RIJKSUNIVERSITEIT TE GRONINGEN
MATHEMATISCH INSTITUUT

GUTS

listings

Wim Bronsvoort

TW-200 deel III

This document contains the following listings:

kernel.mac (the kernel)

clock.i - clock handler

disk.i - disk driver

core.i - core manager

filsys.i - file system

conhan.i - console handler

schedu.i - scheduler

syser.i - external system routines

init.i - initialization

read symbol - read symbol routine in subsystem

loader - loader in subsystem

KERNEL RT-11 MACRO VM02-12 1-DEC-78
TABLE OF CONTENTS

```
                        .ENABL LC
                        .title kernel
        000000          .asect
                        .mcall  .regdef
000000                  .regdef
                
        000040          .=40                    ; set fixed
000040  000137          jmp     @#start         ; start address
        010024
000044  000137          jmp     @#dump          ; dump address
        011254
                
                        .sbttl  globals
                
                        ; interrupt and trap routines
        000440          .=440
                        .globl  errorep         ; error traps
00440   005330          errorep:error
                        .globl  iotep           ; iot
00442   005420          iotep:  iotser
                        .globl  powep           ; power failure
00444   005430          powep:  powint
                        .globl  emtep           ; emt
00446   005434          emtep:  emtser
                        .globl  clockep         ; clock interrupt
00450   005506          clockep:kwpint
                        .globl  diskep          ; disk interrupt
00452   005552          diskep: diskint
                        .globl  cinep           ; console input interrupt
00454   010566          cinep:  consin
                        .globl  coutep          ; console output interrupt
00456   010644          coutep: consout
                
                        ; data entry points
                        .globl  proctab         ; process table
                        .globl  readyq          ; ready queue
                        .globl  begrdy,endrdy   ; + pointers
                        .globl  process         ; current process
                        .globl  apd             ; pointer process table
                        .globl  runproc         ; current user process
                        .globl  suppar,suppdr   ; copy supervisor active page registers
                        .globl  userap          ; copies users active page registers
                        .globl  count           ; clock count
                        .globl  serv            ; service exchange table
                
                        .globl asleep,find ; temporary to recov
                
                        ; data reference
                        .globl  condes          ; console descriptors
                
                        ; reference for initialization
                        .globl  init,initstk
                
                        ; entry point for exit (called after initialization)
                        .globl  kernex
00460   000462          kernex: toexit
00462   012706          toexit: mov     #kstk-4,sp      ; set kernel stack
```

GLOBALS

```
                000514
56 00466 000167                 jmp     exit            ; exit kernel
                006510
57
```

```
                        .sbttl  constants
                
000022                  recd=22         ; entries in proctab
000025                  heldup=25
000030                  rsmoni=30
000032                  asleep=32
                
000000                  closer=0        ; clock service
000001                  dsem=1          ; disk semaphore
000030                  schedu=30       ; scheduler process
000030                  highsup=30      ; highest supervisor process
000031                  frstus=31       ; first user process
000055                  usminup=55      ; lowest user service number minus
                                        ; lowest user process number
000106                  luserv=70.      ; lowest user service number
000047                  idle=47         ; idle process
000074                  recov=60.       ; recovery service
                
000047                  maxproc=47      ; maximum contants
000037                  maxsem=37
000007                  maxup=7
000037                  maxmess=37
000077                  maxfree=77
                
177740                  semmask=177740  ; masks
177600                  servmk=177600
177740                  rdymask=177740
                
177640                  stpar=177640    ; registers
177600                  stpdr=177600
172354                  kapr6=172354
177776                  psw=177776
                
000000                  false=0
000001                  true=1
000001                  on=1            ; used for queues
000000                  off=0
140000                  userps=140000   ; user psw
000020                  pdr=20          ; offset pdr registers in userapr
000520                  kstk=520        ; kernel stack position
000377                  notasl=377      ; indicates process not asleep
000015                  cr=15
000012                  lf=12
```

DATA AREA

```
1                       .sbttl  data area
2
3                       ; kernel stack between 500 and 520
4       000520          .=kstk
5
6 000520 000047 process:.word   maxproc         ; current process
7 000522 000000 apd:    .word   0               ; address current process in process table
8 000524 000047 runproc:.word   idle            ; current user process
9 000526 000000 counter:.word   0               ; semaphores
10 00530 000000 waiting:.word   0
11      000726          .=4*maxsem+.
12 00726 003326 proctab:.=40*maxproc+40+.       ; process table
13 03326 003366 readyq: .=40+.                  ; ready queue
14 03366 000000 begrdy: .word   0               ; begin pointer ready queue
15 03370 000000 endrdy: .word   0               ; end pointer ready queue
16 03372 003572 serv:   .=.+200                 ; service exchange table
17 03572 000000 recproc:.word   0               ; process which is receiving message
18 03574 003614 suppar: .=20+.                  ; supervisor page address registers
19 03614 003634 suppdr: .=20+.                  ; supervisor page descriptor registers
20 03634 004174 userapr:.=40*maxup+.            ; user active page registers
21 04174 004774 messtab:.=14*maxmess+14+.       ; message buffers
22 04774 000000 free:   .word   0               ; queue of free buffers
23 04776    000 link:   .byte   0               ; free space all queues
24 04777    000 info:   .byte   0
25      005176          .=2*maxfree+.
26 05176 000000 asl:    .word   0               ; head cell free space
27 05200 000000 find:   .word   0               ; pointers to process descriptors
28      005320          .=2*maxproc+.
29 05320 000000 intflag:.word   0               ; indicates where a send comes from
30                                              ; ( process or interrupt )
31 05322 000000 callsch:.word   0               ; indicates whether scheduler
32                                              ; should be called
33 05324 000000 psave:  .word   0               ; save word for old processnumber
34 05326 000000 count:  .word   0               ; counter for clock
35
36
37
```

```
                        .sbttl  interrupt and trap routines

005330 113702 error:    movb    @#psw,r2        ; get error code
       177776
005334 042702           bic     #177760,r2
       177760
005340 116202           movb    ernumb(r2),r2   ; map to error number
       005412
005344 042702           bic     #177400,r2
       177400

005350 026727 serer:    cmp     process,#highsup; if from system process
       173144
       000030
005356 003010           bgt     userer

05360 004767 super:     jsr     pc,save         ; then save registers in process table
       001444
05364 010405            mov     r4,r5           ; processtable(process)
05366 012703            mov     #down,r3        ; print 'down:'
       011524
05372 004767            jsr     pc,moni         ; print all registers
       003106
05376 000000            halt                    ; halt the system

05400 012700 userer:    mov     #recov,r0       ; if from user process
       000074
05404 012701            mov     #1,r1
       000001
05410 000403            br      iotser          ; then svc(recover,reset,error number)

05412    004 ernumb:    .byte 4,10,14,34,250
05413    010
05414    014
05415    034
05416    250
                        .even

05420 010077 iotser:    mov     r0,@apd         ; supervisor call
       173076
05424 000167            jmp     svc
       000574

05430 000000 powint:    halt                    ; power failure
05432 000776            br      powint

05434 026727 emtser:    cmp     process,#highsup; not allowed
       173060
       000030
05442 003017            bgt     emter           ; for user processes
05444 010077            mov     r0,@apd         ; save r0 in processtable
       173052
05450 011600            mov     (sp),r0         ; pc of caller
05452 006560            mfpi    -2(r0)          ; #n of emt #n in user mode
       177776
05456 112600            movb    (sp)+,r0
05460 042700            bic     #177774,r0
```

```
            177774
37  05464 006300        asl     r0                  ; switch address
38  05466 016007        mov     sw(r0),pc           ; case #n of
            005472
39
40  05472 005652 sw:    .word   signal,wait,send,receive ; switch table
    05474 005564
    05476 006114
    05500 005744
41
42  05502 005002 emter: clr     r2                  ; error 0
43  05504 000735        br      userer
44
45  05506 005367 kwpint: dec    count               ; count=count-1
            177614
46  05512 001411        beq     sclomes             ; if count#0
47  05514 000002        rti                         ; then return
48
49  05516 012737 kwlint: mov    #100,@#csrkwl       ; clear monitor bit
            000100
            177546
50  05524 162767        sub     #20.,count          ; count=count-20
            000024
            177574
51  05532 003401        ble     sclomes             ; if count<=0
52  05534 000002        rti                         ; then return
53
54  05536 010077 sclomes:mov    r0,@apd             ; generate message to clock process
            172760
55  05542 012700        mov     #closer,r0
            000000
56  05546 000167        jmp     intsend
            000244
57
58  05552 010077 diskint:mov    r0,@apd             ; signal(dsem)
            172744
59  05556 012700        mov     #dsem,r0
            000001
60  05562 000437        br      sig1
61
```

WAIT AND SIGNAL

```
1                               .sbttl  wait and signal
2
3 005564 017700 wait:   mov     @apd,r0
         172732
4 005570 042700         bic     #semmask,r0
         177740
5 005574 006300         asl     r0
6 005576 006300         asl     r0                  ; with semaphore(r0) do
7 005600 005760         tst     counter(r0)         ; if counter>0
         000526
8 005604 001405         beq     wait1               ; then
9 005606 005360         dec     counter(r0)         ; counter:=counter-1
         000526
10 05612 017700         mov     @apd,r0
         172704
11 05616 000002         rti
12 05620 004767 wait1:  jsr     pc,save             ; else
         001204
13 05624 016701         mov     process,r1
         172670
14 05630 012702         mov     #waiting,r2
         000530
15 05634 060002         add     r0,r2
16 05636 004767         jsr     pc,enter            ; enter(process,waiting)
         001564
17 05642 004767         jsr     pc,select           ; select(process,ready)
         001772
18 05646 000167         jmp     exit
         001330
19
20 05652 017700 signal: mov     @apd,r0             ; recover sem
         172644
21 05656 042700         bic     #semmask,r0         ; restrict range
         177740
22 05662 006300 sig1:   asl     r0
23 05664 006300         asl     r0                  ; with semaphore(r0) do
24 05666 105760         tstb    waiting(r0)         ; if empty(waiting)
         000530
25 05672 001005         bne     sig2                ; then
26 05674 005260         inc     counter(r0)         ; counter:=counter+1
         000526
27 05700 017700         mov     @apd,r0
         172616
28 05704 000002         rti
29 05706 004767 sig2:   jsr     pc,save             ; else
         001116
30 05712 012702         mov     #waiting,r2
         000530
31 05716 060002         add     r0,r2
32 05720 004767         jsr     pc,remove           ; remove(candidate,waiting)
         001564
33 05724 010103         mov     r1,r3
34 05726 006303         asl     r3
35 05730 016305         mov     find(r3),r5
         005200
36 05734 004767         jsr     pc,resched          ; reschedule(process,candidate)
         001624
```

```
37 05740 000167          jmp     exit
         001236
38
```

```
                        .sbttl  receive and send
;
; 005744 004767 receive:jsr     pc,save         ; r4->proctab(proc)
        001060
; 005750 105764         tstb    recd(r4)        ; if not empty(received)
        000022
; 005754 001411         beq     rec1            ; then get message from queue
; 005756 004767         jsr     pc,unqmess
        000652
; 005762 010546         mov     r5,-(sp)
; 005764 010405         mov     r4,r5
; 005766 004767         jsr     pc,rmoni
        002444
;10 05772 012605         mov     (sp)+,r5
;11 05774 000167         jmp     exit
        001202
;12 06000 112764 rec1:   movb    #true,heldup(r4) ; else heldup:=true
        000001
        000025
;13 06006 004767         jsr     pc,select
        001626
;14 06012 000167         jmp     exit
        001164
;15
;16 06016 004767 intsend:jsr     pc,save         ; save registers in process table
        001006
;17 06022 005267         inc     intflag         ; indicate send from interrupt
        177272
;18 06026 004767         jsr     pc,comsen       ; do common send operations
        000320
;19 06032 001021         bne     1$              ; have to qmessage
;20 06034 004767         jsr     pc,rmoni
        002376
;21 06040 110015         movb    r0,(r5)         ; service number
;22 06042 000315         swab    (r5)
;23 06044 110015         movb    r0,(r5)         ; source:=service
;24 06046 016765         mov     par1,2(r5)
        003176
        000002
;25 06054 016765         mov     par2,4(r5)
        003172
        000004
;26 06062 116001         movb    serv(r0),r1
        003372
;27 06066 004767         jsr     pc,resched
        001472
;28 06072 000167         jmp     exit
        001104
;29 06076 004767 1$:     jsr     pc,rmoni
        002334
;30 06102 010504         mov     r5,r4           ; else qmessage(candidate)
;31 06104 004767         jsr     pc,qmess
        000572
;32 06110 000167         jmp     exit
        001066
;33
;34 06114 004767 send:   jsr     pc,save         ; r4->proctab(proc)
```

```
                000710
5 06120 004767          jsr     pc,smoni
                002330
5 06124 011400          mov     (r4),r0
7 06126 042700          bic     #servmk,r0      ; restrict range of serviceno
                177600
3 06132 020027          cmp     r0,#luserv
                000106
9 06136 002402          blt     1$
0 06140 005237          inc     @#callsch       ; call scheduler if idle is selected
                005322
1 06144 004767 1$:      jsr     pc,comsen       ; do common send operations
                000202
2 06150 001016          bne     send1           ; have to qmessage
3 06152 010401          mov     r4,r1
4 06154 010502          mov     r5,r2
5 06156 004767          jsr     pc,copy         ; copy(    )
                002072
6 06162 004767          jsr     pc,rmoni
                002250
7 06166 110015          movb    r0,(r5)         ; service number
8 06170 000315          swab    (r5)
9 06172 116001          movb    serv(r0),r1
                003372
0 06176 004767          jsr     pc,resched
                001362
1 06202 000167          jmp     exit
                000774
2 06206 020027 send1:   cmp     r0,#luserv      ; messages to user which
                000106
3 06212 002002          bge     1$              ; have to be queued are ignored
4 06214 004767          jsr     pc,qmess        ; qmessage
                000462
5 06220 000167 1$:      jmp     exit
                000756
6
```

```
SUPERVISOR CALL

                        .sbttl  supervisor call
                ;
006224 004767 svc:      jsr     pc,save         ; r4-> proctab(proc)
       000600
006230 004767           jsr     pc,smoni
       002220
006234 011400           mov     (r4),r0
006236 042700           bic     #servmk,r0      ; restrict range of serviceno
       177600
006242 110064           movb    r0,asleep(r4)   ; sending process becomes asleep
       000032
                                                ; until reply from same service
006246 004767           jsr     pc,comsen       ; do common send operations
       000100
06252 001031            bne     svc1            ; have to qmessage
06254 010401            mov     r4,r1
06256 010502            mov     r5,r2
06260 004767            jsr     pc,copy         ; copy( )
       001770
06264 004767            jsr     pc,rmoni
       002146
06270 110015            movb    r0,(r5)         ; service number
06272 000315            swab    (r5)
06274 026727            cmp     process,#highsup; if user process
       172220
       000030
06302 003404            ble     1$
06304 116715            movb    process,(r5)    ; then source:=process number
       172210
06310 062715            add     #usminup,(r5)   ; source:=service number
       000055
06314 026727 1$:        cmp     recproc,#highsup; if receiving process
       175252
       000030
06322 003007            bgt     svc2            ; is supervisor process
06324 016701            mov     recproc,r1      ; then enter it
       175242
06330 004767            jsr     pc,intread      ; into ready queue
       000154
06334 000402            br      svc2
06336 004767 svc1:      jsr     pc,qmess        ; else qmessage
       000340
06342 004767 svc2:      jsr     pc,select
       001272
06346 000167            jmp     exit
       000630
```

COMMON SEND OPERATIONS

```
1                               .sbttl  common send operations
2
3 006352 116003 comsen: movb    serv(r0),r3     ; if illegal service number
         003372
4 006356 002005         bge     1$
5 006360 012702         mov     #1,r2           ; then simulate error trap
         000001
6 006364 005726         tst     (sp)+
7 006366 000167         jmp     serer
         176756
8 006372 010367 1$:     mov     r3,recproc
         175174
9 006376 006303         asl     r3
10 06400 016305         mov     find(r3),r5     ; with processtable(candidate) do
         005200
11 06404 105765         tstb    heldup(r5)      ; if not heldup
         000025
12 06410 001031         bne     hldup
13 06412 105765         tstb    asleep(r5)      ; and not asleep
         000032
14 06416 002432         blt     setqmes
15 06420 026727         cmp     process,#highsup
         172074
         000030
16 06426 003003         bgt     2$              ; if from user process
17 06430 005767         tst     intflag         ; or from interrupt then
         176664
18 06434 001407         beq     3$
19 06436 120065 2$:     cmpb    r0,asleep(r5)     ; if not asleep until service
         000032
20 06442 001020         bne     setqmess          ; then set qmessage exit
21 06444 112765         movb    #notasl,asleep(r5) ; else set process not asleep
         000377
         000032
22 06452 000412         br      notqmes           ; and take not qmess exit
23 06454 126465 3$:     cmpb    1(r4),asleep(r5) ; else source filled in by process
         000001
         000032
24 06462 001010         bne     setqmess
25 06464 112765         movb    #notasl,asleep(r5)
         000377
         000032
26 06472 000402         br      notqmes
27 06474 105065 hldup:  clrb    heldup(r5)      ; heldup:=false
         000025
28 06500 000264 notqmes:sez                     ; set zero bit: not qmess
29 06502 000207         rts     pc
30 06504 000244 setqmes:clz                     ; clear zero bit: qmess
31 06506 000207         rts     pc
32
```

