Laboratorio di Sistemi Operativi I
Anno Accademico 2007-2008

Micro Filesystem (ufs): specifiche per il progetto d'esame/esonero n.2

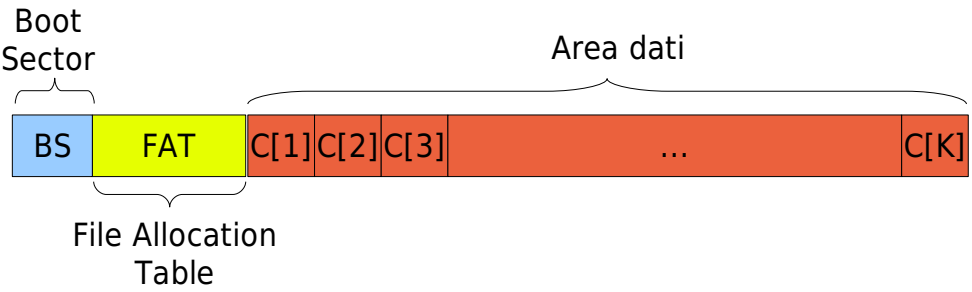
Struttura del Micro Filesystem (µFs)

***ATTENZIONE**: le specifiche evidenziate in giallo NON si applicano all'esonero n.2, ma soltanto al progetto, nel caso in cui si sia scelta come funzionalità opzionale aggiuntiva il supporto dei permessi di accesso ai file (owner/others, read/write). Comunque i campi delle strutture dati necessari per la gestione dei permessi vanno mantenuti (NON vanno quindi eliminati) anche se NON gestiti.*

Il Micro Filesystem deve essere organizzato all'interno di un **unico file regolare** Linux (con filename predefinito `dev_ufs`) che rappresenta, ai fini del progetto/esonero n.2, una partizione di un ipotetico hard disk. La sua struttura dati deve essere basata su un **filesystem FAT-32 semplificato** con gestione opzionale dei permessi in stile Unix/Linux.

Il filesystem è composto da 3 aree:

- **Boot Sector**: contiene le informazioni generali sulla partizione. La dimensione del boot sector è di 32 byte.
- **File Allocation Table (FAT)**: è l'indice della partizione. Mappa i diversi file o directory sui cluster (ovvero blocchi, nella terminologia Unix/Linux) dell'area dati.
- **Area dati**: è divisa in cluster. Un file viene memorizzato in uno o piu' cluster, non necessariamente consecutivi.



Boot Sector

Le informazioni generali sulla partizione sono contenute nei primi 32 byte di `dev_ufs`: il Boot Sector (nella terminologia Unix è chiamato superblock).

All'interno del Boot Sector i numeri, rappresentati da interi da 2 byte o 4 byte, sono memorizzati nella modalità **little-endian** (dal byte meno significativo al byte più significativo: comunque tale precisazione dovrebbe essere ininfluente per chi sviluppa il progetto su architetture Intel x86, cioè la maggior parte dei PC).

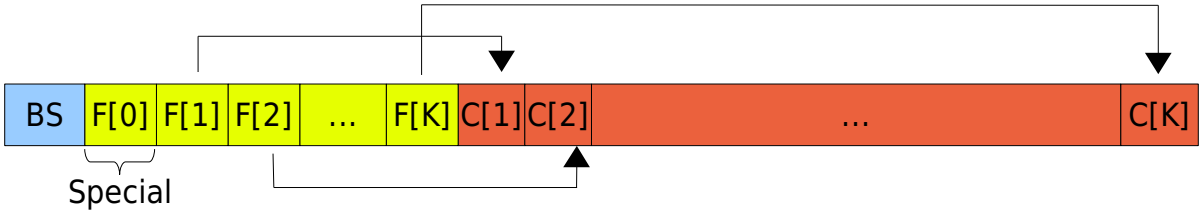
Il contenuto del boot sector è rappresentato dalla seguente tabella:

