



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2333 - SISTEMAS OPERATIVOS Y REDES

Documentación Proyecto Sistemas Operativos

October 8, 2019

2º semestre 2019 - Profesor Cristián Ruz

Grupo Maniha

1 Funciones Generales

Funciones definidas como generales en el enunciado y se encuentran en "cr_API.c".

1.1 cr_mount(char* diskname)

Función que recibe la ruta local hacia el disco que se quiere utilizar.

Se levantan dos errores y se finaliza la ejecución del programa en caso de que la ruta entregada no sea correcta o que el archivo objetivo no sea un archivo con extension binaria (".bin") como espera.

Retorna el puntero a la posición local del archivo ".bin" (nuestro disco virtual) a una variable global definida en "cr_API.h" llamada "DISK_PATH".

1.2 cr_bitmap(unsigned block, bool hex)

1.3 cr_exist(char* path

Función que recibe la ruta absoluta desde el directorio root a un archivo o directorio específico.

Utiliza la función *recorrer_path* para saber si existe un directorio o archivo en la ruta especificada en la ruta entregada.

Retorna 1 si existe y 0 en caso contrario

1.4 `cr_ls(char* path)`

Función que recibe la ruta absoluta desde el directorio root a un archivo o directorio específico.

Si el objetivo es un directorio, lista en consola todos los archivos y subdirectorios que están dentro de él. Si es un archivo, retorna un mensaje diciendo que el objetivo es un archivo. Si la ruta está mala, retorna un error en consola.

Esta función es de tipo void, por lo que no retorna nada.

1.5 `cr_mkdir(char* foldername)`

Recibe la ruta con objetivo final el nombre del directorio que se quiere crear dentro del disco.

Utiliza la ruta entregada para crear un directorio con el nombre del último elemento de la ruta dada (desde ahora el objetivo) en la ruta dada.

Retorna un 1 si se logró crear el directorio en la ruta y un 0 en caso contrario.

2 Funciones de Manejo de Archivos

Funciones definidas como de manejo de archivos en el enunciado y se encuentran en "cr_API.c".

2.1 `cr_open(char* path, char mode)`

Función que recibe la ruta absoluta desde el directorio root a un archivo específico y el modo en el que se quiere abrir el archivo.

Dado el modo entregado, cambia la funcionalidad general. Primero se crea un `crFILE` y se almacena un puntero hacia la estructura. En caso de que `mode` sea 'r', recorre los bloques del archivo especificado por la ruta dentro del disco, generando una representación de los datos en los bloques de tipo `dir` y guardándolas dentro del `crFILE` creado inicialmente, por lo que se rellena esta estructura con bloques y retorna la estructura `crFILE` con la representación del archivo.

En cambio, si el modo es 'w', primero verifica que no exista un archivo con el mismo nombre en la ruta especificada y genera una representación de un nuevo archivo por medio de la creación de un `crFILE` que lo represente. Retorna un puntero al `struct crFILE` creado.

2.2 cr_read(crFILE* file_desc, void* buffer, int nbytes)

2.3 cr_write(crFILE* file_desc, void* buffer, int nbytes)

2.4 cr_close(crFILE* file_desc)

2.5 cr_rm(char* path)

2.6 cr_unload(char* orig, char* dest)

Recibe una ruta de origen que existe dentro del disco y una ruta dentro del computador.

Si la ruta destino no existe, se crean los directorios necesarios para que la ruta exista, luego se revisa el archivo dentro del disko, se lee y se copia la información dada dentro de los bloques de datos en un nuevo archivo que se encontrará dentro de la ruta entregada como destino.

Retorna un 1 en caso de que se logre realizar la copia correctamente y un 0 en caso contrario

2.7 cr_load(char* orig)

3 Funciones Utiles

Funciones definidas en "Util.c".

3.1 encontra_directorio(char* path, int posición)

Función que recibe la ruta absoluta desde el directorio root a un archivo o directorio específico y la posición del bloque de directorio en el que se realizará la búsqueda.

Dada la posición entregada, es decir, el directorio dentro del cual se quiere buscar, se compara cada entrada del directorio con el nombre del archivo, para saber si existe dicho archivo o directorio dentro del directorio en el que se quiere buscar.

Retorna un puntero al directorio o archivo encontrado y NULL en caso de que no exista dicho archivo o directorio.

