

Lab 10 – Word Hash Table

Assigned 11/12/09

Points: 20

Purpose

Create a random access file that uses a hash function to map words into it.

Lab Description

In the file, words.txt, is a list of words. The largest word is 10 characters long. Develop a hash function that will map each word to a random access file. The file should consist of 60,000 10 byte entries. Make sure that the word are uniformly distributed in the file and that collisions are handled correctly.

Support the following type of user interface *or something similar to it*:

```
Welcome the Nifty Word Lookup Program!
```

```
Enter a word to look up:
```

```
Word: dog
```

```
dog was found
```

```
Enter a word to look up:
```

```
Word: dogbones
```

```
dogbones was not found
```

```
Enter a word to look up:
```

```
Word: quit
```

```
Thanks for using this most nifty application!
```

It is suggested that you “zero” out the file and then add words to that position. In developing a hash function, test the function by creating a 60,000 element array and count the number of times each index is accessed. Then check to see how uniform your program distributes the numbers.

Turn in a copy of the program with sample output.

C Implementation Notes

To open a random access file in C use:

```
FILE *file = fopen("words.txt", "r");
```

The “r” means that the file will only be opened for read access. A “w+” means that it is open for read/write access.

The fseek function moves a cursor to that byte position within a file. When seeking remember that the *index* is the byte position in the file.

```
fseek(file2, index, SEEK_SET);
```

The read function returns *size* number of bytes and places it into the location specified by *word*. The first argument needs to be the address in memory. The third argument specifies how many blocks of *size* bytes to read.

```
fread(word, size, 1, filePointer);
```

The fwrite function write *size* byte to the file from the *word* location.

```
fwrite(word, size, 1, filePointer);
```

Java Implementation Notes

Random access is accomplished using instances of the RandomAccessFile class. A file can be opened for input and/or output.

```
public RandomAccessFile(String name, String mode) throws IOException;  
public RandomAccessFile(File file, String mode) throws IOException;
```

The mode may be either “r” or “rw”. Important methods include:

- **seek** – This method uses a single long argument to move to that position in the file.
- **getFilePoint** – This method returns a long value indicating the current position within the file.
- **length** – The **length** method returns the length of the file as a long number.
- **readFully** - This method will read the entire contents of the byte array specified as an argument

```
import java.io.*;

public class RandomFileDemo {
    public static void main(String args[]) throws IOException {
        int recordNumber;
        byte buffer[] = new byte[10];
        int i;

        RandomAccessFile rin = new RandomAccessFile(args[0], "rw");

        for (i=0; i<10; i++) {
            rin.writeBytes("String " + i);
        };

        recordNumber = Integer.parseInt(args[1]);

        rin.seek(recordNumber * 10);
        rin.readFully(buffer);
        System.out.println(new String(buffer));
        rin.close();
    }
}
```