PROCESS INTO READY QUEUE

```
1                       .sbttl  process into ready queue
2
3 006510 016700 intread:mov     begrdy,r0       ; pointer to begin ready queue
         174652
4 006514 020067         cmp     r0,endrdy       ; if equal to pointer to end
         174650
5 006520 001435         beq     4$              ; then queue is empty
6 006522 126001 1$:     cmpb    readyq(r0),r1   ; queue in increasing order
         003326
7 006526 003007         bgt     2$              ; process should be inserted here
8 006530 005200         inc     r0
9 006532 042700         bic     #rdymask,r0     ; ready queue is cyclic
         177740
10 06536 020067         cmp     r0,endrdy
         174626
11 06542 001367         bne     1$              ; reached the end ?
12 06544 000423         br      4$
13
14 06546 016700 2$:     mov     endrdy,r0       ; shuffle rest of queue
         174616
15 06552 010002         mov     r0,r2           ; r0->entry
16 06554 005302         dec     r2              ; r2->previous entry
17 06556 042702         bic     #rdymask,r2
         177740
18 06562 116260 3$:     movb    readyq(r2),readyq(r0)
         003326
         003326
19 06570 010200         mov     r2,r0
20 06572 020067         cmp     r0,begrdy       ; reached the beginning ?
         174570
21 06576 001406         beq     4$
22 06600 005302         dec     r2
23 06602 042702         bic     #rdymask,r2
         177740
24 06606 126201         cmpb    readyq(r2),r1   ; enter position reached ?
         003326
25 06612 003363         bgt     3$
26
27 06614 110160 4$:     movb    r1,readyq(r0)   ; enter process into queue
         003326
28 06620 005267         inc     endrdy          ; step end pointer
         174544
29 06624 042767         bic     #rdymask,endrdy
         177740
         174536
30 06632 000207         rts     pc
31
```

```
1                       .sbttl  message to and from queue
2
3 006634 010402 unqmess:mov     r4,r2
4 006636 062702         add     #recd,r2        ; r2-> recd(r4)
         000022
5 006642 004767         jsr     pc,remove       ; remove(m,received)
         000642
6 006646 010103         mov     r1,r3
7 006650 070127         mul     #12.,r1
         000014
8 006654 062701         add     #messtab,r1     ; r1->message table entry
         004174
9 006660 010402         mov     r4,r2
10 06662 004767         jsr     pc,copy
         001366
11 06666 010301         mov     r3,r1
12 06670 012702         mov     #free,r2
         004774
13 06674 004767         jsr     pc,enter        ; enter(m,free)
         000526
14 06700 000207         rts     pc
15
16 06702 012702 qmess:  mov     #free,r2
         004774
17 06706 010046         mov     r0,-(sp)
18 06710 004767         jsr     pc,remove       ; remove(m,free)
         000574
19 06714 010103         mov     r1,r3
20 06716 070127         mul     #12.,r1
         000014
21 06722 062701         add     #messtab,r1
         004174
22 06726 010102         mov     r1,r2           ; r2-> message table entry
23 06730 010401         mov     r4,r1           ; r4-> curr proc table
24 06732 004767         jsr     pc,copy
         001316
25 06736 112612         movb    (sp)+,(r2)      ; service number
26 06740 000312         swab    (r2)
27 06742 005767         tst     intflag         ; if from interrupt
         176352
28 06746 001411         beq     1$
29 06750 016762         mov     par1,2(r2)      ; then fill in message
         002274
         000002
30 06756 016762         mov     par2,4(r2)
         002270
         000004
31 06764 116212         movb    1(r2),(r2)      ; and source:=service
         000001
32 06770 000410         br      2$
33 06772 026727 1$:     cmp     process,#highsup; if from user process
         171522
         000030
34 07000 003404         ble     2$
35 07002 116712         movb    process,(r2)    ; then source:=process number
         171512
36 07006 062712         add     #usminup,(r2)   ; source:=service number
```

MESSAGE TO AND FROM QUEUE

```
                000055
37  07012 010301 2$:        mov     r3,r1
38  07014 010502             mov     r5,r2
39  07016 062702             add     #recd,r2
                000022
40  07022 004767             jsr     pc,enter
                000400
41  07026 000207             rts     pc
42
```

SAVE PROCESS STATUS

```
                        .sbttl  save process status
                ;
007030 016767   save:   mov     process,psave   ; save processnumber
       171464
       176266
007036 026727           cmp     process,#idle
       171456
       000047
007044 001003           bne     1$
007046 016704           mov     apd,r4
       171450
007052 000207           rts     pc
007054 010046   1$:     mov     r0,-(sp)
007056 016700           mov     apd,r0
       171440
07062 010160            mov     r1,2(r0)
       000002
07066 010260            mov     r2,4(r0)
       000004
07072 010360            mov     r3,6(r0)
       000006
07076 010460            mov     r4,10(r0)
       000010
07102 010560            mov     r5,12(r0)
       000012
07106 012704            mov     #kstk-2,r4
       000516
07112 011460            mov     (r4),20(r0)
       000020
07116 014460            mov     -(r4),16(r0)
       000016
07122 006506            mfpi    sp              ; stackpointer previous mode
07124 012660            mov     (sp)+,14(r0)    ; save stackpointer
       000014
07130 026727            cmp     process,#frstus ; if user process
       171364
       000031
07136 002416            blt     savout          ; then save pdr register
07140 016701            mov     process,r1      ; address
       171354
07144 162701            sub     #frstus,r1
       000031
07150 072127            ash     #5,r1           ; in segmenttable
       000005
07154 062701            add     #userapr+pdr,r1 ; from pdr registers
       003654
07160 012702            mov     #stpdr,r2       ; start address user page descriptor regist
       177600
07164 012703            mov     #10,r3
       000010
07170 052221    next:   bis     (r2)+,(r1)+
07172 077302            sob     r3,next
07174 010004    savout: mov     r0,r4
07176 012600            mov     (sp)+,r0
07200 000207            rts     pc
```

KERNEL EXIT

```
1                               .sbttl  kernel exit
2
3 007202 005067 exit:   clr     intflag         ; clear send flag
         176112
4 007206 026727         cmp     process,#idle
         171306
         000047
5 007214 001474         beq     idlex
6 007216 026767         cmp     process,psave   ; if newprocess=oldprocess
         171276
         176100
7 007224 001437         beq     exout
8 007226 026727         cmp     process,#frstus ; or
         171266
         000031
9 007234 002007         bge     user            ; newprocess is supervisor process
10 07236 026727         cmp     psave,#frstus   ; and oldprocess is supervisor process  the
         176062
         000031
11 07244 002427         blt     exout           ; no changes memory management registers
12 07246 012701         mov     #suppar,r1      ; else
         003574
13 07252 000410         br      fill            ; find
14 07254 016701 user:   mov     process,r1      ; address
         171240
15 07260 162701         sub     #frstus,r1
         000031
16 07264 072127         ash     #5,r1
         000005
17 07270 062701         add     #userapr,r1     ; memory management registers
         003634
18 07274 010102 fill:   mov     r1,r2           ; fill memory management registers
19 07276 062702         add     #pdr,r2         ; r1=startaddress apr,r2=startaddress pdr t
         000020
20 07302 012703         mov     #stpar,r3       ; start address user page address registers
         177640
21 07306 012704         mov     #stpdr,r4       ; start address user page descriptor regist
         177600
22 07312 012705         mov     #10,r5
         000010
23 07316 012123 fagain: mov     (r1)+,(r3)+
24 07320 012224         mov     (r2)+,(r4)+
25 07322 077503         sob     r5,fagain
26 07324 016700 exout:  mov     apd,r0
         171172
27 07330 016001         mov     2(r0),r1
         000002
28 07334 016002         mov     4(r0),r2
         000004
29 07340 016003         mov     6(r0),r3
         000006
30 07344 016004         mov     10(r0),r4
         000010
31 07350 016005         mov     12(r0),r5
         000012
32 07354 012737         mov     #30340,@#psw    ; current mode=kernel,previous mode=user
         030340
```

```
                177776
33 07362 016046         mov     14(r0),-(sp)    ; mov usersp to
                000014
34 07366 006606         mtpi    sp              ; stackpointer previous mode
35 07370 016016         mov     16(r0),(sp)     ; pc and psw on kernel stack
                000016
36 07374 016066         mov     20(r0),2(sp)
                000020
                000002
37 07402 011000         mov     (r0),r0
38 07404 000002         rti
39
40 07406 016700 idlex:  mov     apd,r0
                171110
41 07412 016016         mov     16(r0),(sp)
                000016
42 07416 016066         mov     20(r0),2(sp)
                000020
                000002
43 07424 000002         rti
44
```

```
1                               .sbttl  despatcher
2
3 007564 020167 resched:cmp     r1,psave        ; if candidate<=process
         175534
4 007570 002012         bge     1$
5 007572 020127         cmp     r1,#highsup     ; and candidate
         000030
6 007576 003007         bgt     1$              ; is a supervisor process
7 007600 010167         mov     r1,process
         170714
8 007604 010567         mov     r5,apd          ; then process:=candidate
         170712
9 007610 116701         movb    psave,r1        ; old process
         175510
10 07614 000403         br      2$              ; into ready queue
11 07616 016767 1$:     mov     psave,process   ; else continue old process
         175502
         170674
12 07624 020127 2$:     cmp     r1,#highsup
         000030
13 07630 003002         bgt     3$
14 07632 004767         jsr     pc,intread      ; enter process into ready queue
         176652
15 07636 000207 3$:     rts     pc
16
17 07640 026767 select: cmp     begrdy,endrdy   ; if ready queue empty
         173522
         173522
18 07646 001412         beq     1$              ; then no supervisor process ready
19 07650 016700         mov     begrdy,r0       ; else get
         173512
20 07654 116000         movb    readyq(r0),r0   ; supervisor process
         003326
21 07660 005267         inc     begrdy          ; from queue
         173502
22 07664 042767         bic     #rdymask,begrdy ; ready queue is cyclic
         177740
         173474
23 07672 000405         br      2$
24 07674 016700 1$:     mov     runproc,r0      ; else
         170624
25 07700 020027         cmp     r0,#idle        ; is the idle process ?
         000047
26 07704 001435         beq     selidl
27 07706 006300 2$:     asl     r0
28 07710 016001         mov     find(r0),r1
         005200
29 07714 105761         tstb    asleep(r1)      ; if not userprocess.asleep
         000032
30 07720 002006         bge     sch
31 07722 006200         asr     r0
32 07724 010067         mov     r0,process      ; then start this process
         170570
33 07730 010167         mov     r1,apd
         170566
34 07734 000207         rts     pc
35
```

```
36 07736 005037 sch:    clr     @#callsch
         005322
37 07742 012700         mov     #schedu,r0      ; else start scheduler
         000030
38 07746 006300         asl     r0
39 07750 016001         mov     find(r0),r1
         005200
40 07754 105061         clrb    heldup(r1)
         000025
41 07760 012767         mov     #schedu,process
         000030
         170532
42 07766 010167         mov     r1,apd
         170530
43 07772 012711         mov     #6,(r1)         ; for scheduler to know where
         000006
44 07776 000207         rts     pc              ; it is called from
45
46 10000 005737 selidl: tst     @#callsch       ; select idle process
         005322
47 10004 001354         bne     sch             ; if scheduler does not have to be called
48 10006 010067         mov     r0,process
         170506
49 10012 006300         asl     r0
50 10014 016067         mov     find(r0),apd
         005200
         170500
51 10022 000207         rts     pc
52
```

INITIALIZATION OF DATA AREA AND IDLE PROCESS

```
1                               .sbttl  initialization of data area and idle process
2
3  010024 000005 start:         reset
4  010026 012706                mov     #initstk,sp     ; set up stack pointer
          000000G
5  010032 012737                mov     #30340,@#psw    ; processor status word
          030340
          177776
6  010040 012700                mov     #setboot,r0     ; set fixed bootstrap rt11 address
          010250
7  010044 012037                mov     (r0)+,@#50
          000050
8  010050 011037                mov     (r0),@#52
          000052
9  010054 012701                mov     #process,r1
          000520
10 10060 012702                mov     #psave,r2
          005324
11 10064 004767                jsr     pc,clear        ; clear whole data area
          000226
12 10070 004767                jsr     pc,findst       ; set find table
          000314
13 10074 004767                jsr     pc,aslist       ; set available space list
          000226
14 10100 004767                jsr     pc,freest       ; set queue of free buffers
          000254
15
16                             ; initialization clock
17
18       172542                csbkwp=172542
19       172540                caskwp=172540
20       177546                csrkwl=177546
21
22 10104 012737                mov     #1000.,@#count  ; set counter
          001750
          005326
23 10112 012737                mov     #kw111,@#4      ; set trap vector
          010160
          000004
24 10120 012737                mov     #30340,@#6      ; for time out error
          030340
          000006
25 10126 012737                mov     #kwpint,@#100   ; set interrupt vector kw11-p
          005506
          000100
26 10134 012737                mov     #340,@#102
          000340
          000102
27 10142 012737                mov     #144,@#csbkwp   ; try kw11-p first
          000144
          172542
28 10150 012737                mov     #111,@#caskwp   ; set count set buffer and control/status
          000111
          172540
29 10156 000413                br      inidle
30
31 10160 062706 kw111:         add     #4,sp           ; if kw11-p not there then kw11-l
```

```
                000004
32 10164 012737         mov     #kwlint,@#104   ; set interrupt vector kw11-l
         005516
         000104
33 10172 012737         mov     #340,@#106
         000340
         000106
34 10200 012737         mov     #100,@#csrkwl   ; set clock status register
         000100
         177546
35
36                      ; initialization of idle process
37
38 10206 012700 inidle: mov     #maxproc,r0     ; idle process has highest number
         000047
39 10212 006300         asl     r0
40 10214 016000         mov     find(r0),r0
         005200
41 10220 005060         clr     20(r0)          ; processor status word
         000020
42 10224 012760         mov     #idlepc,16(r0)  ; program counter
         010244
         000016
43 10232 012767         mov     #idle,runproc   ; idle process running user process
         000047
         170264
44 10240 000137         jmp     @#init          ; address of initialisation of processes
         000000G
45
46
47                      ; idle process code
48
49 10244 000001 idlepc: wait
50 10246 000776         br      idlepc
51
52
53 10250 000137 setboot:jmp     @#bootrt        ; bootstrap rt11
         011532
54
```

```
                        .sbttl  routines
                        ; copy message
010254 011112 copy:     mov     (r1),(r2)
010256 016162           mov     2(r1),2(r2)
       000002
       000002
010264 016162           mov     4(r1),4(r2)
       000004
       000004
010272 016162           mov     6(r1),6(r2)
       000006
       000006
10300 016162            mov     10(r1),10(r2)
       000010
       000010
10306 016162            mov     12(r1),12(r2)
       000012
       000012
10314 000207            rts     pc

                        ; clear area

10316 005021 clear:     clr     (r1)+           ; r1  start address
10320 020102            cmp     r1,r2           ; r2 end address
10322 101775            blos    clear
10324 000207            rts     pc

                        ; set up available space list

10326 012701 aslist:    mov     #link+2,r1
       005000
10332 005002            clr     r2
10334 110221 1$:        movb    r2,(r1)+
10336 005201            inc     r1
10340 005202            inc     r2
10342 020227            cmp     r2,#maxfree
       000077
10346 002772            blt     1$
10350 012767            mov     #maxfree,asl
       000077
       174620
10356 000207            rts     pc

                        ; set up queue of free message buffers

10360 005067 freest:    clr     free
       174410
10364 012702            mov     #free,r2
       004774
10370 005001            clr     r1
10372 004767 1$:        jsr     pc,enter
```

```
                177030
42  10376 005201        inc     r1
43  10400 020127        cmp     r1,#maxmess
          000037
44  10404 003772        ble     1$
45  10406 000207        rts     pc
46
47
48                      ; set up pointers to process descriptors
49
50  10410 012700 findst: mov    #find,r0
          005200
51  10414 012701        mov     #proctab,r1
          000726
52  10420 010120 1$:    mov     r1,(r0)+
53  10422 062701        add     #40,r1
          000040
54  10426 020027        cmp     r0,#2*maxproc+2+find
          005320
55  10432 001372        bne     1$
56  10434 000207        rts     pc
57
```

```
1                               .sbttl  monitoring routines
2
3 010436 032765 rmoni:  bit     #1,rsmoni(r5)   ; receive monitored ?
         000001
         000030
4 010444 001416         beq     monex
5 010446 012703         mov     #re,r3
         011516
6 010452 000414         br      moni
7 010454 032764 smoni:  bit     #2,rsmoni(r4)   ; send monitored ?
         000002
         000030
8 010462 001407         beq     monex
9 010464 012703         mov     #sen,r3
         011510
10 10470 010546         mov     r5,-(sp)
11 10472 010405         mov     r4,r5
12 10474 004767         jsr     pc,moni
         000004
13 10500 012605         mov     (sp)+,r5
14 10502 000207 monex:  rts     pc
15
16 10504 010546 moni:   mov     r5,-(sp)
17 10506 010446         mov     r4,-(sp)
18 10510 010046         mov     r0,-(sp)
19 10512 004767         jsr     pc,crlf         ; print cr and lf
         000670
20 10516 004767         jsr     pc,prntxt       ; print text
         000650
21 10522 016704         mov     process,r4      ; print process
         167772
22 10526 004767         jsr     pc,sproct
         000676
23 10532 012767         mov     #10,tel
         000010
         000024
24 10540 012504 1$:     mov     (r5)+,r4
25 10542 004767         jsr     pc,sproct       ; print registers
         000662
26 10546 005367         dec     tel
         000012
27 10552 001372         bne     1$
28 10554 012600         mov     (sp)+,r0
29 10556 012604         mov     (sp)+,r4
30 10560 012605         mov     (sp)+,r5
31 10562 000207         rts     pc
32
33 10564 000000 tel:    .word   0
34
```

```
                        .sbttl  console input and output routines

            000002      ident=2
            000004      maxlin=4
            000010      seman=10
            000012      dmasig=12
            000014      mode=14
            000016      outadr=16
            000020      outmap=20
            000022      numbyt=22
            000024      linwid=24
            000026      state=26
            000030      dmouts=30

10566 013746 177776 consin: mov     @#psw,-(sp)     ; psw contains console number
10572 010077 167724         mov     r0,@apd         ; save r0
10576 012600                mov     (sp)+,r0
10600 042700 177760         bic     #177760,r0      ; console number
10604 010146                mov     r1,-(sp)        ; save r1
10606 070027 000034         mul     #28.,r0         ; determine position in
10612 062701 000000G        add     #condes,r1      ; console descriptor array
10616 011100                mov     (r1),r0         ; address
10620 016067 000002 000424  mov     2(r0),par2      ; character
10626 005067 000416         clr     par1            ; input
10632 016100 000002         mov     ident(r1),r0    ; console service
10636 012601                mov     (sp)+,r1        ; restore r1
10640 000167 175152         jmp     intsend

10644 013746 177776 consout:mov     @#psw,-(sp)     ; psw contains console number
10650 010077 167646         mov     r0,@apd         ; save r0
10654 012600                mov     (sp)+,r0
10656 042700 177760         bic     #177760,r0      ; console number
10662 010146                mov     r1,-(sp)        ; save r1
10664 070027 000034         mul     #28.,r0         ; determine position in
10670 062701 000000G        add     #condes,r1      ; console decriptor array
10674 005761 000012         tst     dmasig(r1)
10700 001405                beq     1$              ; interrupt converted to signal
10702 016100 000010         mov     seman(r1),r0    ; semaphore number
10706 012601                mov     (sp)+,r1        ; restore r1
10710 000167                jmp     sig1
```

```
                174746
41  10714 010246 1$:     mov     r2,-(sp)        ; interrupt handled immediately
42  10716 010346          mov     r3,-(sp)
43  10720 016100          mov     outadr(r1),r0   ; output address
                000016
44  10724 011102          mov     (r1),r2         ; console address
45  10726 005062          clr     4(r2)           ; disable interrupts
                000004
46  10732 016137          mov     outmap(r1),@#kapr6 ; output map register
                000020
                172354
47  10740 005761          tst     mode(r1)
                000014
48  10744 001035          bne     raw             ; ascii mode
49  10746 016103          mov     dmouts(r1),r3   ; output state
                000030
50  10752 006303          asl     r3
51  10754 016307          mov     outsw(r3),pc    ; goto state
                010760
52
53  10760 010770 outsw:   st0,st1,st2,st3
    10762 011070
    10764 011146
    10766 011162
54
55  10770 121027 st0:     cmpb    (r0),#lf        ; if lf then
                000012
56  10774 001006          bne     1$
57  10776 112762          movb    #cr,6(r2)       ; output cr
                000015
                000006
58  11004 005261          inc     dmouts(r1)      ; state=1
                000030
59  11010 000474          br      exdma
60  11012 026161 1$:      cmp     linwid(r1),maxlin(r1) ; if line width=maximum line width
                000024
                000004
61  11020 001007          bne     raw
62  11022 112762          movb    #cr,6(r2)       ; output cr
                000015
                000006
63  11030 012761          mov     #2,dmouts(r1)   ; state=2
                000002
                000030
64  11036 000461          br      exdma
65  11040 111062 raw:     movb    (r0),6(r2)      ; output character
                000006
66  11044 005261          inc     outadr(r1)      ; step address
                000016
67  11050 005261          inc     linwid(r1)      ; line width
                000024
68  11054 005361          dec     numbyt(r1)      ; if number of bytes=0
                000022
69  11060 001050          bne     exdma
70  11062 005067          clr     par2            ; then send message to handler
                000164
71  11066 000456          br      senmes
```

```
11070 112762 st1:    movb   #lf,6(r2)       ; output lf
      000012
      000006
11076 005061         clr    dmouts(r1)      ; state=0
      000030
11102 005261         inc    outadr(r1)      ; step address
      000016
11106 005061         clr    linwid(r1)      ; line width=0
      000024
11112 005361         dec    numbyt(r1)      ; if number of bytes=0
      000022
11116 001003         bne    1$
11120 005067         clr    par2            ; then send message to handler
      000126
11124 000437         br     senmes
11126 026127 1$:     cmp    state(r1),#7    ; if state=outwinp then
      000026
      000007
11134 001022         bne    exdma
11136 012767         mov    #1,par2         ; then send message to handler
      000001
      000106
11144 000427         br     senmes

11146 112762 st2:    movb   #lf,6(r2)       ; output lf
      000012
      000006
11154 005261         inc    dmouts(r1)      ; state=3
      000030
11160 000410         br     exdma

11162 112762 st3:    movb   #'>,6(r2)       ; output >
      000076
      000006
11170 012761         mov    #1,linwid(r1)   ; line width=1
      000001
      000024
11176 005061         clr    dmouts(r1)      ; state=0
      000030

11202 012762 exdma:  mov    #100,4(r2)      ; enable further interrupts
      000100
      000004
11210 012603         mov    (sp)+,r3        ; exit dma
11212 012602         mov    (sp)+,r2
11214 012601         mov    (sp)+,r1
11216 017700         mov    @apd,r0
      167300
11222 000002         rti

1224 016100 senmes:  mov    ident(r1),r0    ; console service number
      000002
1230 012603          mov    (sp)+,r3        ; send message to console process
1232 012602          mov    (sp)+,r2
1234 012601          mov    (sp)+,r1
1236 012767          mov    #1,par1         ; output
```

```
                000001
                000004
106  1244 000167          jmp      intsend
                174546
107
108  1250 000000 par1:    .word    0
109  1252 000000 par2:    .word    0
110
```

