Analisi della concorrenza sulle strutture dati di dev_ufs

ufs_syscall	tipo lock
ufs_creat	W
ufs_remove	W
ufs_rename	W
ufs_open	R
ufs_close	/
ufs_read	R
ufs_write	W
ufs_lseek	/
ufs_mkdir	W
ufs_rmdir	W
ufs_opendir	R
ufs_closedir	/
ufs_readdir	R

Record Locking (1)

- **L'implementazione piu' semplice per la gestione della concorrenza e':**
  - prendere il lock (R o W) su tutto il file (dev_ufs)
  - prendere il lock (R o W) sul primo byte del file (dev_ufs), questo byte e' utilizzato come un semaforo
    - Posso leggere dal file dev_ufs se ho il R-lock sul primo byte del file
    - Posso scrivere sul file dev_ufs se ho il W-lock sul primo byte del file
- **Questa soluzione e' corretta ma non e' efficiente**

Record Locking (2)

- **Ottimizzazione**
  - R-Lock sul boot sector
    - Permette di leggere il boot sector
  - W-Lock sul boot sector
    - Permette di modificare il boot sector (es. dimensione della directory root)
  - R-Lock sull'elemento *i*-esimo della tabella FAT
    - Permette di leggere F[i] e C[i]
  - W-Lock sull'elemento *i*-esimo della tabella FAT
    - Permette di modificare F[i] e C[i]

Record Locking (2): Esempio `ufs_creat`

- **`ufs_creat("/a1/a2/a3/pluto.txt", 0644)`**
- **Controllo del pathname**
  - `/a1/a2/a3` deve essere una directory
- **Creazione del file `pluto.txt` nella directory /a1/a2/a3**
  - Se il file esiste, viene troncato altrimenti viene creato
- **Procedimento:**
  - RLOCK(boot sector) per leggere la dimensione della directory root e l'indice del primo cluster (e' sempre 1)
  - `FAT(/)`: Indica tutti gli elementi della FAT che si riferiscono alla directory root
    - `i = bs.root_index`
    - `while (i != 0xFFFFFFFF)`
      - `RLOCK(&F[i])`
      - `i = F[i]`

Record Locking (2): Esempio `ufs_creat`

- RLOCK(`FAT(/)`)
- RELEASE(boot sector)
- Legge la directory root per trovare il file `a1` (deve essere una directory)
- RLOCK(`FAT(/a1)`)
- RELEASE(`FAT(/)`)
- Legge la directory `/a1` per trovare il file `a2` (deve essere una directory)
- WLOCK(`FAT(/a1/a2)`)
- RELEASE(`FAT(/a1)`)
- Legge la directory `/a1/a2` per trovare il file `a3` (deve essere una directory)
- WLOCK(`FAT(/a1/a2/a3)`)

Record Locking (2): Esempio `ufs_creat`

- Se il file `pluto.txt` esiste ed e' una directory torna errore
- Scrive la directory entry relativa al file `pluto.txt`
  - Se il file era gia' presente, prende il lock in scrittura su `FAT(/a1/a2/a3/pluto.txt)` e modifica opportunamente la FAT per liberare tutti i cluster utilizzati dal file `pluto.txt`
  - Se non c'e' spazio per scrivere la nuova directory entry
    - trova un elemento libero della FAT (RLOCK)
    - scrive sul cluster libero e modifica opportunamente la FAT (WLOCK)
  - Aggiorna la dimensione della directory `/a1/a2/a3` nella directory entry di `/a1/a2`
- RELEASE(`FAT(/a1/a2/a3)`)
- RELEASE(`FAT(/a1/a2)`)

Record Locking (2): Esempio `ufs_remove`

- **`ufs_remove("/a1/a2/a3/pluto.txt")`**
- **Controllo del pathname**
  - `/a1/a2/a3/pluto.txt` deve essere un file regolare
- **Libera i cluster utilizzati dal file ed aggiorna la directory**
- **Procedimento:**
  - RLOCK(boot sector) per leggere la dimensione della directory root e l'indice del primo cluster (e' sempre 1)
  - `RLOCK(FAT(/))`
  - RELEASE(boot sector)
  - Legge la directory root per trovare il file `a1` (deve essere una directory)
  - RLOCK(`FAT(/a1)`)

Record Locking (2): Esempio `ufs_remove`

- RELEASE(`FAT(/)`)
- Legge la directory `/a1` per trovare il file `a2` (deve essere una directory)
- WLOCK(`FAT(/a1/a2)`)
- RELEASE(`FAT(/a1)`)
- Legge la directory `/a1/a2` per trovare il file `a3` (deve essere una directory)
- WLOCK(`FAT(/a1/a2/a3)`)
- Legge la directory `/a1/a2/a3` per trovare il file `pluto.txt` (deve essere un file regolare)
- WLOCK(`FAT(/a1/a2/a3/pluto.txt)`)
- Scrive 0x0 sugli elementi `FAT(/a1/a2/a3/pluto.txt)`
- RELEASE(`FAT(/a1/a2/a3/pluto.txt)`)

Record Locking (2): Esempio `ufs_remove`

- Rimuove la directory entry relativa al file `pluto.txt` dalla directory `/a1/a2/a3`
- Aggiorna la dimensione della directory `/a1/a2/a3` nella directory entry di `/a1/a2`
- RELEASE(`FAT(/a1/a2/a3)`)
- RELEASE(`FAT(/a1/a2)`)