Design data structures and algorithms for in-memory file system

Explain the data structures and algorithms that you would use to design an in-memory file system. Illustrate with an example in the code logic where possible.

Asked In: Amazon

A file system, in its most simplistic version, consists of Files and Directories. Each Directory contains a set of Files and Directories. Since Files and Directories share so many characteristics, we've implemented them such that they inherit from the same class, Entry.

Implemented Main logic in Java

```
// Entry is superclass for both File and Directory
public abstract class Entry
{
    protected Directory parent;
    protected long created;
    protected long lastUpdated;
    protected long lastAccessed;
    protected String name;

    public Entry(String n, Directory p)
    {
        name = n;
        parent = p;
        created= System.currentTimeMillis();
        lastUpdated = System.currentTimeMillis();
        lastAccessed = System.currentTimeMillis();
    }

    public boolean delete()
    {
        if (parent == null)
            return false;
        return parent.deleteEntry(this);
    }

    public abstract int size();

    /* Getters and setters. */
    public long getcreationTime()
    {
```

```
        return created;
    }
    public long getLastUpdatedTime()
    {
        return lastUpdated;
    }
    public long getLastAccessedTime()
    {
        return lastAccessed;
    }
    public void changeName(String n)
    {
        name = n;
    }
    public String getName()
    {
        return name;
    }
}

// A class to represent a File (Inherits
// from Entry)
public class File extends Entry
{
    private String content;
    private int size;

    public File(String n, Directory p, int sz)
    {
        super(n, p);
        size = sz;
    }
    public int size()
    {
        return size;
    }
    public String getContents()
    {
        return content;
    }
    public void setContents(String c)
    {
        content = c;
    }
}

// A class to represent a Directory (Inherits
// from Entry)
public class Directory extends Entry
{
    protected Arraylist<Entry> contents;

    public Directory(String n, Directory p)
    {
        super(n, p);
        contents = new Arraylist<Entry>();
    }
    public int size()
    {
        int size = 0;
        for (Entry e : contents)
            size += e.size();

        return size;
    }
    public int numberOfFiles()
    {
        int count = 0;
        for (Entry e : contents)
        {
```

```
            if (e instanceof Directory)
            {
                count++; // Directory counts as a file
                Directory d = (Directory) e;
                count += d. numberOfFiles ();
            }
            else if (e instanceof File)
                count++;
        }
        return count;
    }

    public boolean deleteEntry(Entry entry)
    {
        return contents.remove(entry);
    }

    public void addEntry(Entry entry)
    {
        contents.add(entry);
    }

    protected ArrayList<Entry> getContents()
    {
        return contents;
    }
}
```

Run on IDE

Alternatively, we could have implemented Directory such that it contains separate lists for files and subdirectories. This makes the nurnberOfFiles () method a bit cleaner, since it doesn't need to use the instanceof operator, but it does prohibit us from cleanly sorting files and directories by dates or names.

For data block allocation, we can use bitmask vector and linear search (see "Practical File System Design") or B+ trees (see Reference or Wikipedia).

References:

https://www.careercup.com/question?id=13618661

https://stackoverflow.com/questions/14126575/data-structures-used-to-build-file-systems

This article is contributed by **Mr. Somesh Awasthi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Design Pattern Amazon

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

How to design a parking lot using object-oriented principles?
How to prevent Singleton Pattern from Reflection, Serialization and Cloning?
Java Singleton Design Pattern Practices with Examples
Design an online hotel booking system like OYO Rooms
Design Patterns | Set 1 (Introduction)
Dependency Inversion Principle (SOLID)
Design Video Sharing System Like Youtube
Design Scalable System like Foursquare
Design Scalable System like Instagram
Mediator Design Pattern