```
1   /* dsread.c - dsread */
2
3   #include <conf.h>
4   #include <kernel.h>
5   #include <proc.h>
6   #include <disk.h>
7
8   /*------------------------------------------------------------------------
9    *  dsread  --  read a block from a disk device
10   *------------------------------------------------------------------------
11   */
12  dsread(devptr, buff, block)
13        struct      devsw *devptr;
14        char  *buff;
15        DBADDR      block;
16  {
17        struct      dreq  *drptr;
18        int   stat;
19        char  ps;
20
21        disable(ps);
22        drptr = (struct dreq *) getbuf(dskrbp);
23        drptr->drdba = block;
24        drptr->drpid = currpid;
25        drptr->drbuff = buff;
26        drptr->drop = DREAD;
27        if ( (stat=dskenq(drptr, devptr->dvioblk)) == DONQ) {
28              suspend(currpid);
29              stat = drptr->drstat;
30        }
31        freebuf(drptr);
32        restore(ps);
33        return(stat);
34  }
35
```

```
1   /* dswrite.c - dswrite */
2
3   #include <conf.h>
4   #include <kernel.h>
5   #include <proc.h>
6   #include <disk.h>
7
8   /*------------------------------------------------------------------------
9    *  dswrite  --  write a block (system buffer) onto a disk device
10   *------------------------------------------------------------------------
11   */
12  dswrite(devptr, buff, block)
13        struct      devsw *devptr;
14        char  *buff;
15        DBADDR      block;
16  {
17        struct      dreq  *drptr;
18        char  ps;
19
20        disable(ps);
21        drptr = (struct dreq *) getbuf(dskrbp);
22        drptr->drbuff = buff;
23        drptr->drdba = block;
24        drptr->drpid = currpid;
25        drptr->drop = DWRITE;
26        dskenq(drptr, devptr->dvioblk);
27        restore(ps);
28        return(OK);
29  }
30
```

```
1   /* dskenq.c - dskenq */
2
3   #include <conf.h>
4   #include <kernel.h>
5   #include <disk.h>
6
7   /*------------------------------------------------------------------------
8    *  dskenq  --  enqueue a disk request and start I/O if disk not busy
9    *------------------------------------------------------------------------
10   */
11  dskenq(drptr, dsptr)
12        struct     dreq  *drptr;
13        struct     dsblk *dsptr;
14  {
15        struct     dreq  *p, *q;          /* q follows p through requests   */
16        DBADDR     block;
17        int   st;
18
19        if ( (q=dsptr->dreqlst) == DRNULL ) {
20              dsptr->dreqlst = drptr;
21              drptr->drnext = DRNULL;
22              dskstrt(dsptr);
23              return(DONQ);
24        }
25        block = drptr->drdba;
26        for (p = q->drnext ; p != DRNULL ; q=p,p=p->drnext) {
27              if (p->drdba==block && (st=dskqopt(p, q, drptr)!=SYSERR))
28                          return(st);
29              if ( (q->drdba <= block && block < p->drdba) ||
30                   (q->drdba >= block && block > p->drdba)   ) {
31                    drptr->drnext = p;
32                    q->drnext = drptr;
33                    return(DONQ);
34              }
35        }
36        drptr->drnext = DRNULL;
37        q->drnext = drptr;
38        return(DONQ);
39  }
40
```

```
1   /* dskqopt.c - dskqopt */
2   #include <conf.h>
3   #include <kernel.h>
4   #include <disk.h>
5   /*------------------------------------------------------------------------
6    *  dskqopt  --  optimize requests to read/write/seek to the same block
7    *------------------------------------------------------------------------
8    */
9   dskqopt(p, q, drptr)
10  Continued on the next page
```

```
1   dskqopt(p, q, drptr)
2   struct      dreq  *p, *q, *drptr;
3
4   {
5         char  *to, *from;
6         int   i;
7         DBADDR      block;
8
9         /* By definition, sync requests cannot be optimized.  Also,    */
10        /* cannot optimize read requests if already reading.           */
11        if (drptr->drop==DSYNC || (drptr->drop==DREAD && p->drop==DREAD))
12              return(SYSERR);
13        if (drptr->drop == DSEEK) {  /* ignore extraneous seeks   */
14              freebuf(drptr);
15              return(OK);
16        }
17        if (p->drop == DSEEK) {           /* replace existing seeks    */
18              drptr->drnext = p->drnext;
19              q->drnext = drptr;
20              freebuf(p);
21              return(OK);
22        }
23        if (p->drop==DWRITE && drptr->drop==DWRITE) { /* dup write     */
24              drptr->drnext = p->drnext;
25              q->drnext = drptr;
26              freebuf(p->drbuff);
27              freebuf(p);
28              return(OK);
29        }
30        if (drptr->drop==DREAD && p->drop==DWRITE) {  /* satisfy read  */
31              to = drptr->drbuff;
32              from = p->drbuff;
33              for (i=0 ; i<DBUFSIZ ; i++)*to++ = *from++;
34              return(OK);
35        }
36        if (drptr->drop==DWRITE && p->drop==DREAD) {  /* sat. old read*/
37              block = drptr->drdba;
38              from = drptr->drbuff;
39              for (; p!=DRNULL && p->drdba==block ; p=p->drnext) {
40                    q->drnext = p->drnext;
41                    to = p->drbuff;
42                    for (i=0 ; i<DBUFSIZ ; i++)
43                          *to++ = *from++;
44                    p->drstat = OK;
45                    ready(p->drpid, RESCHNO);
46              }
47              drptr->drnext = p;
48              q->drnext = drptr;
49              resched();
50              return(OK);
51        }
52        return(SYSERR);
53  }
54
```

```
* dskstrt.c - dskstrt */
#include <conf.h>
#include <kernel.h>
#include <disk.h>
/*------------------------------------------------------------------------
 *  dskstrt  --  start an I/O operation on a disk device
 *------------------------------------------------------------------------
 */

dskstrt(dsptr)
      struct     dsblk *dsptr;
{
      struct     xbdcb *xptr;
      struct     dtc   *dtptr;
      struct     dreq  *drptr;

      /* build command for controller */
      drptr = dsptr->dreqlst;
      xptr = & dsptr->ddcb;
      xptr->xop    = (char) drptr->drop;      /* opcode   */
      xptr->xunit  = (char) 0;                /* top addr bits*/
      xptr->xmaddr = (char) ((drptr->drdba>>8)&0377);    /* mid addr bits*/
      xptr->xladdr = (char) (drptr->drdba & 0377);  /* low addr bits*/
      xptr->xcount = (char) 1;                /* num of blocks*/
      xptr->xcntl  = (char) XRETRY;                /* retry code    */

      /* feed command to controller through interface */
      dtptr = dsptr->dcsr;
      dtptr->dt_dar = drptr->drbuff;
      dtptr->dt_car = xptr;
      dtptr->dt_xdar = dtptr->dt_xcar = 0;
      dtptr->dt_csr = DTINTR | DTGO;

}
```