además se asume que el path existe en el disco ya que se utiliza cr_exist antes de utilizar esta función.

- **uint32_t reserve_unused_block()**
Funcion que retornar la dirección del primer bloque que se encuentre sin ocupar para crear un nuevo directorio, además de modificar el valor del bitmap correspondiente de dicho bloque de 0 (bloque libre) a 1 (bloque ocupado).

- **uint32_t create_new_dir_cont(FILE* disk, char* name)**
"""
FILE* disk: archivo del disco tipo FILE
char* name: nombre del elemento que se desea crear
"""
Esta funcion se utiliza en caso de que al crear un nuevo directorio en un directorio que este lleno, se busca un nuevo bloque que este vacío con reserve_unused_block para que sea un bloque de continuacion y en este bloque se guarda el nuevo elemento creado. Utiliza la función write_name y write_pointer para guardar los datos respectivos. Retorna finalmente la direccion de memoria del nuevo bloque de continuacion.

- **int get_pos_on_directory(FILE* disk, uint32_t dir, char* dirname, char* my_name)**
"""
File* disk: archivo del disco tipo FILE
uint32_t dir: direccion de memoria de directorio de donde se creara un nuevo elemento
char* dirname: nombre del directorio en donde se creara nuevo elemento
char* my_name: nombre del nuevo directorio a crear
"""
Función que escribe el nuevo directorio, para esto utiliza la función auxiliar reserve_unused_block explicada anteriormente, para reservar el bloque en donde estará guardado el nuevo directorio y lo crea con la función create_new_dir_cont. Además guarda el nombre del directorio nuevo en la primera posicion disponible del directorio indicado por el path de cr_mkdir. Retorna un 1 si el directorio es creado con exito, 0 en caso contrario.

- **void write_zero(FILE* disk)**
"""
FILE* disk: puntero a disk
"""

Escribe 27 Byte de 0s correspondiente al nombre de un elemento (borra el nombre).

- **void free_used_block(uint32_t dir)**
  """
  unit32_t dir: direccion de memoria de bloque
  """
  Esta función recibe la direccion de memoria de dir y libera el bitmap asociado al el (cambia un 1 por un 0)

- **void rm_index(FILE* disk, uint32_t dir)**
  """
  FILE* disk: puntero a cierta direccion del disco
  uint32_t dir: direccion de memoria de un directorio
  """
  Esta funcion se encarga de identificar la informacion del bloque indice de un archivo. Identifica los punteros de direccionamiento indirecto simple, doble o triple segun corresponda para proceder a borrar el directorio.

- **int change_dir_file(FILE* disk, uint32_t dir, char* filename)**
  """
  FILE* disk: Puntero al disco
  uint32_t dir: direccion de memoria de un directorio
  char* filename: nombre del directorio que se desea eliminar
  """
  Elimina el directorio filename utilizando las funciones write_zero, write_pointer, rm_index, free_used_block. Retorna 1 en caso de exito, 0 en caso contrario.