NAME

bsearch - binary search of a sorted table

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <stdlib.h>

void *
bsearch(const void *key, const void *base, size_t nmemb, size_t size,
    int (*compar) (const void *, const void *));
```

DESCRIPTION

The **bsearch**() function searches an array of *nmemb* objects, the initial member of which is pointed to by *base*, for a member that matches the object pointed to by *key*. The size of each member of the array is specified by *size*.

The contents of the array should be in ascending sorted order according to the comparison function referenced by *compar*. The *compar* routine is expected to have two arguments which point to the *key* object and to an array member, in that order, and should return an integer less than, equal to, or greater than zero if the *key* object is found, respectively, to be less than, to match, or be greater than the array member. See the *int_compare* sample function in qsort(3) for a comparison function that is also compatible with **bsearch**().

RETURN VALUES

The **bsearch**() function returns a pointer to a matching member of the array, or a null pointer if no match is found. If two members compare as equal, which member is matched is unspecified.

EXAMPLES

A sample program that searches people by age in a sorted array:

```
#include <assert.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct person {
        const char              *name;
```

```
        int                     age;
};

static int
compare(const void *a, const void *b)
{
        const int *age;
        const struct person *person;

        age = a;
        person = b;

        return (*age - person->age);
}

int
main(void)
{
        struct person *friend;
        int age;
        /* Sorted array */
        const struct person friends[] = {
                { "paul", 22 },
                { "anne", 25 },
                { "fred", 25 },
                { "mary", 27 },
                { "mark", 35 },
                { "bill", 50 }
        };
        const size_t len = sizeof(friends) / sizeof(friends[0]);

        age = 22;
        friend = bsearch(&age, friends, len, sizeof(friends[0]), compare);
        assert(strcmp(friend->name, "paul") == 0);
        printf("name: %s\nage: %d\n", friend->name, friend->age);

        age = 25;
        friend = bsearch(&age, friends, len, sizeof(friends[0]), compare);

        /*
```

```
         * For multiple elements with the same key, it is implementation
         * defined which will be returned
         */
        assert(strcmp(friend->name, "fred") == 0 ||
           strcmp(friend->name, "anne") == 0);
        printf("name: %s\nage: %d\n", friend->name, friend->age);

        age = 30;
        friend = bsearch(&age, friends, len, sizeof(friends[0]), compare);
        assert(friend == NULL);
        printf("friend aged 30 not found\n");
}
```

SEE ALSO

db(3), lsearch(3), qsort(3)

STANDARDS

The **bsearch**() function conforms to ISO/IEC 9899:1990 ("ISO C90").