blob: cd6696136101a6f562942aea73f807af44f698f5 [file] [log] [blame]

```
/*      $OpenBSD: qsort.c,v 1.10 2005/08/08 08:05:37 espie Exp $ */
/*-
 * Copyright (c) 1992, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <sys/types.h>
#include <stdlib.h>

static __inline char    *med3(char *, char *, char *, int (*)(const void *, const void *));
static __inline void     swapfunc(char *, char *, int, int);

#define min(a, b)       (a) < (b) ? a : b

/*
 * Qsort routine from Bentley & McIlroy's "Engineering a Sort Function".
 */
#define swapcode(TYPE, parmi, parmj, n) {               \
```

```
43          long i = (n) / sizeof (TYPE);                   \
44          TYPE *pi = (TYPE *) (parmi);                    \
45          TYPE *pj = (TYPE *) (parmj);                    \
46          do {                                            \
47                  TYPE    t = *pi;                        \
48                  *pi++ = *pj;                            \
49                  *pj++ = t;                              \
50          } while (--i > 0);                              \
51  }
52
53  #define SWAPINIT(a, es) swaptype = ((char *)a - (char *)0) % sizeof(long) || \
54          es % sizeof(long) ? 2 : es == sizeof(long)? 0 : 1;
55
56  static __inline void
57  swapfunc(char *a, char *b, int n, int swaptype)
58  {
59          if (swaptype <= 1)
60                  swapcode(long, a, b, n)
61          else
62                  swapcode(char, a, b, n)
63  }
64
65  #define swap(a, b)                                      \
66          if (swaptype == 0) {                            \
67                  long t = *(long *)(a);                  \
68                  *(long *)(a) = *(long *)(b);            \
69                  *(long *)(b) = t;                       \
70          } else                                          \
71                  swapfunc(a, b, es, swaptype)
72
73  #define vecswap(a, b, n)        if ((n) > 0) swapfunc(a, b, n, swaptype)
74
75  static __inline char *
76  med3(char *a, char *b, char *c, int (*cmp)(const void *, const void *))
77  {
78          return cmp(a, b) < 0 ?
79                 (cmp(b, c) < 0 ? b : (cmp(a, c) < 0 ? c : a ))
80                :(cmp(b, c) > 0 ? b : (cmp(a, c) < 0 ? a : c ));
81  }
82
83  void
84  qsort(void *aa, size_t n, size_t es, int (*cmp)(const void *, const void *))
85  {
86          char *pa, *pb, *pc, *pd, *pl, *pm, *pn;
87          int d, r, swaptype, swap_cnt;
88          char *a = aa;
89
90  loop:   SWAPINIT(a, es);
91          swap_cnt = 0;
92          if (n < 7) {
```

```
93                  for (pm = (char *)a + es; pm < (char *) a + n * es; pm += es)
94                          for (pl = pm; pl > (char *) a && cmp(pl - es, pl) > 0;
95                               pl -= es)
96                                  swap(pl, pl - es);
97                  return;
98          }
99          pm = (char *)a + (n / 2) * es;
100         if (n > 7) {
101                 pl = (char *)a;
102                 pn = (char *)a + (n - 1) * es;
103                 if (n > 40) {
104                         d = (n / 8) * es;
105                         pl = med3(pl, pl + d, pl + 2 * d, cmp);
106                         pm = med3(pm - d, pm, pm + d, cmp);
107                         pn = med3(pn - 2 * d, pn - d, pn, cmp);
108                 }
109                 pm = med3(pl, pm, pn, cmp);
110         }
111         swap(a, pm);
112         pa = pb = (char *)a + es;
113
114         pc = pd = (char *)a + (n - 1) * es;
115         for (;;) {
116                 while (pb <= pc && (r = cmp(pb, a)) <= 0) {
117                         if (r == 0) {
118                                 swap_cnt = 1;
119                                 swap(pa, pb);
120                                 pa += es;
121 }
122                         pb += es;
123                 }
124                 while (pb <= pc && (r = cmp(pc, a)) >= 0) {
125                         if (r == 0) {
126                                 swap_cnt = 1;
127                                 swap(pc, pd);
128                                 pd -= es;
129                         }
130                         pc -= es;
131                 }
132                 if (pb > pc)
133                         break;
134                 swap(pb, pc);
135                 swap_cnt = 1;
136                 pb += es;
137                 pc -= es;
138         }
139         if (swap_cnt == 0) {  /* Switch to insertion sort */
140                 for (pm = (char *) a + es; pm < (char *) a + n * es; pm += es)
141                         for (pl = pm; pl > (char *) a && cmp(pl - es, pl) > 0;
142                              pl -= es)
```

```
143                                 swap(pl, pl - es);
144                 return;
145         }
146
147         pn = (char *)a + n * es;
148         r = min(pa - (char *)a, pb - pa);
149         vecswap(a, pb - r, r);
150         r = min(pd - pc, pn - pd - (int)es);
151         vecswap(pb, pn - r, r);
152         if ((r = pb - pa) > (int)es)
153                 qsort(a, r / es, es, cmp);
154         if ((r = pd - pc) > (int)es) {
155                 /* Iterate rather than recurse to save stack space */
156                 a = pn - r;
157                 n = r / es;
158                 goto loop;
159         }
160 /*              qsort(pn - r, r / es, es, cmp);*/
161 }
```