Documentazione del progetto di Sistemi Operativi

Implementazione File System

È stato implementato un filesystem simulando, con un file binario, un disco rigido.

BitMap

La BitMap è stata implementata per mappare blocchi liberi e occupati.

Per l'implementazione della BitMap sono state utilizzate due struct.
La prima struct si occupa della BitMap vera e propria e i campi che la caratterizzano. Sono importanti infatti il numero dei bit, il numero dei blocchi e le entries.
La seconda struct caratterizza le entry della bitmap, sono importanti il numero dell'entry e il numero del bit.
Per una corretta esecuzione della BitMap sono state implementate le seguenti funzioni:

- void BitMap_init(BitMap b, int bitmap_blocks, int disk_blocks, int occupation);*
Inizializza una nuova BitMap vuota.
È una funzione void, quindi non avrà nessun valore di ritorno ma inizializzerà tutti i campi della struct a dei valori di default che indicano che il disco è completamente vuoto.

- BitMapEntryKey BitMap_blockToIndex(int num);
Prende come parametro un intero che indica il numero dei blocchi e restituisce un BitMapEntryKey popolata con rispettivamente il primo campo della struct che indica l'entry della BitMap che contiene l'informazione sul blocco e il secondo campo indica lo spiazzamento all'interno del blocco per individuare il bit di cui abbiamo bisogno.

- int BitMap_indexToBlock(int entry, uint8_t bit_num);
Funzione elementare che converte un bit in un indirizzo lineare.

- int BitMap_get(BitMap bmap, int start, int status);*
Funzione che restituisce l'indice del primo bit con stato status nella BitMap bmap e inizia a cercare dalla posizione start in poi. Restituisce -1 se non viene trovato.

- int BitMap_set(BitMap bmap, int pos, int status);*
Funzione che setta il bit con indice pos a status. Ha come valore di ritorno status in caso di successo.

- int BitMap_test(BitMap bmap, int pos);*
Funzione che ritorna lo status del bit con indice pos all'interno della BitMap bmap.

Common

Il file common.h contiene la macro CHECK_ERR molto importante poiché permette di verificare se una funzione è andata a buon fine, altrimenti stampa un messaggio di errore e chiude il programma.
Al suo interno contiene le variabili globali che si utilizzano all'interno dell'intero progetto SimpleFS.

Diskdriver

Il DiskDriver è il modulo che gestisce la lettura e la scrittura dei blocchi del disco rigido, permetter appunto di scrivere e leggere su un blocco già allocato e di allocare e deallocare un blocco, per controllare o stato di allocazione di un blocco si usa la bitmap presente su disco.
Permette di inizializzare le strutture di base necessarie al Filesystem (Header del disco e bitmap) in caso di formattazione del disco.

Per una corretta e lineare esecuzione del DiskDriver sono state utilizzate due struct.
La prima struct serve per caratterizzare il DiskDriver attraverso 3 parametri che corrispondono a un puntatore ad una struttura DiskHeader, un puntatore ad una BitMap e un file descriptor riguardante il file su cui si crea il filesystem.
La seconda struttura caratterizzare DiskHeader attraverso i campi che identificano: numero dei blocchi, numero dei blocchi usati, numero dei blocchi liberi e l'indice del primo blocco libero.

-void DiskDriver_init(DiskDriver disk, const char* filename, int num_blocks);*
Funzione che crea il file che rappresenta il disco, allocando lo spazio necessario per il disco, attraverso un calcolo delle dimensioni più grandi che può assumere la bitmap.

-int DiskDriver_load(DiskDriver disk, const char* filename);*
Funzione che si occupa del caricamento di un disco già esistente.

-int DiskDriver_readBlock(DiskDriver disk, void* dest, int block_num);*
Funzione che legge il blocco in posizione block_num. Ha come valore di ritorno -1 se il blocco risulta libero nella BitMap e quindi non può essere letto, 0 altrimenti.

-int DiskDriver_writeBlock(DiskDriver disk, void* src, int block_num);*
Funzione che scrive il blocco in posizione block_num. Ha come valore di ritorno -1 se l'operazione non è consentita, 0 altrimenti. Se il blocco in posizione block_num è libero nella BitMap, esso viene scritto, altrimenti viene modificato.

-int DiskDriver_freeBlock(DiskDriver disk, int block_num);*
Funzione che libera il blocco all'interno del disco in posizione block_num. Ritorna -1 se l'operazione non è possibile.

-int DiskDriver_getFreeBlock(DiskDriver disk, int start);*

Funzione che ritorna l’indice del primo bocco libero all’interno del disco, blocchi liberi cercati in maniera circolare.

-int DiskDriver_flush(DiskDriver disk);*
Funzione che obbliga il sistema a scrivere le modifiche attualmente salvate nei buffer di sistema.

-void DiskDriver_shutdown(DiskDriver disk);*
Funzione che si occupa dello svuotamento dei buffer riguardanti il disco e della chiusura dei file relativi al DiskDriver.

Simplefs

Il FileSystem permette la creazione di un albero generico composto da directory e file.

Un File è composto da un blocco iniziale contentente tutti i metadati, più due linked lists, una che contiene tutti i blocchi dati del file e l’altra che contiene le posizioni su disco dei blocchi dati, indicizzandoli e permettendo un accesso random più veloce.