DUMP

```
1                               .sbttl  dump
2
3 011254 004767 dump:   jsr     pc,read         ; read bottom address
         000066
4 011260 010100         mov     r1,r0
5 011262 004767         jsr     pc,read         ; read top address
         000060
6 011266 004767         jsr     pc,crlf         ; newline
         000114
7 011272 160001         sub     r0,r1
8 011274 006201         asr     r1              ; length=(top-bottom)/2
9 011276 042701         bic     #100000,r1      ; highest bit away
         100000
10 11302 004767 1$:     jsr     pc,crlf
         000100
11 11306 010004         mov     r0,r4
12 11310 004767         jsr     pc,proct        ; print first address
         000124
13 11314 012705         mov     #10,r5          ; line count
         000010
14 11320 012702         mov     #'>,r2          ; print '>'
         000076
15 11324 004767         jsr     pc,print
         000026
16 11330 012004 2$:     mov     (r0)+,r4
17 11332 004767         jsr     pc,sproct
         000072
18 11336 005301         dec     r1              ; last word printed ?
19 11340 003745         ble     dump
20 11342 077506         sob     r5,2$
21 11344 000756         br      1$
22
23 11346 000000 read:   halt
24 11350 013701         mov     @#177570,r1     ; read from switches
         177570
25 11354 000207         rts     pc
26
```

PRINT ROUTINES

```
                        .sbttl  print routines
                
011356 105737 print:    tstb    @#177564        ; wait until printer ready
       177564
011362 100375           bpl     print
011364 010237           mov     r2,@#177566     ; put character
       177566
011370 000207           rts     pc
                
011372 112302 prntxt:   movb    (r3)+,r2        ; text ends with 0
011374 001403           beq     1$
 11376 004767           jsr     pc,print        ; print next character
       177754
 11402 000773           br      prntxt
 11404 000207 1$:       rts     pc
                
 11406 012702 crlf:     mov     #cr,r2          ; print cr and lf
       000015
 11412 004767           jsr     pc,print
       177740
 11416 012702           mov     #lf,r2
       000012
 11422 004767           jsr     pc,print
       177730
 11426 000207           rts     pc
                
 11430 012702 sproct:   mov     #' ,r2          ; print octal with leading space
       000040
 11434 004767           jsr     pc,print
       177716
 11440 012703 proct:    mov     #6,r3           ; print octal
       000006
 11444 005002           clr     r2
 11446 006104           rol     r4              ; r4 contains number
 11450 006102           rol     r2
 11452 062702 1$:       add     #60,r2          ; '0' has code 60
       000060
 11456 004767           jsr     pc,print
       177674
 11462 005002           clr     r2
 11464 005303           dec     r3
 11466 001407           beq     2$              ; last digit ?
 11470 006104           rol     r4
 11472 006102           rol     r2
 11474 006104           rol     r4
 11476 006102           rol     r2
 11500 006104           rol     r4
 11502 006102           rol     r2
 11504 000762           br      1$
 11506 000207 2$:       rts     pc
                
 11510     163 sen:     .ascii  /send:/
    11511  145
    11512  156
    11513  144
    11514  072
 11515     000          .byte   0
```

```
42 11516    162 re:     .ascii  /recv:/
   11517    145
   11520    143
   11521    166
   11522    072
43 11523    000         .byte   0
44 11524    144 down:   .ascii  /down:/
   11525    157
   11526    167
   11527    156
   11530    072
45 11531    000         .byte   0
46
```

```
1                               .sbttl  bootstrap rt11 from rk0
2
3 011532 000005 bootrt: reset
4 011534 105737 1$:     tstb    @#177400
         177400
5 011540 100411         bmi     2$
6 011542 005000         clr     r0
7 011544 012701         mov     #10,r1
         000010
8 011550 077001 3$:     sob     r0,3$
9 011552 077102         sob     r1,3$
10 11554 012737         mov     #7,@#177566
         000007
         177566
11 11562 000764         br      1$
12 11564 012700 2$:     mov     #177406,r0
         177406
13 11570 012710         mov     #177400,(r0)
         177400
14 11574 012740         mov     #5,-(r0)
         000005
15 11600 105710 lop:    tstb    (r0)
16 11602 100376         bpl     lop
17 11604 005007         clr     pc
18       011700         .=.-<.&77>+100
19       010024'        .end start
```

SYMBOL TABLE

```
APD     000522 G    ASL     005176      ASLEEP= 000032 G
ASLIST  010326      BEGRDY  003366 G    BOOTRT  011532
CALLSC  005322      CASKWP= 172540      CINEP   000454 G
CLEAR   010316      CLOCKE  000450 G    CLOSER= 000000
COMSEN  006352      CONDES= ****** G    CONSIN  010566
CONSOU  010644      COPY    010254      COUNT   005326 G
COUNTE  000526      COUTEP  000456 G    CR    = 000015
CRLF    011406      CSBKWP= 172542      CSRKWL= 177546
DISKEP  000452 G    DISKIN  005552      DMASIG= 000012
DMOUTS= 000030      DOWN    011524      DSEM  = 000001
DUMP    011254      EMTEP   000446 G    EMTER   005502
EMTSER  005434      ENDRDY  003370 G    ENTER   007426
ENT1    007464      ENT2    007502      ERNUMB  005412
ERROR   005330      ERRORE  000440 G    EXDMA   011202
EXIT    007202      EXOUT   007324      FAGAIN  007316
FALSE = 000000      FILL    007274      FIND    005200 G
FINDST  010410      FREE    004774      FREEST  010360
FRSTUS= 000031      HELDUP= 000025      HIGHSU= 000030
HLDUP   006474      IDENT = 000002      IDLE  = 000047
IDLEPC  010244      IDLEX   007406      INFO    004777
INIDLE  010206      INIT  = ****** G    INITST= ****** G
INTFLA  005320      INTREA  006510      INTSEN  006016
IOTEP   000442 G    IOTSER  005420      KAPR6 = 172354
KERNEX  000460 G    KSTK  = 000520      KWLINT  005516
KWPINT  005506      KW11L   010160      LF    = 000012
LINK    004776      LINWID= 000024      LOP     011600
LUSERV= 000106      MAXFRE= 000077      MAXLIN= 000004
MAXMES= 000037      MAXPRO= 000047      MAXSEM= 000037
MAXUP = 000007      MESSTA  004174      MODE  = 000014
MONEX   010502      MONI    010504      NEXT    007170
NOTASL= 000377      NOTQME  006500      NUMBYT= 000022
OFF   = 000000      ON    = 000001      OUTADR= 000016
OUTMAP= 000020      OUTSW   010760      PAR1    011250
PAR2    011252      PC    =%000007      PDR   = 000020
POWEP   000444 G    POWINT  005430      PRINT   011356
PRNTXT  011372      PROCES  000520 G    PROCT   011440
PROCTA  000726 G    PSAVE   005324      PSW   = 177776
QMESS   006702      RAW     011040      RDYMAS= 177740
RE      011516      READ    011346      READYQ  003326 G
RECD  = 000022      RECEIV  005744      RECOV = 000074
RECPRO  003572      REC1    006000      REMOVE  007510
REM1    007544      REM2    007550      RESCHE  007564
RMONI   010436      RSMONI= 000030      RUNPRO  000524 G
R0    =%000000      R1    =%000001      R2    =%000002
R3    =%000003      R4    =%000004      R5    =%000005
SAVE    007030      SAVOUT  007174      SCH     007736
SCHEDU= 000030      SCLOME  005536      SELECT  007640
SELIDL  010000      SEMAN = 000010      SEMMAS= 177740
SEN     011510      SEND    006114      SEND1   006206
SENMES  011224      SERER   005350      SERV    003372 G
SERVMK= 177600      SETBOO  010250      SETQME  006504
SIGNAL  005652      SIG1    005662      SIG2    005706
SMONI   010454      SP    =%000006      SPROCT  011430
START   010024      STATE = 000026      STPAR = 177640
STPDR = 177600      ST0     010770      ST1     011070
ST2     011146      ST3     011162      SUPER   005360
SUPPAR  003574 G    SUPPDR  003614 G    SVC     006224
```

