SRS - Simple Virtual File System

Russo Antonio

Introduzione

Scopo

Il Virtual File System (VFS) fornisce un'astrazione di un file system gerarchico, implementato interamente in memoria e sincronizzato con un file system reale tramite la politica di **write-through**. È stato progettato per offrire semplicità d'uso, modularità e per servire come base per l'estensione in contesto distribuito (tramite RMI).

Contesto

Il sistema simula un file system classico di tipo UNIX, ma indipendente dal kernel e realizzato interamente in Java. La struttura a nodi gerarchici permette di rappresentare directory, file e link simbolici. Il VFS è pensato per essere modulare, estendibile e indipendente dalla piattaforma, con alcune limitazioni (es. supporto ai symlink dipendente dall'OS).

- **Indipendente** da un vero file system a livello kernel
- **Semplicita'** delle API per le principali operazioni di
  - **Creazione**: mkdir, mknod, symlink
  - **Navigazione**: lookup, readdir, readlink
  - **Manipolazione**: read, write, rename, rmdir

Definizioni e acronimi

- VFS: Virtual File System
- Node: rappresentazione astratta di un elemento del file system
- Write-through: politica che sincronizza subito memoria e disco
- Symlink: collegamento simbolico a un altro nodo
- Hard link: collegamento multiplo che condivide lo stesso contenuto

Riferimenti

- Specifiche NFSv3/NFSv4 (RFC)
- Documentazione Java NIO
- Manuale Java RMI

Descrizione Generale

Funzionalita' del prodotto

Il sistema implementa le seguenti funzionalità, raggruppate per categoria.

Creazione

1. mkdir

   Creazione di una cartella.

   ```
   >> mkdir <path>/nomecartella
   ```

2. mknod

   Creazione di un file.

   ```
   >> mknod <path>/nomefile
   ```

3. symlink

   Creazione di un link simbolico, cioè un nodo che fa da puntatore a un altro "file" o "directory".

   ```
   >> symlink <target> <linkPath>
   ```

Navigazione

1. lookup

   Risoluzione di un path e restituzione del nodo corrispondente (seguendo eventuali symlink).

   ```
   >> lookup <path>
   ```

2. readdir

   Elenco dei contenuti di una directory.

   ```
   >> readdir <path>
   ```

3. readlink

   Restituisce il target di un link simbolico senza risolverlo.

   ```
   >> readlink <path>
   ```

Manipolazione

1. read

   Dato un path, la funzione restituisce i contenuti del file corrispondente.

   ```
   >> read <path>
   ```

2. write

   Scrive una sequenza di bytes su un file.

   ```
   >> write <path> <content>
   ```

   (a) Consistenza

   La consistenza durante l'operazione di scrittura è gestita tramite lock (ReentrantReadWriteLock).

   (b) Write-through

   Ogni modifica effettuata in memoria viene immediatamente riflessa sul filesystem reale montato come root.

3. rename

   Funzione di rinomina di un nodo (file o directory).

   ```
   >> rename <oldpath> <newpath>
   ```

4. rmdir

   Funzione per la cancellazione di directory *vuote.

   ```
   >> rmdir <path>
   ```

Gestione attributi

1. getattr

   Restituisce metadati (nome, tipo, timestamp).

   ```
   >> getattr <path>
   ```

2. setattr

   Permette di modificare attributi di un nodo (es. nome).

   ```
   >> setattr <path> <attributo> <valore>
   ```

Gestione apertura/chiusura

1. open

   Marca un file come aperto.

   ```
   >> open <path>
   ```

2. close

   Chiude un file precedentemente aperto.

   ```
   >> close <path>
   ```

Requisiti non funzionali

- **Consistenza**: lock per `path` con ReentrantReadWriteLock per prevenire race condition.
- **Portabilità**: indipendente dall'OS, con limitazioni sui symlink.
- **Sicurezza**: prevenzione path traversal al di fuori della root montata.
- **Estendibilità**: possibile aggiungere nuovi tipi di nodi (es. DeviceNode).

Architettura

Scelta dell'albero

Il VFS è implementato come una struttura ad albero con radice unica (/). Questo modello rispecchia i file system reali (UNIX-like) e facilita la navigazione e la gestione delle operazioni ricorsive.

Componenti principali

- Node (astratto): rappresenta un nodo generico con nome, timestamp, riferimento al padre.
- DirectoryNode: rappresenta directory, contiene mappa di figli.
- FileNode: rappresenta file, contiene byte[] data e hard link.
- SymlinkNode: rappresenta un link simbolico.
- FileSystem: gestisce operazioni sul VFS e write-through su disco reale.

Vantaggi dell'albero

- Modello naturale per rappresentare un FS.
- Navigazione semplice tramite traversal.
- Supporto naturale ad operazioni ricorsive (es. tree).
- Gestione chiara di symlink e hard link.

