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

Documentación SIGKILLERS

24 de mayo de 2019

Primer semestre 2019

Funciones

Generales

- `void cr_mount(char* diskname)`

  Función que recibe un nombre de un disco (por ejemplo "simdiskfilled.bin") y lo carga como variable global llamada `disk_path`

  De esta forma el disco que montado y puede ser abierto por cualquier función de la API.

- `void cr_bitmap()`

  Función que imprime el Bitmap como una secuencia de '1's y '0's. Cada 1 significa que el bloque en ese índice de la secuencia se encuentra ocupado y cada 0 significa que el bloque en ese índice de la secuencia está libre.

  En la siguiente línea imprime la cantidad de bloques que se encuentran ocupados en el disco y en la última línea se imprimen la cantidad de bloques libres del disco.

- `int cr_exists(char* path)`

  Función que recibe un `path` hacia un archivo o hacia un directorio y retorna 1 si es que el archivo o directorio existe en el disco según el `path` entregado o 0 en caso contrario.

- `void cr_ls(char* path)`

  Función que recibe un `path` hacia un directorio e imprime los archivos y subdirectorios contenidos en este.

  Para imprimir los archivos y subdirectorios del directorio raíz, se debe ingresar como path un string vacío `''`.

  En caso de que el `path` no exista, se imprime en consola para comunicar al usuario.

- `int cr_mkdir(char* foldername)`

  Primero se verifica que la ruta en donde se quiere crear el directorio exista.

  Después, si el nombre de carpeta no se encuentra ya en el directorio, se creará una nueva entrada de directorio en la ruta y se creará un nuevo bloque de directorio en el primer bloque que se encuentre vacío.

  Si el directorio ya existe o no hay memoria disponible para crear uno, retornará 0. Si se crea exitosamente, retorna 1.

Archivos

- `crFILE* cr_open(char* path, char mode)`

  Primero, esta función divide el `path` ingresado en una ruta de directorio y un nombre de archivo. Con esto verifica que la ruta de directorio exista.

  Luego, si el `mode` ingresado es 'r', verificará que el archivo existe en ese directorio y retornará una estructura crFILE con un puntero hacia ese archivo.

  Si el archivo no existe en el directorio y `mode` es 'w', se creará una nueva entrada de archivo en ese bloque de directorio que apunte a un nuevo bloque índice que es inicializado en esta misma función (todos sus valores quedan en 0 menos el número de hardlinks que queda en 1).

  Para que lo anterior suceda, el bloque de directorio debe tener alguna entrada inválida y deben quedar bloques vacíos en el bitmap. Aquí se cambiará a 1 el primer bloque 0 encontrado.

  En los casos contrarios a los mencionados anteriormente, la función retornará 'NULL' e imprimirá un mensaje que advierta el error.

- `int cr_read(crFILE* file_desc, void* buffer, int nbytes)`

  Función que recibe un puntero `file_desc` a un `crFILE`, un `buffer` y una cantidad de bytes `nbytes`. Lo que hace es leer el contenido de `file_desc` que es un archivo del disco y cargar `nbytes` de ese archivo en el `buffer`.

  Como no se especificaba, se decidió que la función primero leía los 2048 Bytes del bloque índice del archivo y luego de esto, empezaba a leer los Bytes del contenido del archivo.

  La función retorna un `int` que es la cantidad de Bytes que fueron leídos del archivo. En caso de que se intente leer más Bytes de los que el archivo está usando, se va a leer solo la cantidad en uso.

En caso de tratar de leer más Bytes del tamaño máximo de un archivo en el disco, se imprime que se superó el tamaño máximo y se retorna la cantidad de Bytes que se alcanzaron a leer antes de superar el máximo.

- `int cr_write(crFILE* file_desc, void* buffer, int nbytes)`

  Función para escribir archivos en el disco .bin. Recibe un crFILE el cual se utiliza para ubicar la dirección en que se van a escribir los nbytes desde la dirección de buffer.

  Su funcionamiento consta de buscar dentro de los 4 bloques de bitmap el primer bit que sea igual a cero, guardando posteriormente el valor de este índice para saber donde se encuentra este bloque de datos.

  Se hace lo mismo pero dentro del bloque índice del archivo, en donde se busca el primero puntero desocupado, para buscar la referencia a un bloque de datos en donde se pueda escribir. En caso de que no se encuentre un puntero desocupado se siguen revisando los siguientes 40 bytes para buscar los punteros de bloques de direccionamiento indirecto. Si es que en estos se encuentra uno vacío, se ingresa a este bloque. Luego dentro de este bloque, se busca el primer puntero vacío y se escribe en esa dirección.

  Para todos los casos se actualiza el bitmap respectivo a uno y se actualiza el puntero vacío a la direccón del bloque.

  Se escribe hasta lo que quepa de memoria.

  Se retorna el número de bytes que se pudieron escribir.

- `int cr_close(crFILE* file_desc)`

  Se preocupa de liberar los recursos pedidos por `cr_open`.

- `int cr_rm(char* path)`

  Función que se encarga de borrar el archivo referenciado por la ruta `path`.

  Solo borra el archivo si no quedan más `hardlinks` que apunten a el. En caso contrario, solo se borra la referencia en el `path` correspondiente.

  Si todo sale bien, se retorna `0`. En caso de que el `path` indicado no exista, se imprimirá un mensaje en consola indicando esto y se retornará `0`

- `int cr_hardlink(char* orig, char* dest)`

  Se verifica que ambas rutas sean válidas, que el archivo exista en la ruta de origen y que no exista en la ruta de destino.

  Luego se crea una nueva entrada en el bloque de directorio de la ruta de destino con los mismos datos de la entrada que apunta al archivo desde el directorio de origen.

  Finalmente se suma 1 a la cantidad de hardlinks que apuntan al archivo.

Retorna 1 en caso de éxito y 0 en caso de que no se haya podido crear, ya sea por falta de memoria o por rutas inválidas.

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

  Función que recibe un path `orig` a una carpeta o archivo en el disco y un path `dest` a una carpeta o archivo en el computador del usuario. En caso de tener un archivo como origen, lo copia hacia el path de destino. En caso de tener un directorio como origen, copia todos los archivos dentro de él y sus subdirectorios (recursivamente) también hacia el path de destino.

  Si los directorios de destino no existen, los crea en el computador.

  Retorna 0 en caso de error y que el path de origen no exista y 1 en caso de copia exitosa.

- `int cr_load(char* orig)`

  Función que se encarga de copiar un archivo o árbol de directorios, referenciado por orig al disco. Si el archivo es demasiado pesado para el disco, se escribir hasta utilizar todo le espacfio disponible.