SYMBOL TABLE

SVC1	006336	SVC2	006342	SW	005472
TEL	010564	TOEXIT	000462	TRUE =	000001
UNQMES	006634	USER	007254	USERAP	003634 G
USERER	005400	USERPS=	140000	USMINU=	000055
WAIT	005564	WAITIN	000530	WAIT1	005620

```
. ABS.   011700     000
         000000     001
ERRORS DETECTED: 0
FREE CORE: 16344. WORDS

KERNEL,KERNEL=TEST
```

```
!**********************
!*** clock handler ***
!**********************

%control k'100001'

%begin

   %constinteger msec=0          ;! index values for the array time
   %constinteger sec=1
   %constinteger min=2
   %constinteger hour=3
   %constinteger day=4
   %constinteger month=5
   %constinteger year=6

   %constinteger lowuser=70      ;! lowest user service number

   %recordformat message (%byteinteger id,servnum,%integer service,msec, %c
                                          %byteintegerarray time(sec:year))
   %recordformat reply (%byteinteger id,source,%integer flag,msec, %c
                                        %byteintegerarray time(sec:year))

   %externalroutinespec receive (%record (message) %name mes)
   %externalroutinespec send (%record (reply) %name rep)

   %externalintegerspec count          ;! counter in the kernel

   %routinespec interval over
   %routinespec adjust time
   %routinespec test waiting queue
   %routinespec message after time
   %routinespec into waiting queue (%integer prev,mstime,stime)
   %routinespec get time
   %routinespec set time
   %routinespec adjust count (%integer interval,%integername negative)
   %routinespec initialize

   %record (message) mes
   %record (reply) rep

   %integerarray time (msec:year)       ;! "software clock"

   ! The values of time are always between a certain minimum and maximum
   ! value. The maximum value for the day in the month varies. Leap-years
   ! are ignored.
   %constintegerarray min time (msec:year)=0,0,0,0,1,1,0
   %constintegerarray max day time (1:12)=31,29,31,30,31,30,31,31,30,31,30,31
   %ownintegerarray max time (msec:year)=1000,60,60,24,32,13,100

   %constinteger false=0
   %constinteger true=1
   %constinteger clock=0

   %switch service (0:2)
```

```
   %recordformat wait format (%integer msec,sec,%byteinteger procid,next)
   %constinteger max waiting=15
   %record (wait format) %array waiting (1:max waiting)

   %integer requested interval,head,free list,extra updates,counter, %c
            message sent,flag


   initialize

   %cycle

      receive(mes)
      rep=0
      flag=0
      %if mes_id=clock %start                ;! message from the clock
         %if extra updates=0 %then interval over %c
                             %else extra updates=extra updates-1
      %else                                  ;! request for a service
         %if 0<=mes_service<=2 %start
            counter=count
            %if counter<=0 %start                        ;! if counter<=0
               extra updates=extra updates+1             ;! then extra update of time
               interval over                             ;! has to be done
            %else                                        ;! else update only
               time(msec)=time(msec)+requested interval-counter ;! msecs
               requested interval=counter                ;! and requested interval
            %finish
            ->service(mes_service)                       ;! select service routine
            service(0): message after time
                        %if flag=0 %then %continue %else ->sendrep
               service(1): get time
                           ->sendrep
               service(2): set time
            %finishelsestart
               flag=1
            %finish
            sendrep: rep_id=mes_id                 ;! identification
                     rep_source=clock
                     rep_flag=flag
                     send(rep)
      %finish

   %repeat


   %routine interval over

   %integer negative

   %until negative=false %cycle
      adjust time                              ;! update time and see
      test waiting queue                       ;! if messages have to be sent
      %if head=0 %or waiting(head)_sec#time(sec) %or %c
       waiting(head)_msec<time(msec) %start
         requested interval=1000                         ;! new value count depends
```

```
      %else                                          ;! on time till msec=1000
         requested interval=waiting(head)_msec       ;! or time next message
      %finish                                        ;! has to be sent
      requested interval=requested interval-time(msec)
      adjust count(requested interval,negative)
   %repeat                                 ;! if count<=0 then again

%end


%routine adjust time

%integer t

time(msec)=time(msec) + requested interval          ;! update msecs
%for t=msec,1,day %cycle
   %if time(t)=max time(t) %start                    ;! if overflow then
      time(t)=min time(t)                           ;! minimum value and
      time(t+1)=time(t+1)+1                         ;! next incremented
   %else
      %return                                       ;! else return
   %finish
%repeat
%if time(month)=max time(month) %start
   time(month)=min time(month)
   max time(day)=max day time(time(month))+1
   time(year)=time(year)+1
   %if time(year)=max time(year) %then time(year)=min time(year)
%finish

%end


%routine test waiting queue

   %integer ptr

   %while head#0 %and waiting(head)_msec=time(msec) %and %c
                      waiting(head)_sec=time(sec) %cycle
      get time                              ;! a message has to be sent
      rep_id=waiting(head)_procid           ;! to this process at this moment
      rep_source=clock
      send(rep)
      ptr=head
      head=waiting(ptr)_next                ;! move from waiting queue
      waiting(ptr)_next=free list
      free list=ptr                         ;! into the free list
   %repeat

%end

%routine message after time

   %integer mstime,stime,prev,ptr
```

```
      %if mes_msec<0 %or mes_msec>1000 %c
        %or mes_time(sec)<0 %or mes_time(sec)>25 %c
        %or (mes_msec=0 %and mes_time(sec)=0) %start
         flag=1                                    ;! illegal request
      %else
         mstime=time(msec)+mes_msec                ;! determine time at
         stime=time(sec)+mes_time(sec)             ;! which a message
         %if mstime>=max time(msec) %start         ;! must be sent back
            mstime=mstime-max time(msec)
            stime=stime+1
         %finish
      %if stime>=max time(sec) %then stime=stime-max time(sec)
      %if head=0 %start                            ;! empty waiting queue
         into waiting queue(0,mstime,stime)        ;! put request into it
      %else                                        ;! else determine position
         prev=0                                    ;! in the waiting queue
         ptr=head
         %if stime>time(sec) %c
         %or (stime=time(sec) %and mstime>time(msec)) %start
            %until ptr=0 %cycle ;! message before present minute ends
               %if waiting(ptr)_sec<time(sec) %c
               %or (waiting(ptr)_sec=time(sec) %and %c
                    waiting(ptr)_msec<time(msec)) %then %exit
               %if waiting(ptr)_sec>stime %c
               %or (waiting(ptr)_sec=stime %and %c
                    waiting(ptr)_msec>=mstime) %then %exit
               prev=ptr
               ptr=waiting(ptr)_next
            %repeat
         %else
            %until ptr=0 %cycle ;! message after present minute ends
               %if waiting(ptr)_sec<time(sec) %c
               %or (waiting(ptr)_sec=time(sec) %and %c
                    waiting(ptr)_msec<time(msec)) %start
                  %if waiting(ptr)_sec>stime %c
                  %or (waiting(ptr)_sec=stime %and %c
                       waiting(ptr)_msec>=mstime) %then %exit
               %finish
               prev=ptr
               ptr=waiting(ptr)_next
            %repeat
         %finish
         into waiting queue(prev,mstime,stime) ;! into queue
      %finish
   %finish

%end

%routine into waiting queue (%integer prev,mstime,stime)
   %integer new,new interval,negative
   new=free list                                ;! get a record
```

```
      free list=waiting(free list)_next ;! from the freelist
      waiting(new)_msec=mstime          ;! set time of message
      waiting(new)_sec=stime
      waiting(new)_procid=mes_id        ;! and identification
      %if prev=0 %start                 ;! first one in queue
         waiting(new)_next=head
         head=new
         %if waiting(head)_sec=time(sec) %c
         %and waiting(head)_msec>time(msec) %start     ;! to be satisfied
            new interval=waiting(head)_msec-time(msec) ;! before present
            adjust count(new interval-requested interval,negative)
            requested interval=new interval           ;! second ends
            %if negative=true %then interval over     ;! time already over
            %if message sent=true %then %c
                  extra updates=extra updates+1 ;! ignore next message clock
         %finish
      %else
         waiting(new)_next=waiting(prev)_next ;! into queue
         waiting(prev)_next=new
      %finish

%end


%routine get time

   %integer t
   rep_flag=0
   rep_msec=time(msec)                ;! make a copy
   %for t=sec,1,year %cycle           ;! of time
      rep_time(t)=time(t)
   %repeat

%end


%routine set time

   %integer t,mdt

   %if mes_id>lowuser %start
      flag=1                          ;! for system manager only
      %return
   %finish
   mdt=max time(day)
   %if mes_msec<0 %or mes_msec>=1000 %then flag=2 %and %return
   %if min time(month)<=mes_time(month)<=max time(month) %c
       %then max time(day)=max day time(mes_time(month))+1
   %for t=sec,1,year %cycle           ;! check the new time
      %if mes_time(t)<min time(t) %or mes_time(t)>=max time(t) %start
         maxtime(day)=mdt
         flag=2                       ;! illegal time
         %return
      %finish
```

```
      %repeat
      time(msec)=mes_msec                    ;! set time
      %for t=sec,1,year %cycle
         time(t)=mes_time(t)                 ;! the given time
      %repeat

   %end


   %routine adjust count (%integer interval,%integername negative)

      %constintegername psw=k'177776'        ;! processor status word
      psw=k'140340'                          ;! disable interrupts clock
      %if count<=0 %then message sent=true %c
                   %else message sent=false
      count=count+interval
      %if count<=0 %then negative=true %else negative=false
      psw=k'140000'                          ;! enable interrupts clock

   %end



   %routine initialize

      %integer i

      %for i=1,1,max waiting %cycle
          waiting(i)=0                       ;! into free list
          waiting(i)_next=i+1
      %repeat
      waiting(max waiting)_next=max waiting
      free list=1
      head=0
      extra updates=0
      %for i=msec,1,year %cycle
         time(i)=min time(i)                 ;! time to minimum values
      %repeat

   %end


%endofprogram
```

```
!*********************
!*** disk driver ***
!*********************

%control k'100001'

%begin

   %recordformat message (%byteinteger id,servnum,service,drive, %c
                                    %integer diskad,start,length,mark)
   %recordformat reply (%byteinteger id,source,%integer flag, %c
                                    mark,par3,par4,par5)

%externalroutinespec receive (%record (message) %name mes)
%externalroutinespec send (%record (reply) %name rep)
%externalroutinespec waitsema (%integer sema)
%externalroutinespec print (%string (63) s)
%externalroutinespec poctal(%integer i)
%externalroutinespec printsymbol(%integer i)

%record (message) mes
%record (reply) rep

%constinteger disksm=1                          ;! disk semaphore number

%recordformat rkf (%integer drivest,error,contrst,wordct,busad,diskad)

%constrecord (rkf) %name rk=k'177400'

%constintegerarray command(0:1)=k'505',k'503' ;! read and write commands

%constinteger lowuser=70                        ;! lowest user service number

%integer flag,com,ba,blk


%cycle

   receive(mes)
   %if mes_id>=lowuser %then flag=1 %and ->setrep ;! no permission
   com=command(mes_service&1)                   ;! command
   %if mes_service&2=0 %start                   ;! mes_stblk=startblock
      com=(mes_start//8&k'60')!com              ;! memory extension bits
      ba=mes_start*512                          ;! bus address
   %else                                        ;! mes_stblk=address
      ba=mes_start                              ;! bus address
   %finish

   %until flag=0 %cycle
      rk_busad=ba                               ;! bus address register
      blk=mes_diskad//12                        ;! convert disk address
      rk_diskad=(mes_drive<<13)!(blk<<4)!(mes_diskad-blk*12)
                                                ;! disk address register
      rk_wordct=-(mes_length*256)               ;! word count register
      rk_contrst=com                            ;! control status register
```

```
        waitsema(disksm)                          ;! wait for interrupt

        %if rk_contrst<0 %start                   ;! error
           %if rk_error&k'20000'#0 %thenstart     ;! write lock out violation
              print('          *** disk on write protect ***')
           %else
              print('          ****     disk error      ****')
              poctal(rk_drivest)
              poctal(rk_error)
              poctal(rk_contrst)                  ;! dump the registers
              poctal(rk_wordct)
              poctal(rk_busad)
              poctal(rk_diskad)
              printsymbol(k'15');printsymbol(k'12')
           %finish
              rk_contrst=1                        ;! do a reset
              flag=1
           %else flag=0
        %repeat until flag=0

        setrep: rep=0
                rep_id=mes_id                     ;! identification
                rep_source=mes_servnum
                rep_flag=flag
                rep_mark=mes_mark                 ;! marker
                send(rep)

     %repeat

%endofprogram
```

```
!**********************
!*** core manager ***
!**********************

%control k'100001'

%begin

   %recordformat message (%byteinteger id,servnum,%integer service, %c
                                                    start,length,par4,par5)
   %recordformat reply (%byteinteger id,source,%integer flag,start, %c
                                                      length,par4,par5)

%externalroutinespec receive (%record (message) %name mes)
%externalroutinespec send (%record (reply) %name rep)

%routinespec get core
%routinespec release core
%routinespec return record (%integer p)
%routinespec initialize

%record (message) mes
%record (reply) rep

%recordformat core area (%integer start,length,%byteinteger prev,next)

%constinteger max free area=40   ;! maximum number of free areas
%record (core area) %array area (1:max free area)

%constinteger false=0
%constinteger true=1
%constinteger core manager=2
%constinteger get=0
%constinteger release=1
%constinteger lowuser=70         ;! lowest user service number

%integer free core list,free records list,free core,flag


initialize


%cycle

   receive(mes)
   flag=0
   rep=0
%if mes_id>=lowuser %start
   flag=1                              ;! illegal request
%else
   %if mes_service=get %then get core %else release core
%finish
rep_id=mes_id                          ;! identification
rep_source=core manager
rep_flag=flag
send(rep)
```

```
   %repeat


   %routine get core

      %integer a

      %if mes_length>free core %start   ;! not enough core available
         flag=2
         %return
      %finish
      a=free core list                  ;! search the free core list
      %until a=free core list %cycle    ;! for a area large enough
         %if area(a)_length>=mes_length %start ;! first area which fits is used
            free core=free core-mes_length
            rep_start=area(a)_start
            rep_length=mes_length
            %if mes_length<area(a)_length %start          ;! if the whole area
               area(a)_start=area(a)_start+mes_length     ;! is not used then
               area(a)_length=area(a)_length-mes_length ;! reset values
            %else
               return record(a)                           ;! else return record
            %finish
            %return
         %finish
         a=area(a)_next
      %repeat
      flag=2                            ;! no free area is large enough
   %end


   %routine release core

      %integer a,prev,new,at end

      %if free core>0 %start            ;! not the only free area
         a=free core list
         at end=false
         %while area(a)_start<mes_start %and %not at end=true %cycle
            a=area(a)_next                                  ;! search free area
            %if a=free core list %then at end=true          ;! behind released area
         %repeat
         prev=area(a)_prev                                  ;! preceding free area
         %if area(prev)_start+area(prev)_length+mes_length=area(a)_start %start
            area(prev)_length=area(prev)_length+mes_length+area(a)_length
            return record(a)                          ;! goes with surrounding areas
         %else
          %if area(prev)_start+area(prev)_length=mes_start %start ;! with the
           area(prev)_length=area(prev)_length+mes_length   ;! preceding area
        %else
            %if mes_start+mes_length=area(a)_start %start
               area(a)_start=mes_start                      ;! with the
               area(a)_length=area(a)_length+mes_length     ;! following area
            %else                           ;! goes in between surrounding areas
```

```
                new=free records list     ;! get a new record
                free records list=area(free records list)_next
                area(prev)_next=new       ;! put it in between
                area(new)_prev=prev
                area(a)_prev=new
                area(new)_next=a
                area(new)_start=mes_start
                area(new)_length=mes_length
                %if free core list=a %and at end=false %c
                              %then free core list=new  ;! new first element
            %finish
        %finish
    %finish
    %else                                 ;! only area of free core
        new=free records list             ;! get a new record
        free records list=area(free records list)_next
        free core list=new
        area(new)_prev=new                ;! and fill it up
        area(new)_next=new
        area(new)_start=mes_start
        area(new)_length=mes_length
    %finish
    free core=free core+mes_length        ;! total free core

%end


%routine return record (%integer a)

    %if free core list=a %then free core list=area(a) next
    area(area(a)_prev)_next=area(a)_next      ;! surrounding
    area(area(a)_next)_prev=area(a)_prev      ;! records
    area(a)_next=free records list
    free records list=a                       ;! into the free list

%end


%routine initialize

    %externalintegerspec start user memory
    %externalintegerspec available user memory

    %integer a

    free core=available user memory           ;! total number of free blocks
    area(1)_start=start user memory           ;! first free block
    area(1)_length=free core
    area(1)_next=1
    area(1)_prev=1
    free core list=1
    %for a=2,1,max free area %cycle           ;! free records list
        area(a)=0
        area(a)_next=a+1
    %repeat
    area(max free area)_next=max free area
```

```
      free records list=2
   %end

.%endofprogram
```

```
**********************
*** file system ***
**********************

%control k'100001'

%begin

  %recordformat message (%byteinteger id,service,%integerarray name(1:3),%c
                                                    %integer par4,par5)
  %recordformat reply (%byteinteger id,source,%integer flag,par2,par3, %c
                                                        par4,par5)

%recordformat entryfm (%integer status,%integerarray name (1:3), %c
                       %integer length,%byteinteger ownap,othersap, %c
                       %integer crdate,lastacc,numacc,ofac,unused)
%constinteger numentr=47          ;! number of entries in directoryblock
%recordformat directoryblock (%integer numdblks,nextdblk,highdblk, %c
            unused,startfiles,%record(entryfm)%array entry(1:numentr))

%recordformat diskmessage (%byteinteger serv,source,service,drive, %c
             %integer diskad,%record (directoryblock) %name address, %c
             %integer length,mark)

%externalroutinespec receive (%record (message) %name mes)
%externalroutinespec send (%record (reply) %name rep)
%externalroutinespec svc (%record (diskmessage) %name diskmes)

%routinespec connect file
%routinespec disconnect file
%routinespec create file
%routinespec close file
%routinespec delete file
%routinespec rename file
%routinespec set access permissions of file
%routinespec get file names and information
%routinespec offer file
%routinespec accept file