Protocolli interni

Flusso di un'operazione

1. Parsing del path in token.

2. Risoluzione dei nodi, seguendo eventuali symlink.
3. Acquisizione lock (read/write).
4. Esecuzione operazione in memoria.
5. Aggiornamento del disco reale (write-through).
6. Rilascio del lock.

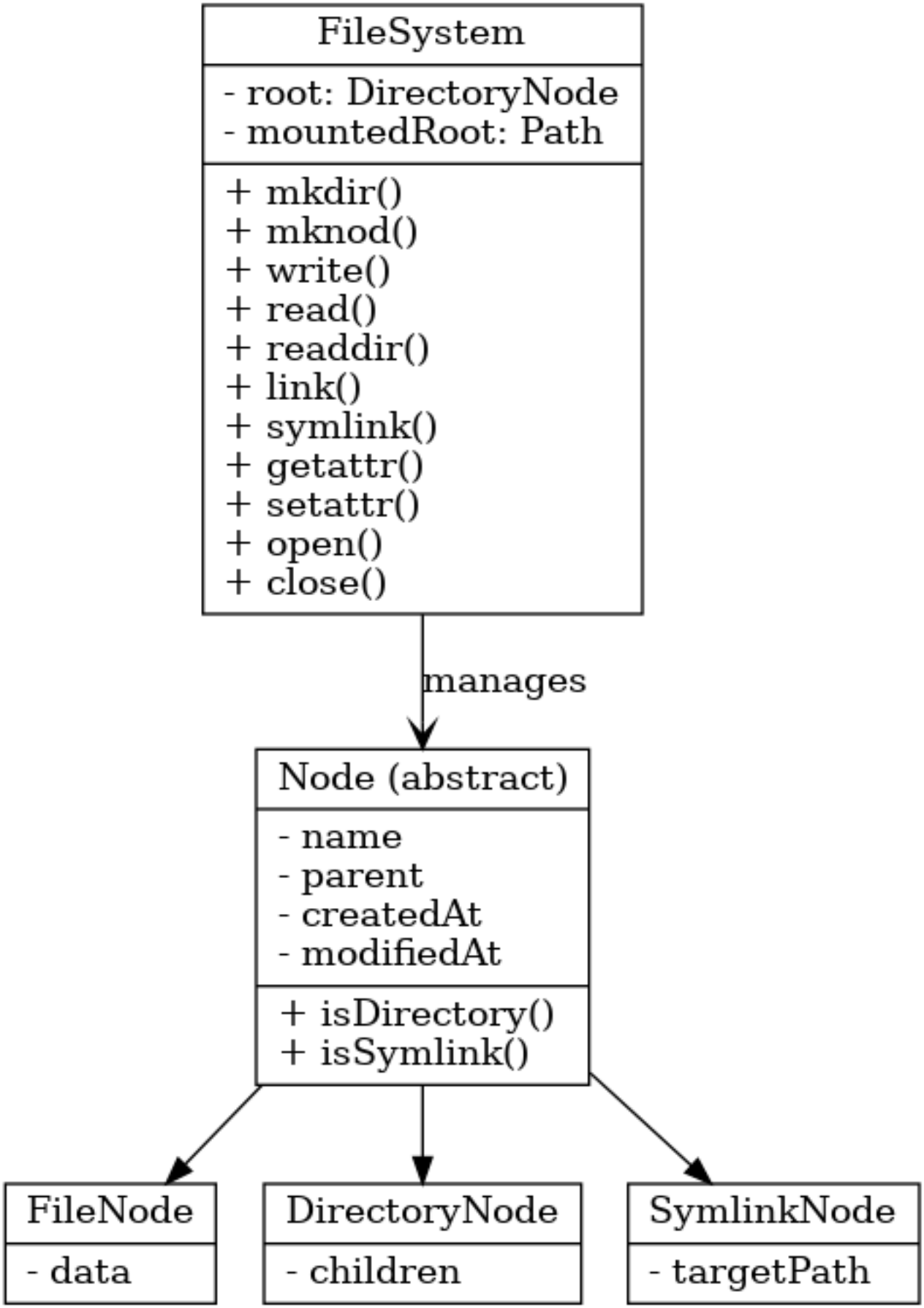
Sicurezza

Durante la risoluzione dei path viene controllato che non sia possibile uscire dalla root montata (protezione traversal).

Gestione errori

In caso di eccezioni IO durante write-through, l'operazione viene comunque mantenuta in memoria, evitando perdita di dati e garantendo consistenza logica.

UML


FileSystem
- root: DirectoryNode
- mountedRoot: Path
+ mkdir()
+ mknod()
+ write()
+ read()
+ readdir()
+ link()
+ symlink()
+ getattr()
+ setattr()
+ open()
+ close()
manages
Node (abstract)
- name
- parent
- createdAt
- modifiedAt
+ isDirectory()
+ isSymlink()
FileNode
- data
DirectoryNode
- children
SymlinkNode
- targetPath