Search Directories

For UNIX

Ashley Smith (ashley3@MIT.edu)

Jackson, 2PM

6.033 DP1: Proposal

February 28, 2014

Overview

This proposal outlines an implementation plan for search directories for UNIX. Search directories are special virtual directories that contain the results of a file search query as symbolic links to the original files.

UNIX is extremely inefficient for repeated searches

Currently, users can perform searches with the `find` command but the results of these searches cannot be saved for future use. Each search takes just as long as the last, so doing repeated searches for the same expression and search path is extremely inefficient.

I can create a command-line tool for search directories for UNIX

I will implement search directories as a UNIX command-line tool called `searchmount`. `searchmount` takes in an expression similar to that which the find command uses and creates a new virtual directory for the results of the search. These directories listen for changes and automatically update when new files are added to the original search path and when files in the original search path are updated to match the search. The results are stored in a database as a linked list with each entry in a directory having a pointer to the next entry.

Design Description

High-level Implementation

This section will describe the high-level implementation of storing and maintaining search results.

Storage of Search Directories

Search directories are stored in a persistent database on disk. In order to store the results of a search, we define a few data structures: `DirPointer` and `DirEntryPointer`. Figure 1 below shows these data structures and the variables and pointers they store.

As shown in Figure 1a, we have a simple lookup table, a database, which contains serialized pointers to the previously created search directories along with counts for the total number of directories and entries. When a `DirPointer` is read, it is cached in memory to ensure fast subsequent access to that search directory.

