Lab I

C Refresher Module

December 29, 2016

1 FS Walker

Consider a Unix system where data is stored in directories. There is a root directory **(denoted by /)** which contains several other directories, each with a unique name. These directories may further contain directories and so on, thus forming a hierarchy. A directory can be uniquely identified by its name and its parent directory (the directory it is directly contained in). This is usually encoded in a path, which consists of several parts each preceded by a forward slash **('/')**. The final part is the name of the directory, and everything else gives the path of its parent directory.
Example: ***'/home/mfrw/linux'*** is the path to the directory ***'linux'***.

You are given a file ***entries.in*** as per the format:

- First set of lines consist of directory path in each line existing on a Unix computer.
- The second set of lines, as separated by an empty line from the first set, consists of a directory path in each line that you want to test for its existence. *Assume all the entries are absolute, i.e start with* ***'/'***

Your task is to:

- Build a directory tree from the first set of lines using an *appropriate* data structure where each node contains the name of the directory and has a pointer to the child directory nodes.
- Use the directory tree to check if the paths given in the second set of lines exist or not.
  Print ***Yes/No*** on the console, corresponding to each of the path to be tested.

Submission: Upload on backpack:

- A zipped[†] file including – a **makefile** to compile the program[‡], a simple one paragraph writeup (**readme**) of the approach used, the **source** code of the program & the ***entries.in*** file.

[†]We will run a plagiarism detector.
[‡]We expect a C program and no other language.

2 Example

An example ***entries.in*** file :

```
/a
/b
/c/d/g
/c/d
/a/e/f

/a/e
/a/g
/c
```

The *output* for this is :

```
YES : /a/e
NO  : /a/g
YES : /c
```

NOTE*: The program would be checked by redirecting the output on our test-cases to a file and then using* ***diff****, so please follow the output pattern.*

3 Hints

- The first node should always be '/'.
- Look at the man page of ***strtok*** to tokenize a string.
- Think simple, please don't over complicate stuff for yourself.
- Write clean, modular code.
- If you think this can't be done in lab. **Trust yourself, it can!**.

4 For those who want more

Any of these or all would make you eligible for a bonus:

- Try solving the same using a ***fancy*** data-structure. *(e.g B/B+ Tree etc)*
- Use a generic linked list implementation.
  See the linux kernel implementation of linked lists

5 I still want more

Google around and understand the subtle details in:

- **hlist** in linux kernel, found in *list.h*, google around.
  It uses double pointers in an intersting way.
- What is the Linux Torvalds' **double pointer** problem.