3.2 recorrer_path(char* path)

Función que recibe la ruta absoluta desde el directorio root a un archivo o directorio específico.

Al ser un ruta absoluta se aprovecha la existencia del caracter "/" como separador y se itera para avanzar por los directorios utilizando los nombres de estos. Una vez encontrado cada nombre de subdirectorio o archivo, se utiliza la función *encontrar_directorio* para saber si existe dentro del directorio en el que nos encontramos. En caso de ser un directorio y no ser el objetivo, se procede a guardar su posición para poder comenzar a buscar dentro de este en la proxima iteración.

Finalmente retorna un puntero al directorio o archivo final.

3.3 **dec_to_bin(int decimal, int* bin_array)**

Recibe un numero decimal y un puntero a un arreglo de enteros.

Transforma el numero decimal a un numero binario y lo guarda en el arreglo entregado como segundo parametro, por lo que se modifica el interior de dicho arreglo, del cual ya se tiene el puntero.

Esta función es de tipo void, por lo que no retorna nada.

3.4 **bin_to_dec(int* bin_array)**

Recibe un puntero a un arreglo de int.

Transforma el numero binario alojado dentro del arreglo recibido, transformandolo a su equivalente en número decimal.

Retorna el numero decimal equivalente al binario recibido.

3.5 **first_free_block()**

No recibe ningún argumento.

Recorre los bloques del disco en forma secuencial, revisando las validez de cada bloque por el que pasa en busqueda de un bloque invalido.

Retorna el numero de del primer bloque que se encuentra actualmente invalidado.

3.6 **obtener_nombre(char* path)**

Función que recibe la ruta absoluta desde el directorio root a un archivo o directorio especifico.

Utiliza la forma de escritura de un path absoluto, aprovechando la existencia del caracter '/' como separador de carpetas, para obtener el nombre del archivo o directorio objetivo.

Retorna el puntero a un arreglo de caracteres con el nombre del objetivo de la ruta.

3.7 directorio_a_agregar(char* path)

Recibe una ruta absoluta desde root a un archivo o directorio.

Utiliza la estructura de un ruta absoluta para crear un arreglo con la ruta hasta el directorio en el que se encuentra contenido el archivo o directorio objetivo.

Retorna la ruta hasta el directorio que contiene al objetivo.

3.8 objective_kind(char*path)

Recibe el ruta absoluta desde root a un archivo o directorio.

Utiliza el bit de validez del directorio o archivo objetivo para saber si es un archivo o un directorio.

Finalmente retorna un 1 si el objetivo es directorio, un 0 si es archivo y un 23 en caso de que no exista la ruta dada.

3.9 print_all(int posicion)

Recibe la posición de un archivo o directorio específico.

A partir de la posición entregada, imprime en consola todos los subdirectorios y archivos que se encuentran contenidos en el directorio específico. Si el directorio tiene un directorio de continuacion, instancia a `print_2_all` para que imprima imprima todo lo que está dentro de dicho bloque de directorios.

Esta función es de tipo void, por lo que no retorna nada.

3.10 print_2_all(int posicion)

Recibe la posición de un archivo o directorio específico.

A partir de la posición entregada, imprime en consola todos los subdirectorios y archivos que se encuentran contenidos en el directorio específico.

Esta función es de tipo void, por lo que no retorna nada.

3.11 **change_bitmap_block(int original_block)**

Recibe el número de un bloque.

Encuentra la entrada correspondiente al numero del bloque ingresado dentro de los bloques de bitmap y cambia su valor a 1 si este estaba en 0 y viceversa, actualizando el bitmap.

Esta función es de tipo void, por lo que no retorna nada.

3.12 **print_ls(char* path)**

Recibe el ruta absoluta desde root a un archivo o directorio especifico.

Permite avanzar por los directorios hasta encontrar el objetivo y utiliza la función *encontrar_directorio* para obtener la posición del directorio dado, una vez que tiene la posicion donde se encuentra el archivo objetivo, entrega dicha posición a *print_all*.

Esta función es de tipo void, por lo que no retorna nada.

3.13 **agregar_primeroinvalido2(int posicion, char* nombre, int puntero)**

Recibe el número del bloque del directorio dentro del cual se creará el archivo, el nombre del archivo a crear y el número del bloque directorio del archivo.