byte	Dimensione (in byte)	Descrizione
0-1	2	Dimensione in byte di un cluster (valori validi: 512, 1024, 2048, 4096, 8192)
2-5	4	Numero di cluster (K) presenti nell'area dati
6-9	4	Dimensione in byte della FAT
10-11	2	Numero di versione del filesystem sviluppato (byte 10: minor version, byte 11: major version)
12-15	4	Indice del primo cluster che contiene i dati della directory di root. Tale valore è sempre uguale a 1, cioè punta al primo cluster dell'area dati.
16-19	4	Dimensione totale in byte della directory di root
20-29	10	Nome attribuito alla partizione al momento della creazione del filesystem
30-31	2	Codice costante che definisce il tipo di filesystem: 0x44BB (byte 30: 0xBB, byte 31: 0x44)

File Allocation Table (FAT)

La tabella di allocazione dei file (FAT) è la struttura dati utilizzata per mappare i file e le directory del filesystem nei cluster dell'area dati. E' organizzata come una serie di liste linkate all'interno di un vettore di interi.

Ogni elemento *i*-esimo della tabella, che indicheremo con F[i], occupa 4 byte (32 bit) e descrive se il cluster C[i] non è correntemente allocato su alcun file/directory oppure se è utilizzato da qualche file.



La seguente tabella mostra i valori da utilizzare per ogni elemento F[i]:

Valore di F[i] (1 <= i <= K con K < 0xFFFFFFFF)	Significato
0x00000000	Cluster non allocato (libero)
0x00000001 – 0xFFFFFFFE	Cluster allocato. Il valore rappresenta un puntatore al prossimo cluster mediante il suo indice numerico.
0xFFFFFFFF	Ultimo cluster di un file (terminatore della lista linkata dei cluster assegnati ad un file/directory).

Il primo elemento F[0] della FAT è speciale in quanto utilizzato per sapere se il filesystem e' stato smontato correttamente. F[0] assume i seguerti valori:

Valore di F[0]	Significato
0x00000000	Partizione non montata
0x0000000F	Partizione montata

Esempio di FAT con K=9 con un unico file il cui cluster iniziale è il numero 1 (ovviamente e' solo un esempio ipotetico in quanto F[1] e' in realta' l'inizio della lista degli indici dei cluster allocati alla directory root).

i	0	1	2	3	4	5	6	7	8	9
F[i]	0x0F	0x04	0x00	0x00	0x06	0x00	0xFFFFFFFF	0x00	0x00	0x00

Dalla tabella FAT leggo che il cluster successivo è il numero 4 ed infine il numero 6. In questo caso per recuperare l'intero contenuto del file si devono leggere in sequenza i cluster 1, 4 e 6.

Area dati

L'area dati di `dev_ufs` è divisa in cluster e contiene i dati dei files e delle directory. Ogni file o directory è memorizzato su uno o più cluster.

I cluster vengono numerati da 1 a K, dove K è il numero di cluster presenti nell'area dati (vedi boot sector). Si ribadisce che il cluster 1 e' sempre il cluster iniziale della directory root.

I dati di un file sono memorizzati nella lista dei cluster allocati per il file stesso senza una struttura interna predefinita (come in Unix/Linux), ma come una sequenza di byte. Al contrario una directory è memorizzata sui vari cluster a lei riservati come una sequenza di strutture predefinite a lunghezza fissa denominate *directory entry,* una per ogni file/subdirectory contenuti nella directory stessa.

Directory entry

Le informazioni pertinenti ad ogni file o subdirectory contenuti all'interno di una certa directory sono rappresentate da un struttura dati di 32 byte, che ne contiene il nome, l'estensione, gli attributi, la data e l'ora di creazione, l'indice del primo cluster che contiene i dati del file/subdirectory e la sua dimensione in byte.

Ad ogni file/subdirectory creato in una certa directory viene assegnata una directory entry. Al momento in cui il file/subdirectory viene rimosso, la directory entry corrispondente diviene libera e, se necessario, tutte le directory entry successive vengono compattate per non lasciare mai una directory entry libera tra 2 allocate.