Una Directory è un file composto da una sola linked list, che contiene la posizione su disco del primo blocco dei files che sono contenuti in essa.

Per l’implementazione della Simplefs sono state utilizzate undici struct.
-BlockHeader che si trova nella prima parte di ogni blocco e rappresenta una lista concatenata di blocchi. In particolare nella struttura sono presenti : blocco precedente e successivo , posizione nel file e la posizione ripetuta del blocco sul disco.
-FileControlBlock struttura che contiene tutte le informazioni relative ad un file come per esempio esprime se è un file o una directory, la dimensione e il directory_block.
-FirstfileBlock struttura che indica tutte le proprietà base del file, contiene l’fcb, la posizione del cursore, la directory padre una parte di indici e il successivo blocco di indici.
-Index struttura che indicizza i blocchi di dati, salvando la loro posizione su disco per un accesso random più veloce.
-FileBlock(molto simile a DirectoryBlock) struttura che indica il blocco dati di un file.
-FirstDirectoryBlock indica il primo blocco di una directory, contiene tutti i metadati della directory.
-SimpleFS struttura che indica la generica forma del nostro filesystem, in particolare contiene il numero dei blocchi e il filename.
-FileHandle(molto simile a DirectoryHandle) struttura usata per riferirsi a file aperti(Directory aperte nel caso di DirectoryHandle).
-SearchResult struttura che contiene il risultato di un'operazione di ricerca in una determinata directory di blocchi di file o directory, se mancanti, sono NULL.

Per una corretta esecuzione della Simplefs sono state implementate le seguenti funzioni:

-DirectoryHandle SimpleFS_init(SimpleFS* fs, DiskDriver* disk);*
Funzione che inizializza il file system su un disco già esistente e restituisce un handle alla directory di livello superiore memorizzata nel primo blocco.

-void SimpleFS_format(SimpleFS fs);*
Funzione che serve per formattare e reinizializzare un file system già esistente. Attraverso questa funzione verrà resettata anche la BitMap dei blocchi occupati sul disco.

-FileHandle SimpleFS_createFile(DirectoryHandle* d, const char* filename);*
Funzione che crea un file vuoto nella directory d e restituisce NULL in caso di errore o altrimenti
un handle ad un file vuoto.

*-int SimpleFS_readDir(char** names, DirectoryHandle* d);*
Funzione che legge il nome di tutti i file e le directory figli della directory passata come parametro.

-FileHandle SimpleFS_openFile(DirectoryHandle* d, const char* filename);*
Funzione che apre un file già esistente nella directory d, allocando e restituendo un FileHandle.

-void SimpleFS_close(FileHandle f);*
Funzione che chiude un file già esistente dellocando il FileHandle fornito e le strutture conenute al suo interno.

-int SimpleFS_write(FileHandle f, void* data, int size);*
Funzione che restituisce il numero di byte scritti, che sovrascrive e alloca nuovo spazio se necessario per scrivere nel file.

-int SimpleFS_read(FileHandle f, void* data, int size)*;
Funzione che restituisce il numero di byte letti,che sovrascrive e alloca nuovo spazio se necessario per leggere il file.

-int SimpleFS_seek(FileHandle f, int pos);*
Funzione che restituisce il numero di byte letti spostando il puntatore corrente in pos, che sarà restituito in caso di successo, -1 in caso di errore (file troppo corto).

-int SimpleFS_changeDir(DirectoryHandle d, const char* dirname);*
Funzione che cerca una directory in d. Se dirname è uguale a "..", sale di un livello. La funzione restituisce 0 in caso di successo, valore negativo in caso di errore. La funzione ha modifica il DirectoryHandle fornito in input.

-int SimpleFS_mkDir(DirectoryHandle d, const char* dirname);*
Funzione che crea una nuova directory figlia della directory passata come parametro. La funzione restituisce 0 in caso di successo, -1 in caso di errore.

-int SimpleFS_remove(DirectoryHandle d, const char* filename);*
Funzione che rimuove il file nella directory corrente. La funzione restituisce 0 in caso di successo,-1 in caso di errore. Se viene rimossa una directory, saranno eliminati tutti i file e le directory contenuti in essa.

Shell

Parte integrante del nostro progetto è anche una shell che permette agli utenti di interagire con il sistema operativo. Essa funziona attraverso funzioni da noi implementate e permette l'utilizzo delle seguenti operazioni :

- quit
  Per chiudere la nostra shell.
- ls
  per mostrare l'elenco dei file contenuti nella working directory.
- cd
  per cambiare directory di lavoro.
- mkdir
  per creare una directory.
- rm
  Per rimuovere un file o una directory.
- touch
  per creare un nuovo file vuoto.
- cat
  per concatenare i file e scriverli sullo schermo.
- cp
  per copiare un file da o verso SimpleFs.
- info
  per avere informazioni sul disco.
- info -bmap
  per avere info sulla bitmap del disco.
- echo
  per replicare un messaggio o per scriverlo su un file.

Il progetto è composto da due file eseguibili, generabili tramite il comando make:

- shell: l'eseguibile della shell, non prende agromenti
- simplefs_test: che esegue una serie di test preliminari per verificare il corretto funzionamento del filesystem, anch'esso non prende argomenti.

Fortunato Tocci
Gabriele Sellani
Mattia Nicolella