%integerfnspec extend
%integerfnspec date
%routinespec getdblock (%integer dblk)
%routinespec writedblock
%predicatespec file exists (%integerarrayname name,%integername index)
%routinespec comprdblock
%predicatespec connected (%integername index)
%predicatespec file in virtual memory
%predicatespec permission (%integer ownperm,othperm)
%predicatespec room in virtual memory (%integer case)
%routinespec insert file information
%routinespec return file information
%routinespec insert block information
%routinespec return blocks
%byteintegerfnspec nextbyte
%routinespec putword (%integer word)
%routinespec transfer buffer (%integer service,block)
```

```
%routinespec svc to disk (%integer service,diskad,stblk,length)
%routinespec release core (%integer stblk,length)

%routinespec user login
%predicatespec identification ok

%routinespec recover
%routinespec user logout

%routinespec disconnect normal files
%routinespec disconnect base files

%routinespec initialize

%constinteger false=0
%constinteger true=1

%constinteger clock=0                   ;! services
%constinteger disk=1
%constinteger core manager=2
%constinteger scheduler=3
%constinteger login=4
%constinteger connect=40
%constinteger disconnect=41
%constinteger recov=60
%constinteger logout=61
%constinteger usminup=45                ;! lowest user service number minus
                                        ;! lowest user process number
%constinteger lowuser=70
%constinteger highuser=76
%constinteger curtime=1                 ;! for clock
%constinteger read=0                    ;! for disk
%constinteger write=1
%constinteger toaddr=2
%constinteger release=1                 ;! for core manager

%constinteger ru=0                      ;! mode constants
%constinteger rs=1
%constinteger rwu=2
%constinteger rws=3
%constinteger shared=1

%constinteger out=1                     ;! segment state
%constinteger out core=1                ;! block state
%constinteger free=0                    ;! iosegmentblock states
%constinteger fil=1
%constinteger written bit=k'100'        ;! written into bit pdr

%constinteger maxlength=250             ;! maximum file length
%constinteger tent=k'400'               ;! entry status words
%constinteger empty=k'1000'
%constinteger perm=k'2000'
%constinteger endmarker=k'4000'
%constinteger allmod=k'17'              ;! all access modes permitted
%constinteger dofset=4                  ;! first dblock at block 6
```

```
%switch service(40:49)

%record (message) mes
%record (reply) rep
%record (diskmessage) diskmes
%record (directoryblock) dblock
%record (entryfm) %name entry

%recordformat processtable entry (%integer r0,r1,r2,r3,r4,r5,sp,pc,psw, %c
              received,%byteinteger unus1,heldup,%integer unus2,rsmoni,%c
              %byteinteger asleep,unus3,%integer unus4,unus5)
%constinteger idle=k'47'
%externalrecord (processtable entry) %array %spec proctab(0:idle)

%recordformat active page registers (%integerarray par,pdr(0:7))
%constinteger hminl=6                    ;! highuser-lowuser
%externalrecord (active page registers) %array %spec userapr(0:hminl)

%recordformat segmentformat (%byteinteger fileptr,blockptr,status)
%recordformat subsegformat (%byteinteger fileptr,start,length, %c
                                                    %integer diskad)
%recordformat iosegformat (%byteintegerarray block(0:15), %c
                           %byteinteger numbfiles, copy ok, %c
                           %record(subsegformat)%array subseg(0:2))
%recordformat segmenttablef (%record(segmentformat)%array seg(0:7),%c
                             %record (iosegformat) %array ioseg(0:1),%c
                             %integer handler)
%externalrecord (segmenttablef) %array segmenttable(0:hminl)

%recordformat file format (%integerarray name(1:3),%integer length, %c
        diskad,users,%byteinteger ownap,othersap,mode,blksptr,next,prev)
%constinteger maxfile=50    ;! maximum number of files to connect
%externalrecord (file format) %array file(0:maxfile)

%recordformat block format (%integer par,pdr,stblk,diskad,file block, %c
              %byteinteger status,users,users in core,length,prev,next)
%constinteger maxblock=50   ;! maximum number of blocks to connect
%externalrecord (block format) %array block(0:maxblock)

%recordformat recovery information (%integer r5,sp,pc)
%record (recovery information) %array recov info (lowuser:highuser)

%integerarray id (lowuser:highuser)

%record (segmenttablef) %name segtable
%record (iosegformat) %name ioseg
%record (subsegformat) %name subseg

%byteintegerarray buffer (0:511)

%integer dblkincore,dblkend,highdblk,diskad,flag,full,nextpos,fblock, %c
        file length,files,freefiles,freeblocks,startup,segment,drive, %c
        startblock,length,mode,index,segind,subind,fileptr,ident

initialize
```

```
   drive=0

   %cycle

      receive(mes)
      flag=0
      rep=0

      %if 40<=mes_service<=49 %then ->service(mes_service) ;! file service
      %if mes_service=login %start
         %if mes_id<lowuser %then user login %else flag=1
         -> setrep
      %finish
      %if mes_service=recov %then recover %and ->setrep
      %if mes_service=logout %then user logout %and  %continue

      service(40): connect file
                   ->setrep

      service(41): disconnect file
                   ->setrep

      service(42): create file
                   ->setrep

      service(43): close file
                   ->setrep

      service(44): delete file
                   ->setrep

      service(45): rename file
                   ->setrep

      service(46): set access permissions of file
                   ->setrep

      service(47): get file names and information
                   ->setrep

      service(48): offer file
                   ->setrep

      service(49): accept file
                   ->setrep

      setrep: rep_id=mes_id                ;! user identification
              rep_source=mes_service
              rep_flag=flag
              send(rep)

   %repeat


   %routine connect file
```

```
segtable==segmenttable(mes_id-lowuser)
segment=mes_par4&7                              ;! segment number
startblock=mes_par5&k'777'                      ;! start block in file
length=mes_par5>>9&k'77'                        ;! requested length
%if segment=0 %then mode=mes_par4>>8&1 %elsestart
%if segment=1 %then mode=rwu %c
%else mode=mes_par4>>8&3 ; %finish
%if %not connected(fileptr) %start              ;! file not yet connected
   %if %not file exists(mes_name,index) %or %c
       %not permission(dblock_entry(index)_ownap, %c
                       dblock_entry(index)_othersap) %c
      %then flag=1 %and %return                 ;! no permission
   %if startblock>=dblock_entry(index)_length %c
      %then flag=2 %and %return                 ;! block does not exist
   %if length=0 %c
      %then length=dblock_entry(index)_length   ;! whole file
   %if startblock+length>dblock_entry(index)_length %c
      %then length=dblock_entry(index)_length-startblock
   %if %not room in virtual memory(0) %c
      %then flag=4 %and %return
   insert file information
   dblock_entry(index)_lastacc=date             ;! update date last access
   dblock_entry(index)_numacc=  %c
      dblock_entry(index)_numacc+1              ;! and number of accesses
   writedblock
%else
   %if startblock>=file(fileptr)_length %c
      %then flag=2 %and %return                 ;! block does not exist
   %if length=0 %c
      %then length=file(fileptr)_length   ;! whole file
%if startblock+length>file(fileptr)_length %c
   %then length=file(fileptr)_length-startblock
%if %not file in virtual memory %start ;! not connected by process
   %if %not permission(file(fileptr)_ownap,file(fileptr)_othersap) %c
      %then flag=1 %and %return                 ;! no permission
   %if mode#file(fileptr)_mode %or file(fileptr)_mode&shared=0 %c
      %then flag=3 %and %return                 ;! mode conflict
   %if %not room in virtual memory(0) %c
      %then flag=4 %and %return
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                 ;! to get dblock into core
   file(fileptr)_users=file(fileptr)_users+1
   dblock_entry(index)_lastacc=date             ;! update date last access
   dblock_entry(index)_numacc=  %c
      dblock_entry(index)_numacc+1              ;! and number of accesses
   writedblock
%else
      %if mode#file(fileptr)_mode %c
         %then flag=5 %and %return
      %if %not room in virtual memory(1) %c
         %then flag=4 %and %return
      %if segind>=6 %or (segind<=1 %and subind=-1) %c
         %then flag=6 %and %return ;! not allowed to disconnect this file
      return blocks
   %finish
%finish
```

```
      insert block information
      %if segment<=1 %and startup=false %start  ;! subsegment
         svc to disk(read,subseg_diskad, %c
                     block(segtable_seg(segment)_blockptr)_stblk + %c
                     subseg_start,subseg_length)
         rep_par2=segment*8192+subseg_start*512 ;! virtual address
         rep_par3=length*512                    ;! connected length
      %else                                     ;! normal segment
         %if startup=false %start
            rep_id=scheduler                    ;! message
            rep_source=2                        ;! to scheduler
            rep_flag=mes_id                     ;! process identification
            send(rep)
            rep_par2=segment*8192               ;! virtual address
            rep_par3=length*512                 ;! connected length
         %finish
      %finish
      rep_par4=file(fileptr)_length             ;! length of file

%end


%routine disconnect file

      segtable==segmenttable(mes_id-lowuser)
      %if %not file in virtual memory %then flag=1 %and %return
      %if segind>=6 %or (segind<=1 %and subind=-1) %c
         %then flag=2 %and %return  ;! not allowed to disconnect this file
      return blocks
      return file information
      rep_id=scheduler                          ;! send message
      rep_source=3                              ;! to scheduler
      send(rep)

%end


%routine create file

      %recordformat large (%integer length,dblock)

      %record (large) largest,seclargest    ;! to find largest entries
      %integer i,j,d

      %if mes_id#lowuser %then %c
         mes_name(3)=mes_name(3)>>1//800*1600+id(mes_id) ;! set owner name
      %if mes_par4>maxlength %or mes_par4+1<0 %then flag=2 %and %return
      %if file exists(mes_name,i) %then flag=1 %and %return
      getdblock(1)                                  ;! read first dblock
      highdblk=dblock_highdblk                      ;! remember highest
      largest_length=0;seclargest_length=0
      %cycle
         comprdblock                                ;! compress dblock
         i=1;entry==dblock_entry(1)
         %until entry_status&endmarker#0 %cycle
            %if entry_status&empty#0 %start         ;! empty entry
```

```
%if mes_par4<=0 %start                    ;! look for largest
   %if entry_length>largest_length %start;! new largest
      seclargest=largest                  ;! and 2 nd largest
      largest_length=entry_length
      largest_dblock=dblkincore           ;! remember dblock
   %else
      %if entry_length>seclargest_length %start ;! new 2 nd
         seclargest_length=entry_length   ;! largest
         seclargest_dblock=dblkincore     ;! remember dblock
      %finish
   %finish
%else
%if entry_length>=mes_par4 %start   ;! request satisfied
   %if dblkend+3>numentr %start     ;! have to open new
                                    ;! directoryblock
      d=dblkincore                  ;! remember dblock
      %if extend=0 %start           ;! extend success
         getdblock(d)               ;! restart search
         comprdblock
         i=1;entry==dblock_entry(1)
         %continue                  ;! at a good place
      %finishelsestart
         flag=5;%return             ;! directory overflow
      %finish
   %else
      entry_length=entry_length-mes_par4 ;! rest entry
      %if i-1>0 %and dblock_entry(i-1)_status&tent#0 %start
         dblkend=dblkend+1              ;! previous entry
      %for j=dblkend,-1,i+1 %cycle ;! is tentative:
         dblock_entry(j)=dblock_entry(j-1) ;! have
      %repeat                       ;! to insert
      dblock_entry(i)_status=empty ;! empty
      dblock_entry(i)_length=0     ;! of length 0
      i=i+1;entry==dblock_entry(i)
%finish
%for j=dblkend+1,-1,i+1 %cycle       ;! make room
   dblock_entry(j)=dblock_entry(j-1) ;! for new entry
%repeat
entry_status=tent        ;! tentative
entry_name(1)=mes_name(1)
entry_name(2)=mes_name(2)
entry_name(3)=mes_name(3)
entry_length=mes_par4
entry_ownap=allmod       ;! owner all permissions,
entry_othersap=0         ;! others none
entry_crdate=date        ;! creation date
entry_lastacc=0          ;! last access
entry_numacc=0           ;! number accesses
entry_ofac=0             ;! offer/accept
entry_unused=0
writedblock              ;! write back dblock
rep_par2=mes_par4        ;! length
%return                  ;! success
      %finish
   %finish
%finish
```

```
      %finish
      i=i+1;entry==dblock_entry(i)          ;! try next entry
   %repeat
   %if dblock_nextdblk=0 %start             ;! this was last dblock
      %if mes_par4>0 %start                 ;! when asking for
         flag=3                             ;! specific length
         %return                            ;! then not enough space
      %finishelsestart                      ;! convert to specific one
         %if mes_par4=0 %start              ;! asking for half largest
            %if largest_length//2>seclargest_length %start
               mes_par4=largest_length//2   ;! or 2 nd largest
               d=largest_dblock
            %else
               mes_par4=seclargest_length
               d=seclargest_dblock
            %finish
         %else                              ;! asking for largest
            mes_par4=largest_length
            d=largest_dblock
         %finish
         %if mes_par4=0 %then flag=4 %and %return ;! no more room at all
         %if mes_par4>maxlength %then mes_par4=maxlength
                                            ;! give only the maximum
         getdblock(d)                       ;! restart search
         %continue                          ;! at right dblock
      %finish
   %else
      getdblock(dblock_nextdblk)            ;! get next dblock
   %finish
%repeat

%end


%routine close file

   %integer index,allen

   %if mes_par4<=0 %then flag=5 %and %return    ;! illegal length
   %if mes_id#lowuser %then %c
      mes_name(3)=mes_name(3)>>1//800*1600+id(mes_id) ;! set owner name
   %if connected(index) %then flag=2 %and %return  ;! file connected
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                 ;! file does not exist
   %if dblock_entry(index)_status&perm#0 %c
      %then flag=3 %and %return                 ;! file already closed
   allen=dblock_entry(index)_length             ;! allocated length
   %if allen<mes_par4 %then flag=4 %and %return ;! asking more
   dblock_entry(index)_length=mes_par4          ;! requested length
   dblock_entry(index)_status=perm              ;! status permanent
   dblock_entry(index+1)_length= %c
   dblock_entry(index+1)_length+allen-mes_par4  ;! reclaim unused space
   comprdblock                                  ;! compress the dblock
   writedblock                                  ;! and write it back

%end
```

```
%routine delete file

   %integer index

   %if mes_id#lowuser %then %c
      mes_name(3)=mes_name(3)>>1//800*1600+id(mes_id) ;! set owner name
   %if connected(index) %then flag=2 %and %return     ;! file connected
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                       ;! file does not exist
   dblock_entry(index)_status=empty                   ;! make entry empty
   comprdblock                                        ;! compress dblock
   writedblock                                        ;! and write it back

%end


%routine rename file

   %integerarray newname (1:3)
   %integer index,d

   newname(1)=mes_name(3)>>8!mes_par4<<8               ;! new name
   newname(2)=mes_par4>>8!mes_par5<<8
   newname(3)=(mes_par5>>8)*1600+id(mes_id)            ;! set owner name
   mes_name(3)=(mes_name(3)&k'377')*1600+id(mes_id)
   %if connected(index) %then flag=3 %and %return ;! file connected
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                        ;! file does not exist
   d=dblkincore                                        ;! save dblock number
   %if file exists(newname,index) %c
      %then flag=2 %and %return                        ;! already file with new name
   getdblock(d)                                        ;! get dblock with entry of file
   dblock_entry(index)_name(1)=newname(1)
   dblock_entry(index)_name(2)=newname(2)
   dblock_entry(index)_name(3)=newname(3)
   writedblock                                         ;! write dblock back

%end


%routine set access permissions of file

   %integer index

   mes_name(3)=mes_name(3)>>1//800*1600+id(mes_id) ;! set owner name
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                        ;! file does not exist
   %if dblock_entry(index)_ofac#0 %c
      %then flag=1 %and %return                        ;! file offered
   dblock_entry(index)_ownap=mes_par4&k'17'            ;! permissions owner
   dblock_entry(index)_othersap=mes_par4>>8&k'17'      ;! permissions others
   writedblock                                         ;! write dblock back

%end
```

```
%routine get file names and information

   %integer nextdblk,ownerid,index,i

   ownerid=id(mes_id)
   mes_name(3)=mes_name(3)>>1//800*1600+ownerid   ;! set owner name
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                   ;! dfile does not exist
   filelength=dblock_entry(index)_length
   fblock=0;nextpos=0;full=false
   nextdblk=1
   %until nextdblk=0 %cycle
      getdblock(nextdblk)                         ;! read directory block
      i=1;entry==dblock_entry(1)
      %until entry_status&endmarker#0 %cycle
         %if (entry_status&perm#0 %or entry_status&tent#0) %c
           %and entry_name(3)-entry_name(3)>>1//800*1600=ownerid %start
            putword(entry_name(1))                ;! file belongs to user
            putword(entry_name(2))
            putword(entry_name(3))                ;! put information
            putword(entry_length)                 ;! into dfile
            putword(entry_ownap!entry_othersap<<8)
            putword(entry_crdate)
            putword(entry_lastacc)
            putword(entry_numacc)
            putword(entry_ofac)
            %if flag#0 %then %return              ;! dfile too short
         %finish
         i=i+1;entry==dblock_entry(i)
      %repeat
      nextdblk=dblock_nextdblk                    ;! next dblock
   %repeat
   putword(0)
   %if flag=0 %then transfer buffer(write,fblock) ;! write last block

%end


%routine offer file

   %integer index

   mes_name(3)=mes_name(3)>>1//800*1600+id(mes_id) ;! set owner name
   %if connected(index) %then flag=2 %and %return  ;! file connected
   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                    ;! file does not exist
   dblock_entry(index)_ofac=mes_par4               ;! ident new owner
   %if mes_par4#0 %start                           ;! file offered
      dblock_entry(index)_ownap=0                  ;! no one has permission
      dblock_entry(index)_othersap=0               ;! to use file
   %else                                           ;! offer revoked
      dblock_entry(index)_ownap=allmod             ;! only owner
   %finish                                         ;! permission
   writedblock                                     ;! back to disk
```

```
%end

%routine accept file

   %integer index

   %if %not file exists(mes_name,index) %c
      %then flag=1 %and %return                    ;! file does not exist
   %if dblock_entry(index)_ofac#id(mes_id) %c
      %then flag=1 %and %return                    ;! not offered to user
   dblock_entry(index)_name(3)=  %c
      dblock_entry(index)_name(3)>>1//800*1600+id(mes_id) ;! set new owner
   dblock_entry(index)_ownap=allmod                ;! new owner all perm.
   dblock_entry(index)_ofac=0                      ;! offer/accept word
   writedblock                                     ;! back to disk

%end

%integerfn extend

   %integer half,newstb,newend,savest,savelink,i

   %if highdblk=dblock_numdblks %then %result=1    ;! directory overflow
   half=dblkend//2+1
   newstb=dblock_startfiles                        ;! startblock
   %for i=1,1,half-1 %cycle
      newstb=newstb+dblock_entry(i)_length         ;! in new dblock
   %repeat
   newend=half
   %while dblock_entry(newend)_status&perm=0 %and %c
         dblock_entry(newend)_status&tent=0 %cycle  ;! look for first
      newstb=newstb+dblock_entry(newend)_length     ;! permanent or
      newend=newend+1                               ;! tentative file
   %repeat
   savest=dblock_entry(newend)_status              ;! new end position
   dblock_entry(newend)_status=endmarker           ;! shortened dblock
   savelink=dblock_nextdblk
