/\*

\* lookup5 : local file ; setup a memory map of the file

\* Use bsearch. We assume that the words are already

\* in dictionary order in a file of fixed-size records

\* of type Dictrec

\* The name of the file is what is passed as resource

\*/

#include <stdlib.h>

#include <fcntl.h>

#include <string.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <unistd.h>

#include "dict.h"

/\*

\* This obscure looking function will be useful with bsearch

\* It compares a word with the word part of a Dictrec

\*/

int dict\_cmp(const void \*a,const void \*b) {

return strcmp((char \*)a,((Dictrec \*)b)->word);

}

int lookup(Dictrec \* sought, const char \* resource) {

static Dictrec \* table;

static int numrec;

Dictrec \* found, temp;

int i;

static int first\_time = 1;

if (first\_time) { /\* table ends up pointing at mmap of file \*/

int fd;

long filsiz;

first\_time = 0;

/\* Open the dictionary.

\* Fill in code. \*/

fd=open(resource,O\_RDONLY);

if(fd < 0){

perror("open");

}

/\* Get record count for building the tree. \*/

filsiz = lseek(fd,0L,SEEK\_END);

numrec = filsiz / sizeof(Dictrec);

/\* mmap the data.

\* Fill in code. \*/

table = mmap(NULL,numrec\*sizeof(Dictrec),PROT\_READ,MAP\_SHARED,fd,0);

if(table < 0){

perror("mmap");

exit(1);

}

close(fd);

}

/\* search table using bsearch

\* Fill in code. \*/

strcpy(temp.word,sought->word);

found = bsearch(&temp,table,numrec,sizeof(Dictrec),dict\_cmp);

if (found) {

strcpy(sought->text,found->text);

return FOUND;

}

return NOTFOUND;

}