NAME

lsearch, lfind – linear search and update

SYNOPSIS

#include <search.h>

void ∗**lsearch(const void** ∗*key*, **void** ∗*base*, **size_t** ∗*nelp*, **size_t** *width*,
int (∗*compar*) **(const void** ∗, **const void** ∗**));**

void ∗**lfind(const void** ∗*key*, **const void** ∗*base*, **size_t** ∗*nelp*, **size_t** *width*,
int (∗*compar*)**(const void** ∗, **const void** ∗**));**

MT-LEVEL

Safe

DESCRIPTION

lsearch( ) is a linear search routine generalized from Knuth (6.1) Algorithm S. (See *The Art of Computer Programming, Volume 3, Section 6.1, by Donald E. Knuth.*) It returns a pointer into a table indicating where a datum may be found. If the datum does not occur, it is added at the end of the table. *key* points to the datum to be sought in the table. *base* points to the first element in the table. *nelp* points to an integer containing the current number of elements in the table. The integer is incremented if the datum is added to the table. *width* is the size of an element in bytes. *compar* is a pointer to the comparison function that the user must supply (**strcmp**, for example). It is called with two arguments that point to the elements being compared. The function must return zero if the elements are equal and non-zero otherwise.

lfind( ) is the same as **lsearch( )** except that if the datum is not found, it is not added to the table. Instead, a null pointer is returned.

Note that:

- the pointers to the key and the element at the base of the table may be pointers to any type.
- The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.
- The value returned should be cast into type pointer-to-element.

EXAMPLES

This program will read in less than **TABSIZE** strings of length less than **ELSIZE** and store them in a table, eliminating duplicates, and then will print each entry.

```
#include <search.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#define TABSIZE 50
#define ELSIZE 120

main( )
{
        char line[ELSIZE];      /* buffer to hold input string */
        char tab[TABSIZE][ELSIZE];   /* table of strings */
```

```
        size_t nel = 0;          /* number of entries in tab */
        int i;

        while (fgets(line, ELSIZE, stdin) != NULL &&
                nel < TABSIZE)
                (void) lsearch(line, tab, &nel, ELSIZE, mycmp);
        for( i = 0; i < nel; i++ )
                (void)fputs(tab[i], stdout);
        return 0;
}
```

SEE ALSO

bsearch(3C), **hsearch**(3C), **string**(3C), **tsearch**(3C)

The Art of Computer Programming, Volume 3, Sorting and Searching by Donald E. Knuth, published by Addison-Wesley Publishing Company, 1973.

NOTES

If the searched-for datum is found, both **lsearch( )** and **lfind( )** return a pointer to it. Otherwise, **lfind( )** returns **NULL** and **lsearch( )** returns a pointer to the newly added element.

Undefined results can occur if there is not enough room in the table to add a new item.