   dblock_nextdblk=highdblk+1
   writedblock                                     ;! write it back
   dblock_nextdblk=savelink
   dblock_startfiles=newstb
   dblock_entry(newend)_status=savest
   %for i=1,1,dblkend-newend+1 %cycle              ;! move rest of entries
      dblock_entry(i)=dblock_entry(newend+i-1)     ;! to top of
   %repeat                                         ;! new dblock
   dblkincore=highdblk+1                           ;! new dblock number
   writedblock                                     ;! write new dblock
   getdblock(1)                                    ;! get first dblock
   dblock_highdblk=dblock_highdblk+1               ;! highest dblock in use
   writedblock                                     ;! write first dblock
   %result=0                                       ;! success

%end
```

```
%integerfn date

   %recordformat clmesf (%byteinteger serv,source,%integer service,msec,%c
                                                    minsec,dayhour,yrmon)
   %record (clmesf) clm
   %externalroutinespec svc (%record (clmesf) %name clm)

   clm_serv=clock
   clm_source=mes_service
   clm_service=curtime
   svc(clm)                                          ;! get current time
   %result=((clm_yrmon>>8&k'377'-72)&k'37')!(clm_yrmon<<10&k'76000') %c
               !(clm_dayhour>>3&k'1740')              ;! put it in right format
%end


%routine getdblock (%integer dblk)

   %if dblk#dblkincore %start                        ;! dblock not in core
      diskmes_serv=disk
      diskmes_source=mes_service
      diskmes_service=read!toaddr                    ;! read to address
      diskmes_drive=drive
      diskmes_diskad=2*dblk+dofset                   ;! start block on disk
      diskmes_address==dblock                        ;! start address dblock
      diskmes_length=2                               ;! 2 blocks
      svc(diskmes)
      dblkincore=dblk                                ;! remember dblock in core
   %finish
%end


%routine writedblock          drive number??

   diskmes_serv=disk
   diskmes_source=mes_service
   diskmes_service=write!toaddr                      ;! write to address
   diskmes_diskad=2*dblkincore+dofset                ;! start block on disk
   diskmes_address==dblock                           ;! start address dblock
   diskmes_length=2                                  ;! 2 blocks
   svc(diskmes)

%end


%predicate file exists (%integerarrayname name,%integername index)

%integer nextdblk,i

nextdblk=1
%until nextdblk=0 %cycle
   getdblock(nextdblk)                               ;! read directory block
   diskad=dblock_startfiles                          ;! start block files
   i=1;entry==dblock_entry(1)
```

```
      %until entry_status&endmarker#0 %cycle
         %if (entry_status&perm#0 %or entry_status&tent#0)%and %c
              entry_name(1)=name(1) %and %c
              entry_name(2)=name(2) %and %c
              entry_name(3)=name(3) %start
            index=i
            %true                                   ;! file exists
         %finish
         diskad=diskad+entry_length                 ;! increase start block
         i=i+1;entry==dblock_entry(i)
      %repeat
      nextdblk=dblock_nextdblk                      ;! try next dblock
   %repeat
      %false                        ;! file does not exist

%end


%routine comprdblock

   %integer i,j

   i=1;entry==dblock_entry(1)
   %until entry_status&endmarker#0 %cycle
      %if entry_status&empty#0 %start                       ;! empty entry
         %if dblock_entry(i+1)_status&empty#0 %start
            entry_length=entry_length+dblock_entry(i+1)_length
                                                ! combine consecutive empties
            j=i+1
            %until dblock_entry(j)_status&endmarker#0 %cycle
               dblock_entry(j)=dblock_entry(j+1)        ;! compress dblock
               j=j+1
            %repeat
            %continue
         %finish
         %if entry_length=0 %and i>1 %c
           %and dblock_entry(i-1)_status&perm#0 %start
            ! destroy empty entry of length 0 following permanent entry
            j=i
            %until dblock_entry(j)_status&endmarker#0 %cycle
               dblock_entry(j)=dblock_entry(j+1)        ;! compress dblock
               j=j+1
            %repeat
            %continue
         %finish
      %finish
      i=i+1;entry==dblock_entry(i)                      ;! next entry
   %repeat
      dblkend=i                                         ;! end of dblock

%end

%predicate connected (%integername index)

   %integer ptr
```

```
ptr=files                                          ;! pointer to files
%while ptr#0 %cycle
   %if file(ptr)_name(1)=mes_name(1) %andc
       file(ptr)_name(2)=mes_name(2) %andc
       file(ptr)_name(3)=mes_name(3) %start
          index=ptr                                ;! pointer to information
          %true                                    ;! of connected file
   %finish
   ptr=file(ptr)_next
%repeat
%false                          ;! file not connected

%end


%predicate file in virtual memory

   %integer fptr,i,j

   %for i=0,1,1 %cycle                             ;! look in io segments
      %for j=0,1,2 %cycle                          ;! subsegments
         fptr=segtable_ioseg(i)_subseg(j)_fileptr
         %if fptr#0 %andc
             file(fptr)_name(1)=mes_name(1) %andc
             file(fptr)_name(2)=mes_name(2) %andc
             file(fptr)_name(3)=mes_name(3) %start
                segind=i                           ;! indicates segment
                subind=j                           ;! indicates subsegment
                ioseg==segtable_ioseg(i)
                subseg==ioseg_subseg(j)
                fileptr=fptr
                %true
         %finish
      %repeat
   %repeat
   %for i=0,1,7 %cycle                             ;! look in other segments
      fptr=segtable_seg(i)_fileptr
      %if fptr#0 %andc
          file(fptr)_name(1)=mes_name(1) %andc
          file(fptr)_name(2)=mes_name(2) %andc
          file(fptr)_name(3)=mes_name(3) %start
             segind=i                              ;! indicates segment
             subind=-1                             ;! for check
             fileptr=fptr
             %true
      %finish
   %repeat
   %false                                          ;! file not connected
                                                   ;! by process

%end


%predicate permission (%integer ownperm,otherperm)
```

```
      %constbyteintegerarray modebit(0:3)=1,2,4,8
      %integer mb

      mb=modebit(mode)
      %if mes_name(3)-mes_name(3)>>1//800*1600=id(mes_id) %start
         %if ownperm&mb#0 %then %true              ;! user is ownner
      %else
         %if otherperm&mb#0 %then %true            ;! user not owner
      %finish
      %false

   %end

%predicate room in virtual memory (%integer case)
                        ! case=0 - no room released
                        ! case=1 - room released

   %integer blk,stblk,s

   %if segment<=1 %and startup=false %start              ;! io segment
      %if case=0 %and segtable_ioseg(segment)_numbfiles=3 %then %false
      stblk=mes_par4>>3&k'17'                    ;! startblock in segment
      %for blk=stblk,1,stblk+length-1 %cycle
          %if segtable_ioseg(segment)_block(blk)#free %and %c
               %not ( case=1 %and %c
                      subseg_start<=blk<subseg_start+subseg_length) %c
          %then %false
      %repeat
   %else                                                ;! other segment
      %for s=segment,1,segment+(length-1)//16 %cycle
          %if segtable_seg(s)_fileptr#0 %and %not (case=1 %and %c
            segtable_seg(s)_fileptr=segtable_seg(segind)_fileptr) %c
          %then %false
      %repeat
   %finish
   %true

%end


%routine insert file information

   %integer new

   new=freefiles                                    ;! first element freelist
   freefiles=file(freefiles)_next                   ;! reset head cell freelist
   %if files#0 %then file(files)_prev=new
   file(new)_prev=0                                 ;! no previous file on list
   file(new)_next=files
   files=new                                        ;! put file at head of list
   file(new)_name(1)=mes_name(1)
   file(new)_name(2)=mes_name(2)
   file(new)_name(3)=mes_name(3)
   file(new)_length=dblock_entry(index)_length
   file(new)_diskad=diskad                          ;! start block on disk
```

```
   file(new)_users=1
   file(new)_ownap=dblock_entry(index)_ownap ;! owner access permission
   file(new)_othersap=dblock_entry(index)_othersap  ;! others access
   file(new)_mode=mode
   file(new)_blksptr=0                              ;! no blocks connected yet
   %if segment>1 %or startup=true %c
      %then segtable_seg(segment)_fileptr=new
   fileptr=new

%end
```

```
%routine return file information

   file(fileptr)_users=file(fileptr)_users-1
   %if file(fileptr)_users=0 %start             ;! last user of file
      %if file(fileptr)_next#0 %c
         %then file(file(fileptr)_next)_prev=file(fileptr)_prev
      %if files=fileptr %start                  ;! first element
         files=file(fileptr)_next
      %else                                     ;! in between
         file(file(fileptr)_prev)_next=file(fileptr)_next
      %finish
      file(fileptr)_next=freefiles              ;! record to freelist
      freefiles=fileptr
   %finish

%end
```

```
%routine insert block information

   %integer ssin,s,total,len,blk,new,first,pdr

   %if segment<=1 %and startup=false %start  ;! io segment
      ioseg==segtable_ioseg(segment)
      ssin=0
      %while ioseg_subseg(ssin)_fileptr#0 %cycle
         ssin=ssin+1                                ;! search unused
      %repeat                                       ;! subsegment
      subseg==ioseg_subseg(ssin)
      subseg_fileptr=fileptr
      subseg_start=(mes_par4>>3)&k'17'
      subseg_length=length
      subseg_diskad=file(fileptr)_diskad+startblock
      %for blk=subseg_start,1,subseg_start+subseg_length-1 %cycle
         ioseg_block(blk)=fil                       ;! indicate file
      %repeat
      ioseg_numbfiles=ioseg_numbfiles+1         ;! subsegment used
      ioseg_copy ok=false                       ;! disk copy not ok
   %else                                        ;! other segment
      s=segment                                 ;! from this segment
      total=length                              ;! onwards
      %until total=0 %cycle
         %if total>=16 %then len=16 %else len=total
         total=total-len
```

```
        blk=file(fileptr)_blksptr               ;! connected blocks
        %while (blk#0) %and (block(blk)_file block#startblock %or %c
                             block(blk)_length#len) %cycle
           blk=block(blk)_next
        %repeat
        %if blk#0 %start                        ;! block already connected
           block(blk)_users=block(blk)_users+1
           pdr=block(blk)_pdr
           segtable_seg(s)_blockptr=blk
        %else
           new=freeblocks                       ;! first element
           freeblocks=block(new)_next           ;! from freelist
           first=file(fileptr)_blksptr          ;! first element
           %if first#0 %then block(first)_prev=new  ;! blocks list
           block(new)_prev=0                    ;! new one first one
           block(new)_next=first                ;! on list
           file(fileptr)_blksptr=new
           block(new)_users=1                   ;! number of users
           %if mode<2 %then pdr=2 %else pdr=6 ;! page descriptor
           pdr=pdr!(len*8-1)<<8                 ;! register
           block(new)_pdr=pdr
           block(new)_diskad=file(fileptr)_diskad+startblock
           block(new)_file block=startblock
           block(new)_length=len
           block(new)_status=out core           ;! not yet in core
           block(new)_users in core=0
           segtable_seg(s)_blockptr=new
        %finish
        userapr(mes_id-lowuser)_pdr(s)=pdr
        startblock=startblock+len               ;! startblock next one
           segtable_seg(s)_fileptr=fileptr      ;! set file pointer
           segtable_seg(s)_status=out           ;! set segment status
           s=s+1
        %repeat
     %finish

%end


%routine return blocks

     %integer blk,fileptr,s

     %if segind<=1 %and startup=false %start        ;! io segment
        %if segind=1 %then svc to disk(write,subseg_diskad, %c
              block(segtable_seg(segind)_blockptr)_stblk+subseg_start, %c
              subseg_length)
        %for blk=subseg_start,1,subseg_start+subseg_length-1 %cycle
           ioseg_block(blk)=free                    ;! release blocks
        %repeat
        subseg_fileptr=0                            ;! release subsegment
        ioseg_numbfiles=ioseg_numbfiles-1
     %else                                          ;! other segment
        fileptr=segtable_seg(segind)_fileptr        ;! return all connected
        s=segind                                    ;! blocks from this file
        %until segtable_seg(s)_fileptr#fileptr %cycle
```

```
        segtable_seg(s)_fileptr=0
        blk=segtable_seg(s)_blockptr                ;! points to block
        block(blk)_pdr=block(blk)_pdr!userapr(mes_id-lowuser)_pdr(s)
                                                    ;! save written into bit
        userapr(mes_id-lowuser)_pdr(s)=0
        block(blk)_users in core=block(blk)_users in core-1
        %if block(blk)_users in core=0 %start  ;! last user in core
           block(blk)_status=out core
           %if block(blk)_pdr&written bit#0 %start  ;! has to be saved
              block(blk)_pdr=block(blk)_pdr&(\written bit)
              svc to disk(write,block(blk)_diskad,block(blk)_stblk, %c
                                                    block(blk)_length)
           %finish
           release core(block(blk)_stblk,block(blk)_length)
        %finish
        block(blk)_users=block(blk)_users-1    ;! last user
        %if block(blk)_users=0 %start
           %if block(blk)_next#0 %c
              %then block(block(blk)_next)_prev=block(blk)_prev
           %if file(fileptr)_blksptr=blk %start  ;! release block
              file(fileptr)_blksptr=block(blk)_next
           %else
              block(block(blk)_prev)_next=block(blk)_next
           %finish
           block(blk)_next=freeblocks               ;! back to
           freeblocks=blk                           ;! the free list
        %finish
        s=s+1
     %repeat
  %finish
%end


%byteintegerfn nextbyte

   %integer byte

   %if nextpos=0 %start                     ;! read a block
      transfer buffer(read,fblock)
      fblock=fblock+1
   %finish
   byte=buffer(nextpos)                     ;! get a byte
   nextpos=(nextpos+1)&511                  ;! next position
   %result=byte

%end


%routine putword(%integer word)

   %if full=true %then flag=2 %and %return   ;! file full
   buffer(nextpos)=word&k'377'               ;! put word into buffer
   nextpos=nextpos+1
   buffer(nextpos)=word>>8
   nextpos=nextpos+1
```

```
    %if nextpos=512 %start                          ;! file buffer full
       nextpos=0
       transfer buffer(write,fblock)                ;! write block
       fblock=fblock+1
       %if fblock>=filelength %then full=true       ;! file full
    %finish

%end


%routine transfer buffer(%integer service,block)

    %recordformat diskmessage (%byteinteger serv,source,service,drive, %c
        %integer diskad,%byteintegerarrayname address,%integer length,mark)
    %externalroutinespec svc (%record (diskmessage) %name diskmes)
    %record (diskmessage) diskmes

    diskmes_serv=disk
    diskmes_source=mes_service
    diskmes_service=service!toaddr                  ;! write to address
    diskmes_drive=drive
    diskmes_diskad=diskad+block                     ;! start block on disk
    diskmes_address==buffer                         ;! start address buffer
    diskmes_length=1                                ;! 1 block
    svc(diskmes)

%end


%routine svc to disk (%integer service,diskad,stblk,length)

    %recordformat diskmessage (%byteinteger serv,source,service,drive,%c
                                        %integer diskad,stblk,length,mark)
    %record (diskmessage) diskmes

    diskmes_serv=disk
    diskmes_source=mes_service
    diskmes_service=service                         ;! parameters
    diskmes_drive=drive
    diskmes_diskad=diskad
    diskmes_stblk=stblk
    diskmes_length=length
    svc(diskmes)

%end


%routine release core (%integer stblk,length)

    %recordformat message (%byteinteger serv,source,%integer service, %c
                                                stblk,length,par4,par5)
    %record (message) mes

    mes_serv=core manager
    mes_source=disconnect
    mes_service=release                             ;! release
```

```
      mes_stblk=stblk                                  ;! from stblk onwards
      mes_length=length                                ;! length blocks
      svc(mes)

  %end


**************
*** login ***
**************

  %routine user login

  ! When a user process is started, the files seg0,seg1,seg6 and seg7
  ! are connected in ru,rwu,rwu and rs mode in the segments 0,1,6 and 7.
  ! Seg and segnum contain the RADIX 50 constants for these names,
  ! basepar4 contains the other relevant parameters for the connect.
  ! System is the owner id of the system manager, scratch and object
  ! are extensions.
  %constinteger seg=k'073617'
  %constintegerarray segnum(1:4)=k'163500',k'140700',k'160400',k'135600'
  %constintegerarray basepar4(1:4)=k'407',k'1001',k'1006',0
  %constinteger system=k'1421'
  %constinteger scratch=k'73300'
  %constinteger object=k'56700'

  %integer unum,consol,i

  %if %not identification ok %then flag=1 %and %return ;! not admitted
  %for unum=lowuser,1,highuser %cycle
     %if id(unum)=ident %then flag=2 %and %return      ;! already logged in
  %repeat
  %if ident=system %start                              ;! system manager
     unum=lowuser
  %else
     %for unum=lowuser+1,1,highuser %cycle             ;! search free
        %if id(unum)=0 %then %exit                     ;! service number
     %repeat
  %finish
  %if id(unum)=0 %start                                ;! found
     id(unum)=ident
  %else                                                ;! not found:
     flag=4 %and %return                               ;! too many users
  %finish
  startup=true                                         ;! indicate startup
  consol=mes_id                                        ;! console service
  mes_id=unum                                          ;! service number
  %for i=1,1,4 %cycle
     mes_name(1)=seg                                   ;! connect the
     mes_name(2)=segnum(i)                             ;! base files
     mes_par4=basepar4(i)
     %if i#1 %start
        mes_name(3)=scratch+ident                      ;! scratch file
        mes_par5=0
     %else
        mes_name(3)=object+system                      ;! cli+perm
```

```
      mes_par5=1                                    ;! starts at block 1
   %finish
   connect file
   %if flag#0 %start                                ;! connect error
      disconnect base files                         ;! disconnect
      id(unum)=0                                    ;! already connected
      startup=false                                 ;! base files
      mes_id=consol                                 ;! reply to console
      flag=3
      %return
   %finish
%repeat
startup=false
   rep_id=unum                                      ;! send message
   rep_source=login                                 ;! to user
   rep_flag=consol                                  ;! console id
   rep_par2=ident                                   ;! user id
   rep_par4=0
   send(rep)
   rep_id=scheduler                                 ;! send message
   rep_source=2                                     ;! to scheduler
   rep_flag=unum                                    ;! service number
   send(rep)
   rep_par2=unum                                    ;! service number
   mes_id=consol                                    ;! reply to console

%end


%predicate identification ok

%constintegerarray passwdfile (1:3)=k'62073',k'75134',k'63421'

%constbyteintegerarrayname memory=0

%integer index,adr,char,namelength,found,ch,count,first,second

%if %not file exists(passwdfile,index) %then %false
fblock=0;nextpos=0
adr=mes_name(3)+mes_par4                            ;! address of given name
found=false
namelength=nextbyte                                 ;! length of first name
%while namelength#0 %cycle
   char=0
   %if namelength=memory(adr) %start                ;! compare lengths
      found=true
   %for char=1,1,namelength %cycle                  ;! compare characters
      %if nextbyte#memory(adr+char) %c
         %then found=false %and %exit
   %repeat
%finish
%for count=char+1,1,namelength %cycle
   ch=nextbyte                                      ;! rest of characters
%repeat
%if found=true %then %exit
%for count=1,1,6 %cycle                             ;! password and id
```

```
            ch=nextbyte
