NAME

heapsort, **heapsort_b**, **mergesort**, **mergesort_b**, **qsort**, **qsort_b**, **qsort_r** — sort functions

SYNOPSIS

```
#include <stdlib.h>

int
heapsort(void *base, size_t nel, size_t width,
    int (*compar)(const void *, const void *));

int
heapsort_b(void *base, size_t nel, size_t width,
    int (^compar)(const void *, const void *));

int
mergesort(void *base, size_t nel, size_t width,
    int (*compar)(const void *, const void *));

int
mergesort_b(void *base, size_t nel, size_t width,
    int (^compar)(const void *, const void *));

void
qsort(void *base, size_t nel, size_t width,
    int (*compar)(const void *, const void *));

void
qsort_b(void *base, size_t nel, size_t width,
    int (^compar)(const void *, const void *));

void
qsort_r(void *base, size_t nel, size_t width, void *thunk,
    int (*compar)(void *, const void *, const void *));
```

所在的库

所在的库

1.数组
2.数组个人
3.元素的字节数
4.排序函数。

DESCRIPTION

The **qsort**() function is a modified partition-exchange sort, or quicksort. The **heapsort**() function is a modified selection sort. The **mergesort**() function is a modified merge sort with exponential search, intended for sorting data with pre-existing order.

The **qsort**() and **heapsort**() functions sort an array of *nel* objects, the initial member of which is pointed to by *base*. The size of each object is specified by *width*. The **mergesort**() function behaves similarly, but *requires* that *width* be greater than or equal to "sizeof(void *) / 2".

The contents of the array *base* are sorted in ascending order according to a comparison function pointed to by *compar*, which requires two arguments pointing to the objects being compared.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

The **qsort_r**() function behaves identically to **qsort**(), except that it takes an additional argument, *thunk*, which is passed unchanged as the first argument to function pointed to *compar*. This allows the comparison function to access additional data without using global variables, and thus **qsort_r**() is suitable for use in functions which must be reentrant.

The algorithms implemented by **qsort**(), **qsort_r**(), and **heapsort**() are *not* stable; that is, if two members compare as equal, their order in the sorted array is undefined. The **mergesort**() algorithm is stable.

The **qsort**() and **qsort_r**() functions are an implementation of C.A.R. Hoare's "quicksort" algorithm, a variant of partition-exchange sorting; in particular, see D.E. Knuth's *Algorithm Q*. **Quicksort** takes O N lg N average time. This implementation uses median selection to avoid its O N**2 worst-case behavior.

The **heapsort**() function is an implementation of J.W.J. William's "heapsort" algorithm, a variant of selection sorting; in particular, see D.E. Knuth's *Algorithm H*. **Heapsort** takes O N lg N worst-case time. Its *only* advantage over **qsort**() is that it uses almost no additional memory; while **qsort**() does not allocate memory, it is implemented using recursion.

The function **mergesort**() requires additional memory of size *nel* * *width* bytes; it should be used only when space is not at a premium. The **mergesort**() function is optimized for data with pre-existing order; its worst case time is O N lg N; its best case is O N.

Normally, **qsort**() is faster than **mergesort**() which is faster than **heapsort**(). Memory availability and pre-existing order in the data can make this untrue.

The **heapsort_b**(), **mergesort_b**(), and **qsort_b**() routines are like the corresponding routines without the _b suffix, expect that the *compar* callback is a block pointer instead of a function pointer.

RETURN VALUES

The **qsort**(), **qsort_b**() and **qsort_r**() functions return no value.

The **heapsort**(), **heapsort_b**(), **mergesort**(), and **mergesort_b**() functions return the value 0 if successful; otherwise the value −1 is returned and the global variable *errno* is set to indicate the error.

COMPATIBILITY

Previous versions of **qsort**() did not permit the comparison routine itself to call **qsort**(*3*). This is no longer true.

ERRORS

The **heapsort**(), **heapsort_b**(), **mergesort**(), and **mergesort_b**() functions succeed unless:

[EINVAL] The *width* argument is zero, or, the *width* argument to **mergesort**() or **mergesort_b**() is less than "sizeof(void *) / 2".

[ENOMEM] The **heapsort**(), **heapsort_b**(), **mergesort**(), or **mergesort_b**() functions were unable to allocate memory.

SEE ALSO

sort(1), radixsort(3)

Hoare, C.A.R., "Quicksort", *The Computer Journal*, 5:1, pp. 10-15, 1962.

Williams, J.W.J, "Heapsort", *Communications of the ACM*, 7:1, pp. 347-348, 1964.

Knuth, D.E., "Sorting and Searching", *The Art of Computer Programming*, Vol. 3, pp. 114-123, 145-149, 1968.

McIlroy, P.M., "Optimistic Sorting and Information Theoretic Complexity", *Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1992.

Bentley, J.L. and McIlroy, M.D., "Engineering a Sort Function", *Software--Practice and Experience*, Vol. 23(11), pp. 1249-1265, November 1993.

STANDARDS

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