Busca el primer bloque inválido dentro del directorio en el que se va a crear el archivo y asigna un bloque a dicha posición, el bloque tendra como nombre el nombre del archivo que se quiere crear y validez 4.

Retorna un 1 en caso de que se agregue correctamente y un 0 en caso contrario.

3.14 **agregar_primeroinvalido(int posicion, char* nombre, int puntero)**

Recibe el número del bloque del directorio dentro del cual se creará el archivo, el nombre del archivo a crear y el número del bloque directorio del archivo.

Busca el primer bloque inválido dentro del directorio en el que se va a crear el archivo y asigna un bloque a dicha posición, el bloque tendra como nombre el nombre del archivo que se quiere crear y validez 4. En caso de que el primer bloque invalido del directorio se encuentre en el directorio de continuación, instancia a *agregar_primeroinvalido2* para que revise dentro de este.

Retorna un 1 en caso de que se agregue correctamente y un 0 en caso contrario.

3.15 isBin(char* path)

Recibe el ruta absoluta desde root a un archivo o directorio especifico.

Se preocupa de ver si el archivo objetivo de la ruta es de extensión ".bin", compara las ultimas 4 letras de la ruta para saber esto.

Retorna un char* con un string de los ultimos 4 caracteres de la ruta.

3.16 agregar_carpeta_invalido2(int posicion, char* nombre, int puntero)

Recibe el número del bloque del directorio dentro del cual se creará el nuevo directorio, el nombre del directorio a crear y el número del bloque directorio de este.

Busca el primer bloque inválido dentro del directorio en el que se va a crear el uno nuevo y asigna un bloque a dicha posición, el bloque tendra como nombre el nombre del directorio que se quiere crear y validez 2.

Retorna un 1 en caso de que se agregue correctamente y un 0 en caso contrario.

3.17 agregar_carpeta_invalido(int posicion, char* nombre, int puntero)

Recibe el número del bloque del directorio dentro del cual se creará el nuevo directorio, el nombre del directorio a crear y el número del bloque directorio de este.

Busca el primer bloque inválido dentro del directorio en el que se va a crear el uno nuevo y asigna un bloque a dicha posición, el bloque tendra como nombre el nombre del directorio que se quiere crear y validez 2. En caso de que el bloque invalido se encuentre dentro del directorio de continuación, instancia a agregar_carpeta_invalido2 para que revise dentro de este.

Retorna un 1 en caso de que se agregue correctamente y un 0 en caso contrario.

3.18 leer_bloques_directos(crFILE* file_desc, uint8_t* buffer, int nbytes)

3.19 get_entry_index(int dir_block, char* path)

3.20 get_file_pointer(int dir_block, char* path)

3.21 invalidate_entry(int dir_block, int entry_index)

3.22 free_simple_indirect(int simple_block)

3.23 free_double_indirect(int double_block)

3.24 free_triple_indirect(int triple_block)

3.25 actual_locals(char* path)

Recibe una ruta absoluta de directorios.

Revisa directorio a directorio si estos existen.

Retorna la ruta de los directorios que ya existen en la ruta entregada.

3.26 locals_to_create(char* path)

Recibe una ruta absoluta de directorios.

Instancia a actual_locals para poder sacar que directorios existen en la ruta entregada inicialmente y separa las rutas existentes y las no existentes.

Retorna la ruta de los directorios que no existen en la ruta entregada.

3.27 get_first_folder(char* path)

Recibe una ruta absoluta de directorios.

Utiliza las propiedades de las rutas absolutas para encontrar y separar el primer directorio de la ruta entregada.

Retorna el primer directorio de la ruta entregada con un '/' al comienzo.

3.28 next_folder(char* real_path, char* new_folder)

Recibe una ruta existente en el local actual y el nombre de un directorio a crear con un '/' al inicio.

Realiza el manejo de strings para poder juntar la ruta existente y el nuevo directorio, dejandolo como una ruta valida hacia el directorio que se quiere crear.

Retorna la ruta existente sumandole el nombre del directorio que se quiere crear.

3.29 create_local_directory(char* path)

Recibe una ruta absoluta de directorios.

Función que instancia a `actual_locals`, `locals_to_create`, `get_first_folder` y `nex_folder` para comparar la ruta entregada con las rutas reales en el computador local, para generar de forma recursiva los directorios no existentes en el computador creando la ruta entregada.

No retorna nada debido a que es void, pero crea en el computador la ruta especificada.