         %repeat
         namelength=nextbyte
      %repeat
      %if found=true %start
         adr=mes_name(3)+mes_par5                    ;! address of password
         first=9681                                  ;! first part encoded password
         %for char=1,1,memory(adr)//2 %cycle
            first=(first+13*memory(adr+char))!!73519
         %repeat
         second=1869                                 ;! second part
         %for char=memory(adr)//2+1,1,memory(adr) %cycle
            second=(second+17*memory(adr+char))!!91537
         %repeat
         %if nextbyte=first&k'377' %and nextbyte=first>>8 %and %c
           nextbyte=second&k'377' %and nextbyte=second>>8 %start
            ident=(nextbyte-k'140')*40+nextbyte-k'140'   ;! ident in RAD 50
            %true                                    ;! success
         %finish
      %finish
      %false

   %end


!***************
!*** recover ***
!***************

   %routine recover

   %integer procnum

   %if mes_name(1)&1=0 %start                        ;! set recover registers
      recov info(mes_id)_r5=mes_name(2)
      recov info(mes_id)_sp=mes_name(3)
      recov info(mes_id)_pc=mes_par4
   %else
      procnum=mes_id-usminup
      rep_par2=proctab(procnum)_r5                   ;! old r5
      rep_par3=proctab(procnum)_sp                   ;! old sp
      rep_par4=proctab(procnum)_pc                   ;! old pc
      rep_par5=recov info(mes_id)_r5                 ;! new r5
      proctab(procnum)_sp=recov info(mes_id)_sp      ;! new sp
      proctab(procnum)_pc=recov info(mes_id)_pc      ;! new pc
      proctab(procnum)_asleep=recov
      segtable==segmenttable(mes_id-lowuser)
      %if mes_name(2)#k'177777' %c
         %then disconnect normal files               ;! clean virtual memory
         flag=mes_name(2)                            ;! error number
      %finish

   %end

!***************
```

```
!*** logout ***
!**************

  %routine user logout

     segtable==segmenttable(mes_id-lowuser)
     disconnect normal files
     startup=true
     disconnect base files
     startup=false
     segmenttable(mes_id-lowuser)=0
     id(mes_id)=0
     recov info(mes_id)=0
     proctab(mes_id-usminup)_asleep=login        ;! process asleep on login
     rep_id=scheduler                            ;! message to scheduler
     rep_source=7
     rep_flag=mes_id
     send(rep)

%end


%routine disconnect normal files

   %integer fptr,i,j

   %for i=0,1,1 %cycle                           ;! disconnect files
      %for j=0,1,2 %cycle                        ;! from io segment
         fptr=segtable_ioseg(i)_subseg(j)_fileptr
         %if fptr#0 %start
            segind=i
            subind=j
            ioseg==segtable_ioseg(i)
            subseg==ioseg_subseg(j)
            fileptr=fptr
            return blocks
            return file information
         %finish
      %repeat
%repeat
%for i=2,1,5 %cycle                              ;! disconnect files
   fptr=segtable_seg(i)_fileptr                  ;! from other segments
   %if fptr#0 %start
      segind=i
      fileptr=fptr
      return blocks
      return file information
   %finish
   %repeat
   rep_id=scheduler                              ;! message to scheduler
   rep_source=3
   send(rep)

%end
```

```
%routine disconnect base files

   %integer fptr,i,j

   %for i=0,6,6 %cycle                              ;! base files in
      %for j=i,1,i+1 %cycle                         ;! segments 0,1,6,7
         fptr=segtable_seg(j)_fileptr
         %if fptr#0 %start
            fileptr=fptr
            segind=j
            return blocks
            return file information
            %finish
         %repeat
   %repeat

%end


%routine initialize

   %integer i

   %for i=0,1,hminl %cycle
      segmenttable(i)=0                             ;! segment table
   %repeat
   %for i=0,1,maxblock %cycle
      block(i)=0                                    ;! blocks list
      block(i)_next=i-1
   %repeat
   block(0)_next=0
   freeblocks=maxblock
   %for i=0,1,maxfile %cycle
      file(i)=0                                     ;! files list
      file(i)_next=i-1
   %repeat
   file(0)_next=0
   freefiles=maxfile
   files=0                                          ;! no files yet
   %for i=lowuser,1,highuser %cycle
      id(i)=0                                       ;! identification
      recov info(i)=0                               ;! recovery information
   %repeat
   dblkincore=0                                     ;! no dblock in core
   startup=false

%end

endofprogram
```

```
**************************
*** console handler ***
**************************

%control k'100001'

%begin

  %recordformat message (%byteinteger id,consid,%integer service,par2,par3, %c
                                                              par4,par5)
  %recordformat reply (%byteinteger id,source,%integer flag,par2,par3,par4, %c
                                                                 par5)
  %recordformat login message (%byteinteger id,source,%integer flag,servnum,%c
                          %byteintegerarrayname buffer,%integer name,passwd)

%record (message) mes
%record (reply) rep
%record (login message) logmes

%recordformat console registers (%integer istatus,ibuffer,ostatus,obuffer)
%recordformat console descriptor (%record(console registers)%name address,%c
              %integer ident,max line width,delete char,sema number,%c
              dmasignal,mode,output address,output map register, %c
              number of bytes,line width,state,dma output status,user)

%constinteger highest console=5
%externalrecord (console descriptor) %arrayspec condes(0:highest console)
%record (console descriptor) %name console

%recordformat segmentformat (%byteinteger fileptr,blockptr,status)
%recordformat subsegformat (%byteinteger fileptr,start,length, %c
                            %integer diskad)
%recordformat iosegformat (%byteintegerarray block(0:15), %c
                           %byteinteger numbfiles,copy ok, %c
                           %record (subsegformat) %array seg(0:2))
%recordformat segmenttablef (%record (segmentformat) %array seg(0:7), %c
                             %record (iosegformat) %array ioseg(0:1), %c
                             %integer handler)

%constinteger lowuser=k'31'
%constinteger highuser=k'37'
%constinteger hminl=6          ;! highuser-lowuser
%externalrecord (segmenttablef) %array %spec segmenttable (0:hminl)

%constintegername upar6=k'177654'
%recordformat active page registers (%integerarray par,pdr(0:7))
%externalrecord (active page registers) %arrayspec userapr (0:hminl)

%externalroutinespec waitsema(%integer semanumber)
%externalroutinespec receive (%record (message) %name mes)
%externalroutinespec send (%record (reply) %name rep)
%externalroutinespec svc (%record (login message) %name logmes)

%predicatespec input available
%predicatespec end of block
%routinespec start output
```

```
%routinespec release output area
%routinespec process character (%integer char)
%routinespec put character (%byteinteger char)
%routinespec check buffer full
%routinespec block complete
%routinespec echo (%byteinteger char)
%routinespec printchar (%byteinteger char)
%routinespec outputchar (%byteinteger char)
%routinespec print (%byteintegerarrayname string)
%routinespec get ready for copy input
%routinespec copy input
%routinespec copy block (%integer numbchar)
%routinespec initialize

%constinteger false=0
%constinteger true=1

%constinteger ascii=0                   ;! ascii mode
%constinteger raw=1                     ;! raw mode

%constinteger unused=0                  ;! possible states
%constinteger name=1
%constinteger passwd=2
%constinteger rest=3
%constinteger input=4
%constinteger output=5
%constinteger inahead=6
%constinteger outwinp=7
%constinteger inwoutp=8

%constinteger dma=0                     ;! indicate where an interrupt
%constinteger signal=1                  ;! from output goes to

%constinteger cr=k'15'                  ;! special characters
%constinteger lf=k'12'
%constinteger bs=k'10'
%constinteger cancel=k'25'
%constinteger tab=k'11'
%constinteger eot=4
%constinteger etx=3
%constinteger bell=7
%constinteger esc=k'33'

%constinteger scheduler=3               ;! scheduler service number
%constinteger login=4                   ;! login service number
%constinteger recover=60                ;! recovery service number

%constbyteintegerarray name prompt (0:7)=7,cr,lf,'n','a','m','e',':'
%constbyteintegerarray passwd prompt (0:7)=7,'p','a','s','s','w','d',':'
%constbyteintegerarray name too long (0:17)=17,cr,lf,'n','a','m','e',' ', %c
                                        't','o','o',' ','l','o','n','g',cr,lf
%constbyteintegerarray passwd too long (0:21)=21,cr,lf,'p','a','s','s', %c
                    'w','o','r','d',' ','t','o','o',' ','l','o','n','g',cr,lf
%constbyteintegerarray illegal (0:27)=26,'i','l','l','e','g','a','l',' ', %c
    'n','a','m','e',' ','o','r',' ','p','a','s','s','w','o','r','d',cr,lf,0
%constbyteintegerarray already logged in (0:19)=19,'a','l','r','e','a', %c
```

```
                        'd','y',' ','l','o','g','g','e','d',' ','i','n',cr,lf
%constbyteintegerarray base file (0:17)=17,'b','a','s','e',' ','f','i', %c
                                        'l','e',' ','e','r','r','o','r',cr,lf
%constbyteintegerarray too many users (0:17)=16,'t','o','o',' ','m','a', %c
                                      'n','y',' ','u','s','e','r','s',cr,lf,0
%constbyteintegerarray wait (0:9)=9,cr,lf,'w','a','i','t','!', cr,lf
%constbyteintegerarray continue (0:11)=10,'c','o','n','t','i','n','u','e',%c
                                          cr,lf,0

%byteintegerarray inputbuffer (0:255)
%byteintegerarray lengthblock (0:255)
%integerarray saveinput (1:10)
%byteintegerarray promptstring (0:8)

%integer flag,ident,src,echo mode,delmode,passwst,bufferfull,eoi,%c
         start,lineptr,linelen,charptr,numbblocks,inaddress,maxinlength,%c
         firstblock,nextblock,leninp,sindex,outblk,outlen,i

%switch userserv (0:7)
%switch consserv (0:1)
%switch in (unused:inwoutp)
%switch out (output:outwinp)
%switch f1(0:4)


%cycle

   receive(mes)
   console==condes((mes_consid-10)>>1)                    ;! pointer descriptor
   flag=0
   ident=mes_id                                           ;! identification
   src=console_ident
%if ident=console_user %start                             ;! from user
   %if mes_service&7>5 %then flag=3 %and ->exit %c
                       %else ->userserv(mes_service&7)
%finish
%if ident=console_ident %then ->consserv(mes_service)  ;! from console
%if ident=scheduler %start                                ;! from scheduler:
   %if mes_service=0 %start                               ;! segment 0 in core
      rep=0
      copy input                                          ;! copy input to
      ident=console_user                                  ;! user area
      ->sendrep
   %finishelsestart
      outblk=mes_par2;outlen=mes_par3
      console_output address=k'140000'+mes_par2*512
      console_output map register=userapr(console_user-70)_par(1)
      console_number of bytes=mes_par3
      %if console_state#inahead %start                    ;! start output
         start output
      %else                                               ;! state=input while
         console_state=inwoutp                            ;! output pending
      %finish
      %continue
   %finish
%finish
```

```
      %if mes_service=0 %then ->userserv(0)                ;! open service
      flag=1;->exit                                        ;! already opened

      userserv(0): %if console_state=unused %start         ;! open
                      console_state=rest
                      console_user=ident                   ;! user identification
                      initialize                           ;! initial values
                      console_dmasignal=signal
                      outputchar(cr);outputchar(lf)
                      console_mode=mes_par2&1
                      echomode=mes_par3&1
                   %else
                      flag=1                               ;! already in use
                   %finish
                   ->exit

   userserv(1): inaddress=mes_par2;maxinlength=mes_par3 ;! get input
                %if input available %start
                   get ready for copy input                ;! get seg 0 into core
                %else                                      ;! no input available
                   %if console_state#inahead %start        ;! input not busy
                      console_dmasignal=signal             ;! outp. int. to signal
                      %if linelen=0 %then print(promptstring);! print prompt
                  %finish                                  ;! if no char left after continue
                   console_state=input
                %finish
                %continue

   userserv(2): promptstring(0)=mes_service>>8             ;! set input
                promptstring(1)=mes_par2&k'377'            ;! request message
                promptstring(2)=mes_par2>>8&k'377'
                promptstring(3)=mes_par3&k'377'
                promptstring(4)=mes_par3>>8&k'377'
                promptstring(5)=mes_par4&k'377'
                promptstring(6)=mes_par4>>8&k'377'
                promptstring(7)=mes_par5&k'377'
                promptstring(8)=mes_par5>>8&k'377'
                ->exit

   userserv(3): initialize                                 ;! change mode
                console_state=rest
                console_mode=mes_par2&1
                echomode=mes_par3&1
                ->exit

   userserv(4): rep_par2=console_mode                      ;! get mode
                rep_par3=echo mode
                rep_par4=0
                ->sendrep

   userserv(5): bufferfull=false                           ;! close
                %if console_state=output %or console_state=outwinp %start
                   console_address_ostatus=0               ;! stop output
                   release output area
                %finish
                console_state=unused
```

```
                console_user=0
                ->exit

    consserv(0): %if console_mode=ascii %and mes_par2&127=esc %andc
                                        console_state>passwd %start
                   rep_id=recover                          ;! send message
                   rep_source=console_user                 ;! to recover
                   rep_flag=1                              ;! on behalf of user
                   rep_par2=k'177777'                      ;! error code
                   send(rep)
                   initialize                              ;! clear input buffer
                   %if console_state=output %or console_state=outwinp %start
                      console_address_ostatus=0            ;! no more output
                      release output area
                      %finish
                      console_state=rest
                      %continue
                   %finish
                   %if bufferfull=true %then %continue     ;! ignore character
                   ->in(console_state)

    in(unused):    console_state=name                      ;! read user name
                   initialize                              ;! initial values
                   console_mode=ascii
                   echomode=true
                   linelen=1                               ;! hole for length
                   print(name prompt)                      ;! print 'name:'
                   %continue

    in(name):      process character(mes_par2)             ;! line reconstruct
                   %if linelen=0 %then linelen=1           ;! line cancelled
                   %if inputbuffer(charptr)=lf %start      ;! end of name
                      passwst=lengthblock(0)               ;! password start
                      inputbuffer(0)=passwst-2             ;! length name
                      console_state=passwd                 ;! read password
                      echomode=false                       ;! no echo
                      linelen=1                            ;! hole for length
                      print(passwd prompt)                 ;! print 'passwd:'
                      %continue
                   %finish
                   %if linelen>21 %start                   ;! name too long
                      print(name too long)
                      console_state=unused                 ;! end of login
                   %finish
                   %continue

    in(passwd):    process character(mes_par2)             ;! line reconstruct
                   %if linelen=0 %then linelen=1           ;! line cancelled
                   %if inputbuffer(charptr)=lf %start      ;! end of password
                      printchar(cr);printchar(lf)
                      inputbuffer(passwst)=lengthblock(1)-2 ;! length password
                      logmes_id=login                      ;! login service
                      logmes_source=console_ident
                      logmes_buffer==inputbuffer           ;! checks
                      logmes_name=0                        ;! name
                      logmes_passwd=passwst                ;! and password
```

```
                svc(logmes)
                %if logmes_flag=0 %start              ;! admitted
                   console_user=logmes_servnum        ;! user service
                   initialize                         ;! initial values
                   echomode=true
                   console_state=rest
                %else                                 ;! not admitted
                   console_state=unused
                   ->fl(logmes_flag)                  ;! print error message
                   fl(1): print(illegal)
                          %continue
                   fl(2): print(already logged in)
                          %continue
                      fl(3): print(base file)
                             %continue
                      fl(4): print(too many users)
                   %finish
                   %continue
                %finish
                %if linelen>21 %start                 ;! password too long
                   print(passwd too long)
                   console_state=unused               ;! end of login
                %finish
                %continue

in(rest):       console_state=inahead                 ;! input ahead
                console_dmasignal=signal              ;! outp. int. to signal
                process character(mes_par2)           ;! line reconstruct
                %if end of block %or console mode=raw %c
                %then console_state=rest
                %continue

in(input):      process character(mes_par2)           ;! line reconstruct
                %if input available %start            ;! input available
                   console_state=rest
                   get ready for copy input           ;! get seg 0 into core
                %finish
                %continue

in(output):     saveinput(1)=mes_par2                 ;! save character
                sindex=1
                console_state=outwinp                 ;! output while
                %continue                             ;! input pending

in(inahead):    process character(mes_par2)           ;! line reconstruct
                %if end of block %then console_state=rest ;! end of a block
                %continue

in(outwinp):    %if sindex<=9 %start                  ;! saveinput not full
                   sindex=sindex+1
                   saveinput(sindex)=mes_par2         ;! save character
                %finish
                %continue

in(inwoutp):    process character(mes_par2)           ;! line reconstruct
                %if end of block %start               ;! (re)start output
```

```
                    start output
                 %finish
                 %continue

consserv(1):  %if console_state#output %and console_state#outwinp %c
                 %then %continue          ;! message after escape or close
              ->out(console_state)

out(output): release output area
              console_state=rest                          ;! output done
              ident=console_user                          ;! send message to user
              src=console_ident+1
              ->exit

out(outwinp):console_dmasignal=signal                     ;! output int to signal
              i=1
              %while i<=sindex %and bufferfull=false %cycle
                 process character(saveinput(i))          ;! do line reconstruct
                 i=i+1                                    ;! and echo for
              %repeat                                     ;! saved characters
              %if mes_par2=0 %start                       ;! output done
                 release output area
                 %if %not end of block %start             ;! input busy
                    console_state=inahead
                 %else                                    ;! input ended
                    console_state=rest
                 %finish
                 ident=console_user                       ;! send message to user
