HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
BBM204 PROGRAMMING LAB.
ASSIGNMENT #3

Subject: Radix Trees
Submission Date: March 19,2015
Deadline: April 2,2015
Programming Language: ANSI C
Operation System: CentOS 7 (gcc 4.8.2)
Advisors: R.A. Aysun Koçak

1 Introduction

In this experiment, you are supposed to develop a program that will index given folder path by radix tree. The main purpose of the program is locating files according to given options which are explained below.

1.1 Background

A radix tree, Patricia trie /tree is a specialized set data structure based on the trie that is used to store a set of strings. A radix tree is an provides a flexible means of storing, indexing, and retrieving information in a large environment, which is economical of index space and of reindexing time.

In contrast with a regular trie, the edges or nodes of a Patricia trie can be labeled with sequences of characters rather than with single characters. These can be strings of characters, bit strings such as integers or IP addresses, or generally arbitrary sequences of objects in lexicographical order. Each node of radix tree is merged with its parent. An example can be seen in Fig.1.

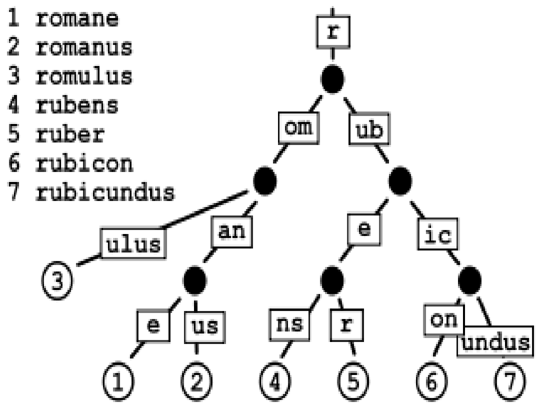

Figure 1: Radix Tree

2 Problem

As mentioned before, your aim is to index given directory path by radix tree. Your problem consists of two main steps; building radix tree and locating file(s)/folder(s) via radix tree. So, your first step is to build a radix tree with the file or folder names. You should get all files and folders under the given directory and use their names to build a radix tree like in Fig. 1. After you build the tree, your index tree should be saved as a file (file format is up to you). These steps will be executed after **updatedatabase** command. The formal definition of **updatedatabase** is given below:

NAME

updatedatabase update a database.

SYNOPSIS

updatedatabase [OPTION] PATH

DESCRIPTION

updatedatabase creates or updates a database used by locater. This command initiates building index tree process for given PATH. Then, the tree will be saved to DBPATH, if DB-PATH is given with -o option. If it is not given, the tree should be saved to a default location which is determined by implementer.

OPTIONS

The following options shall be supported:

–help

Output this help message.

-o DBPATH

Write the database to DBPATH instead of using the default database.

Shortly, you should build and save your index tree after **updatedatabase** command. Second part aims to locate file(s)/folder(s) according to given options. The formal definition of locater is given below:

NAME

locater find files by name

SYNOPSIS

locater [OPTION]... PATTERN

DESCRIPTION

locater reads database prepared by *updatedatabase* and writes file names matching PATTERN to standard output, one per line and **in an alphabetical order**. By default, *locater* does not check whether files found in database still exist. *locater* can never report files created after the most recent update of the relevant database. For this part, you should reconstruct the index tree and use it to locate files that has a name matching with PATTERN.

OPTIONS

The following options shall be supported:

–help
  Output this help message

-f, –file
  Print only files

-d, –directory
  Print only directories

-p [PERMISSION]
  Print files/folders with matching permission. PERMISSION will be in octal format (e.g. 755)

-i, –ignore-case
  Ignore case distinctions when matching patterns.

-db, –databasefile [DBPATH]
  Replace the default database with DBPATH. You should reconstruct the radix tree which is located in DBPATH (saved via updatedatabase).