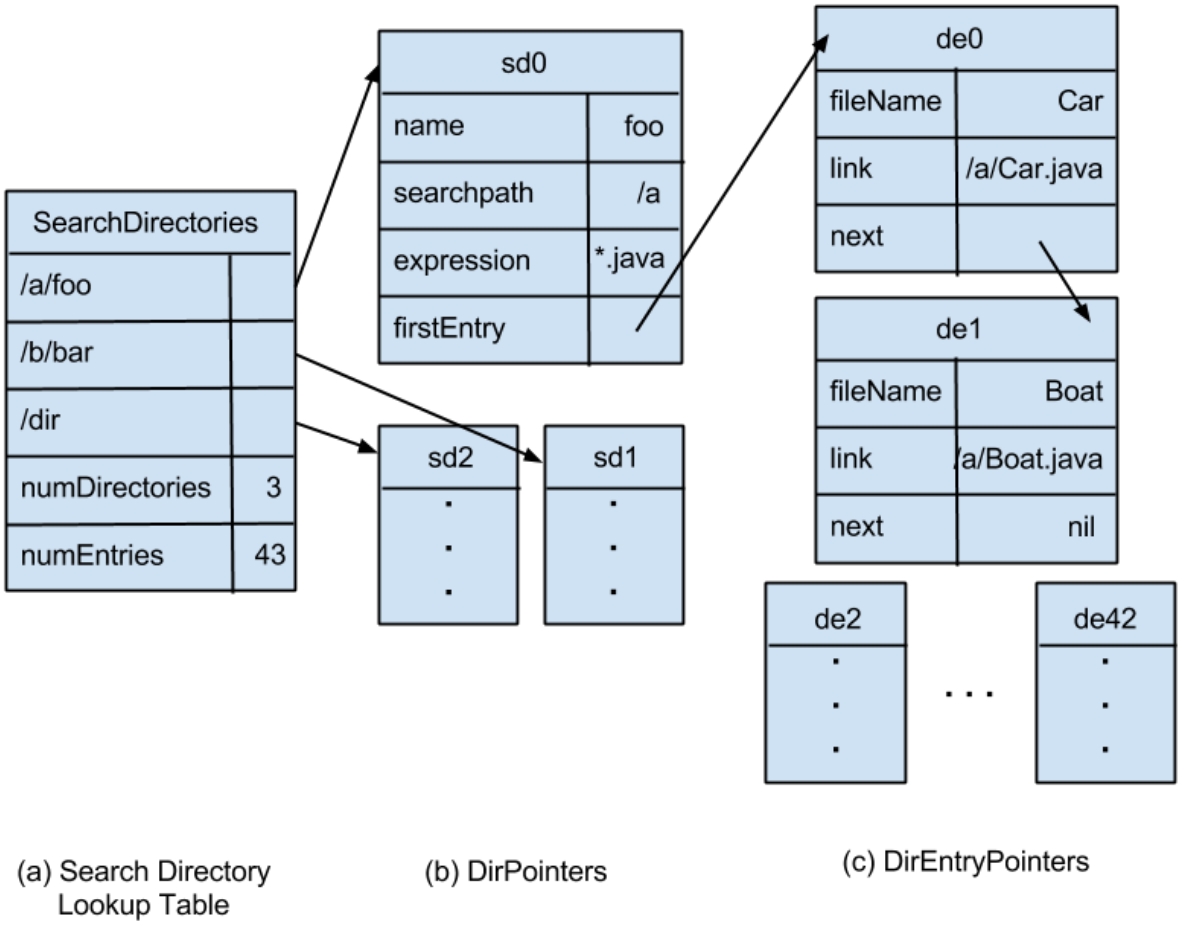

Figure 1: (a) The lookup table (database) provides us easy access to the search directories. (b) `DirPointers` contain important metadata about the search directory. (c) `DirEntryPointers` act as a linked list with each entry having a pointer to the next entry or to nil if it is the last entry in that search directory.

Maintenance of Search Directories

We utilize *inotify* to listen for changes to files in each directory in each `searchpath`. When we notice a change, we examine the directory, locate the added/updated file, and check if it now matches the expression. If it matches and is not already in the directory entry linked list, it will be added to the beginning. If it does not match and previously did, it will be removed from the directory entry linked list.

Application Programming Interface

Users create new directories using the `searchmount` command and access previously created directories through an application programming interface (API).

Creating new directories and removing old directories using searchmount

We provide the following two commands for users to create and remove search directories:

```
searchmount mountpoint searchpath [expression]
```

```
searchmount –u mountpoint
```

`searchmount` creates a new directory `mountpoint` and makes the new directory appear to contain symbolic links to files under `searchpath` matching the search query given in expression. The second command removes the `mountpoint` directory.

Internally, when `searchmount` is run, the following method will be called to create the new directory:

```
createDirectory(name, searchPath, expression)
```

> If the database does not already contain an entry with that information (`name, searchPath,` and `expression`), call the `find` command to get all files that should be virtually linked to. Set up the database to contain these results by initializing a `DirPointer` and many `DirEntryPointers`, linking them together, and putting a link to the `DirPointer` in the "SearchDirectories" database.

When the second command is run, we remove `mountpoint` from the "SearchDirectories" database. The `DirPointer` and `DirEntryPointers` will be automatically garbage collected.

Accessing previously created directories through the API

We provide the following API to access previously created directories:

```
DirEntryPointer getFirstDirectoryEntry(name)
```

> Get the `DirPointer` with key `name` from the "SearchDirectories" database and return the value of `firstEntry` from the `DirPointer`.

```
DirEntryPointer readNextDirectoryEntry(DirEntryPointer previous)
```

> Return the value of `next` from `previous`.

```
String fileName(DirEntryPointer entry)
```

> Return the value of `fileName` from `entry`.

```
String readSymbolicLink(DirEntryPointer entry)
```

> Return the value of `link` from `entry`.

Conclusion

This document proposes a design for search directories for UNIX that allows users to create directories that contain the results of a search and to be able to access this result in the future without having to repeat the search. We provide two UNIX commands for both advanced and novice users to create and remove these directories. Future tasks include implementation details for maintenance of search directories, improving efficiency for search folders with the same target path, and investigating using a virtual directory as a search target directory.

Word Count: 798