Q

qsort() — General utility (libc)

Sort an array

void qsort(void **array*, **size_t** *number*, **size_t** *size*, **int (****comparison*)
(const void **arg1*, **const void** **arg2*));

qsort sorts the elements within an array. *array* points to the base of the array being sorted; it has *number* members, each of which is *size* bytes long. In practice, *array* is usually an array of pointers and *size* is the **sizeof** the object to which each points.

comparison points to the function that compares two members of *array*. *arg1* and *arg2* each point to a member within *array*. The comparison routine must return a negative number, zero, or a positive number, depending upon whether *arg1* is, respectively, less than, equal to, or greater than *arg2*. If two or more members of *array* are identical, their ordering within the sorted array is unspecified.

Example

This example prints the command-line arguments in alphabetical order.

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

int
compar(char *cp1[], char *cp2[])
{
      return(strcmp(*cp1, *cp2));
}

main(int argc, char *argv[])
{
      qsort((void *)++argv, (size_t)--argc, sizeof(*argv), compar);

      while(argc--)
            printf("%s ", *argv++);
      return(EXIT_SUCCESS);
}
```

Cross-references

Standard, §4.10.5.2
The C Programming Language, ed. 2, p. 87
The Art of Computer Programming, vol. 3

See Also

bsearch, general utilities

Notes

The name "qsort" reflects the fact that most implementations of this function (including **Let's C**) use C. A. R. Hoare's "quicksort" algorithm. This algorithm is recursive and makes heavy use of the stack. It is also specified by the Association for Computing Machinery's algorithm 271.

Quicksort works on the basis of partitioning its input, and is highly dependent on the first element that starts the partitioning process. Given appropriate data, it can have a worst-case performance of *O(n^2)*.