3 Usage

Suppose that you have a file hierarchy as given below.

```
| –– <Example> (777)
|     | –– <BBM> (755)
|     |     | –– exp3.c (755)
|     |     | –– exp4.c (755)
|     |     | –– <data> (644)
|     |     |     | –– data1.txt (644)
|     |     |     | –– data2.txt (644)
|     | –– <bil> (777)
|     |     | –– report.tex (700)
|     |     | –– latex_info.tex (755)
|     | –– bam.dat (666)
|     | –– Exper.txt (777)
```

Here are some usage examples:

```
$ ./updatedatabase ./
```
Create index tree for current directory and save it a location that you determine.
```
$ ./updatedatabase -o ./Example/tree.db .
```
Create index tree for current directory and save it under ./Example folder.
```
$ ./locater -f -i ”exp*”
./BBM/exp3.c
./BBM/exp4.c
./Exper.txt
$ ./locater -f -i ”exp?”
```

```
./BBM/exp3.c
./BBM/exp4.c
$ ./locater -i -db ./Example/tree.db ”B*”
Load tree.db from ./Example location and find matching patterns:
./bam.dat
./BBM
./bil
$ ./locater -i -d -db ./Example/tree.db ”B*”
Load tree.db from ./Example location and find matching patterns:
./BBM
./bil
$ ./locater -i -d -p ”755”
./BBM
$ ./locater -i -f -p ”755”
./BBM/exp3.c
./BBM/exp4.c
./bil/latex_info.tex
$ ./locater -i -f -p ”755” ”exp*”
./BBM/exp3.c
./BBM/exp4.c
./bil/latex_info.tex
$ ./locater -i -f -p ”755” ”exp*”
./BBM/exp3.c
./BBM/exp4.c
```

Patterns

As mentioned above, your program will accept a pattern. This means given abstract string can match to a number of strings. You will handle two types of patterns in this experiment:

- Star (*) : any number of alpha-numeric character (0..∞)
- Question Mark (?) : One and only one alpha-numeric character

These marks can be used in various ways to express a pattern. These marks can occur at the beginning, middle or the end of a sentence. Here are some examples for star:
dat* will match with: ”dat”, ”data2”, ”daTagram” and so on, and will not match with DATA, Dat
B*L will match with: BIL, BASDFG-23L and so on, and will not match with BBM, B
*L will match with BIL, BBML, L and son on, and will not match with BBM, B

Here are some examples for Question Mark:
B?L will match with BIL, B1L and so on, and will not match with BL, BBML
?IL will match with BIL, 1IL and so on, and will not match with IL, BBIL
BI? will match with BIL, BI1 and so on, and will not match with BI, BILL

Hint

You will implement your program in C. You have to perform file and directory operations using UNIX system calls. You may want to use the following system functions:

open(), close(), lseek(), read(), fcntl(),write() stat(), fstat(), lstat(), unlink()utime(), chdir(), getcwd(), opendir(), readdir(), closedir(),rmdir()

Important Issues

- Test your program on ”dev.cs.hacettepe.edu.tr” before submission.
- Projects without a proper **Makefile** won't be graded.
- Your output has to be in alphabetical order(according to file/folder name, not full path).
- DO NOT use ”system” functions(except listed above).
- Do not use fopen() or fclose() functions; use open() and close() functions instead if needed.
- Use **dirent.h**, **stat.h** libraries and related libraries for file and directory operations
- And please pay attention items that are listed below for your report:
  1. Use report format in **ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/**
  2. Choose proper topics and give meaningful information by considering grammar and spelling rules
  3. Define the problems and explain your solutions (the report should contain your algorithm,too)
  4. Give information about radix tree(e.g. advantages, disadvantages)
  5. Give references that you used

Notes

Give necessary details in your report. Save all your work until the assignment is graded.You can ask your questions about the experiment on Piazza.

Your submission will be in the format below:

```
studentid.zip
—– <report>
—–—– report.pdf
—– <src>
—–—–*.c
—–—– *.h
—–—– Makefile
```

You will submit it to dersler.cs.hacettepe.edu.tr , BBM204 course. You have a right to do late submission. April 3, 2015 (evaluated over 90 points) and April 4, 2015 (evaluated over 80 points).

References

- http://en.wikipedia.org/wiki/Radix_tree
- Donald R. Morrison. 1968. PATRICIA – Practical Algorithm To Retrieve Information Coded in Alphanumeric. J. ACM 15, 4 (October 1968)
- http://linux.die.net/man/8/updatedb
- http://linux.die.net/man/1/locate