NAME

qsort – quick sort

SYNOPSIS

#include <stdlib.h>

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

MT-LEVEL

Safe

DESCRIPTION

qsort( ) is an implementation of the quick-sort algorithm. It sorts a table of data in place. The contents of the table are sorted in ascending order according to the user-supplied comparison function.

base points to the element at the base of the table. *nel* is the number of elements in the table. *width* specifies the size of each element in bytes. *compar* is the name of the comparison function, which is called with two arguments that point to the elements being compared. The function must return an integer less than, equal to, or greater than zero to indicate if the first argument is to be considered less than, equal to, or greater than the second argument.

The contents of the table are sorted in ascending order according to the user supplied comparison function.

EXAMPLES

The following program sorts a simple array:

```
static  int intcompare(int *i, int *j)
{
      if (*i > *j)
            return (1);
      if (*i < *j)
            return (–1);
      return (0);
}

main( )
{
      int a[10];
      int i;

      a[0] = 9;
      a[1] = 8;
      a[2] = 7;
      a[3] = 6;
      a[4] = 5;
      a[5] = 4;
      a[6] = 3;
      a[7] = 2;
      a[8] = 1;
      a[9] = 0;
```

```
        qsort((char *) a, 10, sizeof(int), intcompare);

        for (i=0; i<10; i++) printf(" %d",a[i]);
        printf("\n");
}
```

SEE ALSO

sort(1), **bsearch**(3C), **lsearch**(3C), **string**(3C)

NOTES

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 relative order in the output of two items that compare as equal is unpredictable.