PROGETTO DI PROGRAMMAZIONE PER STUDENTI PART-TIME

Docenti: A. Marin, S. Rota Bulò

Si desidera simulare il funzionamento di un semplice file system. A tal fine si memorizzeranno delle stringhe al posto dei file e il loro reperimento, stampa, cancellazione ecc... modelleranno le classiche operazioni su file. Il nostro disco fittizio è costituito da tanti blocchi della stessa dimensione che possono memorizzare al massimo N byte più un intero che denota il numero del prossimo blocco e un altro intero che denota quanti degli N byte sono effettivamente occupati. Quando un file da memorizzare è di dimensione M > N byte, esso va spezzettato in più blocchi. Supponiamo che il primo blocco sia in posizione n_1, il secondo in posizione n_2, ecc.. Allora il blocco n_1 conterrà N byte e l'intero associato conterrà il valore n_2. Il blocco n_2, a sua volta, conterrà gli N byte successivi a quelli memorizzati in n_1 e il suo intero conterrà n_3. L'intero associato all'ultimo blocco conterrà un carattere riservato (ad esempio -1) che indica che il file è terminato. Ad ogni file è attribuito un nome e un intero che identifica il numero del suo primo blocco (nell'esempio n_1).
Si desidera implementare:

1. Una struttura dati in grado di memorizzare le informazioni nella modalità specificata
2. L'operazione di inserimento di un file
3. L'operazione di cancellazione di un file
4. L'operazione di lettura di un file
5. L'operazione di lettura di un file da un carattere *start* ad un *carattere* end
6. L'operazione di tentativo di recupero di tutti i file erroneamente cancellati
7. Un file di test che controlli le funzionalità del programma

Da un punto di vista della simulazione, come già detto, si memorizzano stringhe al posto di byte del file, l'operazione di lettura consisterà nella stampa della stringa (o della porzione richiesta). Attenzione! A causa delle cancellazioni i file possono essere memorizzati in blocchi non contigui.

Strutturazione della parte in C
La struttura dati che modella il disco sarà un array di MAXDIM di strutture come la seguente:

```
struct blocco{
    char info[5];
    int blocco_succ;
    int byte_usati;
};
```

Un blocco è libero se `byte_usati` contiene `0` è la stringa vuota.
La struttura dai che memorizzerò il nome e i blocchi iniziali dei file sarà invece una lista semplice, la cui struttura della cella sarà la seguente:

```
typedef struct cella* t_listapos;
struct cella{
    char* nomefile;
    int primoblocco;
    t_listapos next;
};
```

Si ponga attenzione alle gestione dei nomi dei file essendo il tipo nella cella un puntatore a carattere e non una stringa. Le firme dei metodi da implementare sono le seguenti:

```
int crea_file(char nome[], char contenuto[], t_listapos* fs,
struct blocco[] disco);
```

restituisce `0` se il file è stato correttamente memorizzato, `1` altrimenti (esempio, non c'è spazio sufficiente).

`int cancella_file(char nome[], t_listapos* fs, struct blocco[] disco);`
cancella solo la cella descrittore del file e marca tutti i blocchi da esso usati come liberi senza rimuoverne l'informazione. Restituisce `0` se il file esiste, `1` altrimenti.

`int stampa_file(char nome[], t_listapos fs, struct blocco[] disco);`
stampa la stringa che modella il file `nome`. Restituisce `0` se il file esiste, `1` altrimenti.

`int stampa_file_pos(char nome[], int da, int a, t_listapos fs, struct blocco[] disco);`
stampa la stringa che modella il file `nome` dal byte `da` al byte `a`. Restituisce `0` se il file esiste, `1` altrimenti.

`recupera_file(t_listapos* fs, struct blocco[] disco)`
Dall'analisi della struttura del disco recuperare i file cancellati attribuendo il nome `file1`, `file2` ecc... Ovviamente può accadere che un file cancellato non sia (interamente) recuperabile. Questa è un'operazione da svolgersi al “best effort”.