Il formato di ogni directory entry è il seguente:

byte	Dimensione (in byte)	Descrizione
0-11	12	Nome del file/directory (caratteri alfanumerici)
12-13	2	Attributi del file/directory
14-15	2	User ID (Linux UID) del proprietario del file/directory
16-19	4	Data e Ora di creazione del file/directory (numero di secondi dal 1/1/1970)
20-23	4	Data e Ora dell'ultima modifica del file/directory (numero di secondi dal 1/1/1970)
24-27	4	Indice del cluster iniziale che contiene i dati del file/directory
28-31	4	Dimensione in byte del file/directory

Attributi di un file/directory

Gli attributi di un file o directory (byte 12-13 di una directory entry) sono codificati dai seguenti valori, i quali possono essere combinati in OR bit a bit (eccetto che per la combinazione non ammessa di *File regolare* OR *Directory*):

Valore	Tipo del file
0x0001	File regolare
0x0010	Directory
Valore	**Permessi**
0x0400	Il proprietario ha il permesso in lettura
0x0200	Il proprietario ha il permesso in scrittura
0x4000	Gli altri hanno il permesso in lettura
0x2000	Gli altri hanno il permesso in scrittura

Data e ora di creazione o modifica di un file/directory

La data e l'ora di creazione e modifica di un file rispecchiano la convenzione Linux (vedi *man 2 time*) e quindi contengono il numero di secondi dall'epoca (epoch -> 01/01/1970).

Creazione del filesystem `dev_ufs`

Il filesystem deve essere creato con il seguente comando:

```
ufs_mke2fs [-c dim-cluster] [-k num-cluster] [-n nome-vol]
```

dove:

- **`-c dim-cluster`** specifica la dimensione in byte di un cluster dell'area dati. I valori validi sono: **512, 1024, 2048, 4096, 8192**. Il default è **1024**.
- **`-k num-cluster`** specifica il numero di cluster dell'area dati. Il default è **1024**.
- **`-n nome-vol`** specifica il nome attribuito alla partizione. Il default è **`lso1ufs`**.

Si ribadisce che il file regolare Linux su cui creare il filesystem ha nome fisso predefinito `dev_ufs`. Questo comando inizializza le seguenti strutture all'interno del file:

- Boot Sector (32 byte)
- FAT
- Root directory vuota

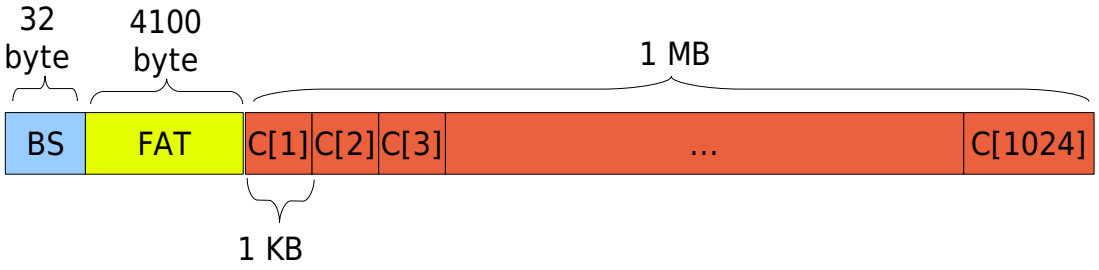
Con i dati disponibili al momento della creazione del filesystem mediante `ufs_mke2fs` si possono facilmente calcolare le dimensioni in byte delle varie aree del filesystem:

- dimensione dell'area dati: **dim-cluster * num-cluster**
- dimensione della FAT: **4 * (num-cluster + 1)**
- dimensione dell'intero file `dev_ufs`:

  32 + (4 * (num-cluster + 1)) + (dim-cluster * num-cluster)

Esempi di utilizzo del comando di creazione del filesystem

```
$ ufs_mke2fs
```




```
$ ufs_mke2fs -c 4096 -k 10000 -n myufs
```