                 src=console_ident+1
                 ->exit
              %finishhelsestart                           ;! line done
                 %if %not end of block %start             ;! do input first
                    console_state=inwoutp
                 %else
                    start output                          ;! restart output
                 %finish
                 %continue
              %finish

exit:     rep=0
sendrep:  rep_id=ident
          rep_source=src
          rep_flag=flag
          send(rep)

%repeat

%predicate input available

   %if numbblocks>0 %or bufferfull=true %or %c
       (console_mode=raw %and linelen>=maxinlength) %then %true
   %false

%end
```

```
%predicate end of block

   %if console_mode=ascii %and inputbuffer(charptr)=lf %then %true
   %if eoi=true %or bufferfull=true %then %true
   %false

%end

%routine start output

   console_state=output
   console_dmasignal=dma                        ;! output interrupts
   console_address_ostatus=k'100'               ;! handled immediately

%end

%routine release output area

   rep_id=scheduler                             ;! release used blocks
   rep_source=5                                 ;! in segment 1
   rep_flag=console_user
   rep_par2=outblk
   rep_par3=outlen
   send(rep)

%end

%routine process character(%integer char)

   %integer tabcount

   eoi=false                                    ;! end of input indicator
   %if console_mode=ascii %start
      char=char&127                             ;! parity bit
      %if char=cancel %start                    ;! cancel line
         %if linelen>0 %start
            linelen=0
            echo(cancel)
            echo(cr);echo(lf)
         %else
            outputchar(bell)                    ;! no line to cancel
         %finish
         %return
      %finish
      %if char=console_delete char %start       ;! delete character
         %if linelen>0 %start
            linelen=linelen-1
   %if console_delete char=bs %start            ;! if backspace
      %if echomode=true %then printchar(bs) ;! then echo it
   %else                                        ;! else echo \......\
      %if delmode=false %start                  ;! first deleted character
         delmode=true
         echo('\')
      %finish
      charptr=(lineptr+linelen)&255
      echo(inputbuffer(charptr))                ;! echo deleted character
```

```
            %finish
         %else outputchar(bell)                    ;! no characters to delete
         %return
      %finish
      %if delmode=true %start                      ;! end of delete mode
         echo('\')
         delmode=false
      %finish
      %if char=tab %start                          ;! tab character
         tabcount=linelen&7
         %until tabcount&7=0 %or bufferfull=true %cycle
            putcharacter(' ')                      ;! repace it by spaces
            echo(' ')
            check buffer full                      ;! see if buffer is full
            tabcount=tabcount+1
         %repeat
         %return
      %finish
      %if char=etx %start                          ;! etx character
         put character(lf)                         ;! put line feed into buffer
         check buffer full                         ;! see if buffer is full
         block complete                            ;! end of block
         %return
      %finish
      %if char=eot %and linelen=0 %start           ;! eot character
         put character(0)
         eoi=true                                  ;! end of input
         echo(eot);echo(cr);echo(lf)
         check buffer full                         ;! see if buffer is full
         block complete                            ;! end of block
         lengthblock((nextblock+255)&255)=0        ;! make length 0
         %return
      %finish
      %if char=lf %then char=cr                    ;! lf same effect as cr
   %else
      %if char&k'20000'#0 %start                   ;! break
         block complete                            ;! end of block
         eoi=true                                  ;! end of input
         put character(0)
         check buffer full                         ;! see if buffer full
         block complete                            ;! block for end of input
         lengthblock((nextblock+255)&255)=0        ;! make length 0
         %return
      %finish
   %finish
   put character(char)                             ;! put character into buffer
   echo(char)                                      ;! echo character
   %if char=cr %and console mode=ascii %start
      block complete                               ;! end of block
      inputbuffer(charptr)=lf                      ;! cr to lf
      echo(lf)                                     ;! echo cr and lf
   %finish
   check buffer full                               ;! see if bufer is full

%end
```

```
%routine put character(%byteinteger char)

   %if bufferfull=true %then %return
   charptr=(lineptr+linelen)&255                ;! next position
   linelen=linelen+1
   inputbuffer(charptr)=char                    ;! character into buffer

%end

%routine check buffer full

   %if (charptr+1)&255=start %start             ;! buffer full
      bufferfull=true
      print(wait)
   %finish

%end

%routine block complete

   numbblocks=numbblocks+1
   lengthblock(nextblock)=linelen               ;! store length
   nextblock=(nextblock+1)&255                  ;! reset pointers
   lineptr=(lineptr+linelen)&255
   linelen=0

%end

%routine echo(%byteinteger char)

   %if echomode=true %start                     ;! in echo mode
      %if console mode=ascii %start             ;! in ascii mode
         %if char<=k'37' %and char#cr %and char#lf %start
            printchar('^')                      ;! make invisible character
            char=char!k'100'                    ;! visible as ^..
         %finish
         %if char=k'177' %start                 ;! del character
            printchar('^')
            char='?'
         %finish
      %finish
      printchar(char)
   %finish

%end

%routine printchar(%byteinteger char)

%if console_mode=ascii %start
   %if console_line width=console_max line width %c
                        %and char#cr %and char#bs %start
      outputchar(cr)                            ;! console width exceeded
      outputchar(lf)
      outputchar('>')                           ;! indicate overflow
      console_line width=1
   %finish
```

```
      %if char=cr %start
         console_line width=0
      %finishelsestart
         %if char=bs %start
            console_line width=console_line width-1
         %finishelsestart
            %if char#lf %then console_line width=console_line width+1
         %finish
      %finish
   %finish
   outputchar(char)

   %end

   %routine outputchar(%byteinteger char)

      console_address_ostatus=k'100'          ;! enable interrupt from output
      waitsema(console_sema number)           ;! wait for interrupt
      console_address_ostatus=0               ;! disable interrupt
      console_address_obuffer=char            ;! put character in buffer

   %end

   %routine print(%byteintegerarrayname string)

      %integer i

      console_dmasignal=signal
      %for i=1,1,string(0) %cycle
         printchar(string(i))
      %repeat

   %end

   %routine get ready for copy input

      rep_id=scheduler                                    ;! ask the scheduler to
      rep_source=4                                        ;! get segment 0 into core
      rep_flag=console_user<<8!console_ident
      rep_par2=inaddress
      rep_par3=maxinlength
      send(rep)

   %end

   %routine copy input

   %integer savesuppar

   inaddress=inaddress*512                                ;! address in bytes
   leninp=0
   %if bufferfull=true %start
      bufferfull=false                                    ;! input can be continued
      print(continue)
   %finish
   savesuppar=upar6                                       ;! save old mapping register
```

```
    upar6=userapr(console_user-70)_par(0)      ;! setup mapping
    %if numbblocks>0 %start                    ;! complete blocks available
       %if lengthblock(firstblock)=0 %start    ;! end of input
          start=(start+1)&255
          firstblock=(firstblock+1)&255
          numbblocks=numbblocks-1
       %else
          %if lengthblock(firstblock)>maxinlength %start
             copyblock(maxinlength)           ;! copy part of first block
             lengthblock(firstblock)=lengthblock(firstblock)-maxinlength
          %else
             %until numbblocks=0 %or leninp+lengthblock(firstblock) > %c
                maxinlength %or lengthblock(firstblock)=0 %cycle
                copy block(lengthblock(firstblock)) ;! copy as many
                firstblock=(firstblock+1)&255       ;! complete blocks
                numbblocks=numbblocks-1             ;! as possible
             %repeat
          %finish
       %finish
    %else                                      ;! buffer full or mode=raw
                                               ;! and linelen>maxinlength
       copyblock(maxinlength)                  ;! copy the maximum
       lineptr=(lineptr+maxinlength)&255
       linelen=linelen-maxinlength
    %finish
    upar6=savesuppar                           ;! reset mapping register
    segmenttable(console_user-70)_handler=0
    rep_par2=leninp

%end

%routine copy block(%integer numbchar)

    %recordformat bufferformat (%byteintegerarray char(0:17777))
    %constrecord(bufferformat)%name userbuffer=k'140000'

    %integer i

    %for i=0,1,numbchar-1 %cycle
       userbuffer_char(inaddress+leninp+i)=inputbuffer((start+i)&255)
    %repeat
    leninp=leninp+numbchar
    start=(start+numbchar)&255

%end

%routine initialize

    bufferfull=false                           ;! give initial values
    start=0                                    ;! to variables
    lineptr=0
    linelen=0
    firstblock=0
    nextblock=0
    numbblocks=0
    console_line width=0
```

```
        delmode=false
        promptstring(0)=0                              ;! no prompt set

    %end

%endofprogram
```

```
!******************
!*** scheduler ***
!******************

%control k'100001'

%begin

   %recordformat message(%byteinteger id,source,%integer par1,par2,par3, %c
                                                          par4,par5)
   %externalroutinespec receive (%record (message) %name mes)
   %externalroutinespec send (%record (message) %name mes)
   %externalroutinespec svc (%record (message) %name mes)

%recordformat processtable entry (%integer r0,r1,r2,r3,r4,r5,sp,pc,psw, %c
              received,%byteinteger unus1,heldup,%integer unus2,rsmoni, %c
              %byteinteger asleep,unus3,%integer unus4,unus5)
%constinteger idle=k'47'
%externalrecord (processtable entry) %array %spec proctab(0:idle)

%recordformat segmentformat (%byteinteger fileptr,blockptr,status)
%recordformat subsegformat (%byteinteger fileptr,start,length, %c
                            %integer diskad)
%recordformat iosegformat (%byteintegerarray block(0:15), %c
                           %byteinteger numbfiles,copy ok, %c
                           %record (subsegformat) %array subseg(0:2))
%recordformat segmenttablef (%record (segmentformat) %array seg(0:7),%c
                             %record (iosegformat) %array ioseg(0:1),%c
                             %integer handler)

%constinteger lowuserproc=k'31'        ;! lowest user process number
%constinteger highuserproc=k'37'       ;! highest user process number
%constinteger hminl=6                  ;! highuserproc-lowuserproc
%constinteger lowuser=70               ;! lowest user service number
%constinteger usminup=45               ;! lowest user service number minus
                                       ;! lowest user process number

%externalrecord (segmenttablef) %array %spec segmenttable(0:hminl)

%recordformat active page registers (%integerarray par,pdr(0:7))
%externalrecord (active page registers) %array %spec userapr(0:hminl)

%recordformat block format (%integer par,pdr,stblk,diskad,file block, %c
        %byteinteger status,users,users in core,length,prev,next)
%constinteger maxblock=50
%externalrecord (block format) %array %spec block(0:maxblock)

%recordformat console registers (%integer istatus,ibuffer,ostatus,obuffer)
%recordformat console descriptor (%record(console registers)%name address,%c
              %integer ident,max line width,delete char,sema number, %c
              dmasignal,mode,output address,output map register, %c
              number of bytes,line width,state,dma output status,user)
%constinteger highest console=5
%externalrecord (console descriptor) %arrayspec condes(0:highest console)

%externalintegerspec runproc
```

```
%routinespec adjust time and queue(%integer to queue)
%routinespec update status processes
%routinespec check processes(%integer queue,to queue)
%routinespec select running process
%routinespec select candidate
%routinespec load candidate
%routinespec choose victim
%routinespec search(%integer head)
%predicatespec segments in core(%integer process)
%routinespec remove core(%integername result)
%routinespec request to clock(%integer service,interval)
%routinespec call disk handler(%integer service,diskad,length,stblk,mark)
%routinespec get core(%integer length,%integername stblk,result)
%routinespec release core(%integer stblk,length)
%routinespec release core from output segment(%record (segmenttablef) %c
                                              %name segtable)
%predicatespec error in blocks(%integer segment)
%routinespec change(%integer process,new queue)
%routinespec initialize

%constinteger false=0
%constinteger true=1

%constinteger in=0                      ;! segment states
%constinteger out=1
%constinteger free=0                    ;! iosegmentblock states
%constinteger file=1
%constinteger output=2
%constinteger in core=0                 ;! block states
%constinteger out core=1
%constinteger to core=2
%constinteger from core=3

%constinteger run queue=0               ;! queues
%constinteger candidate queue=1
%constinteger file system queue=2
%constinteger high prior queue=3
%constinteger low prior queue=4
%constinteger blocked queue=5
%constinteger login queue=6

%constinteger clock=0                   ;! message identification
%constinteger disk handler=1
%constinteger connect=2
%constinteger disconnect=3
%constinteger handler input=4
%constinteger handler output=5
%constinteger kernel=6
%constinteger logout=7
%constinteger output request=8

%constinteger core manager=2            ;! services
%constinteger scheduler=3
%constinteger connect service=40
%constinteger disconnect service=41
```

```
%constinteger recover service=60
%constinteger logout service=61

%constinteger interrupt=0              ;! for clock messages
%constinteger time=1
%constinteger read=0                   ;! for disk messages
%constinteger write=1
%constinteger get=0                    ;! for core manager messages
%constinteger release=1

%constinteger written bit=k'100'       ;! written into bit pdr register

%constinteger timeslice=150            ;! timing constants
%constinteger large allocation=4
%constinteger small allocation=2
%constinteger min rest=25
%constinteger maxrun=3
%constinteger highlowratio=3

%switch called from(clock:output request)
%switch diskh(read:write)

%recordformat userlist entry (%byteinteger queue,complete,succ,pred, %c
                              slices,restslice,%integer slicebegin)
%record (userlist entry) %array userlist(lowuserproc:highuserproc)

%record (message) mes

%record (segmenttablef) %name segtable

%integerarray head(run queue:login queue)

%integername candidate

%integer run,current time,highlowcounter,process,segment,blk,src, %c
         complete,drive,parts from core,clock reply coming,victim, %c
         previous interval,s,h

initialize

drive=0

%cycle

   receive(mes)
   %if clock<=mes_id<=logout %then ->called from(mes_id)  %c
   %else %start                                ;! called from user
      %if mes_source#scheduler %start          ;! put output request
         ->called from(output request)
      %finishelsestart                         ;! illegal request
         process=mes_id                        ;! send refusal
         mes=0
         mes_id=process
         mes_source=scheduler
         send(mes)
         %continue
```

```
      %finish
   %finish


   called from(clock):

      current time=mes_par2                        ;! contains current time
      clock reply coming=false                     ;! no more replies on the way
      %if runproc#idle %start
         %if proctab(runproc)_asleep=connect service %or %c
           proctab(runproc)_asleep=disconnect service %or %c
           proctab(runproc)_asleep=recover service %or %c
           proctab(runproc)_asleep=logout service %start
            adjust time and queue(file system queue)          ;! adjust
         %else                                                ;! queue
            %if proctab(runproc)_asleep#k'377'                ;! of
               adjust time and queue(blocked queue)           ;! running
            %else                                             ;! process
               adjust time and queue(run queue)
            %finish
         %finish
      %finish
      check processes(file system queue,run queue)
      check processes(blocked queue,high prior queue)
      update status processes
      %continue

   called from(disk handler):

   process=mes_par2>>1&k'37'                       ;! mes_par2 contains
   segtable==segmenttable(process-lowuserproc)     ;! information about
   segment=mes_par2>>6&7                           ;! request
   ->diskh(mes_par2&1)

   diskh(read):                                    ;! segment read
      segtable_seg(segment)_status=in
      block(segtable_seg(segment)_blockptr)_status=in core
      %if segment=0 %and segtable_handler>0 %start;!input can be
         mes_id=segtable_handler                   ;! copied now
         mes_source=scheduler
         mes_par1=0
         send(mes)
         segtable_ioseg(0)_copy ok=false           ;! copy segment not ok
      %finish
      complete=true                                ;! process in core ?
      s=0
      %until complete=false %or s=8 %cycle
         %if segtable_seg(s)_fileptr#0 %c
            %and segtable_seg(s)_status#in %then complete=false
         s=s+1
      %repeat
      userlist(process)_complete=complete
      %if userlist(candidate)_complete=false %then load candidate
      %if userlist(candidate)_complete=true %c
        %and run=0 %start                                ;! start process
         request to clock(time,0)
```

```
            run=1
            change(candidate,run queue)
            select running process
         %finish
         %continue

      diskh(write):                                          ;! segment written
         blk=segtable_seg(segment)_blockptr
         block(blk)_status=out core
         %if segment#1 %start
            release core(block(blk)_stblk,block(blk)_length)
         %else                                               ;! segment 1 is output
            release core from output segment(segtable)
         %finish
         parts from core=parts from core-1                   ;! parts on the way
         %if candidate#0 %then load candidate
         %continue

   called from(connect):
         process=mes_par1-usminup
         request to clock(time,0)                            ;! get current time
         %if process=runproc %start                          ;! process to
            adjust time and queue(candidate queue)           ;! candidate queue
         %else
            %if runproc#idle %then adjust time and queue(run queue)
            %if userlist(process)_queue=run queue %then run=run-1
            change(process,candidate queue)
         %finish
         userlist(process)_complete=false                    ;! process not in core
         load candidate
         select running process
         %continue

   called from(disconnect):
         request to clock(time,0)                            ;! get current time
         %if runproc#idle %then adjust time and queue(run queue)
         check processes(file system queue,run queue)
         update status processes
         %continue

   called from(handler input):
         segtable==segmenttable(mes_par1>>8-lowuser)
         %if error in blocks(0) %start                       ;! illegal blocks
            mes_id=mes_par1>>8                               ;! send refusal
            mes_source=mes_par1&k'377'
            mes_par2=2
            mes_par3=0;mes_par4=0;mes_par5=0
            send(mes)
         %else
            segtable_handler=mes_par1&k'377'                 ;! remember handler
            %if segtable_seg(0)_status=in %start
               mes_id=mes_par1&k'377'                        ;! send reply
               mes_source=scheduler                          ;! immediately
               mes_par1=0
               send(mes)
               segtable_ioseg(0)_copy ok=false               ;! copy segment not ok
```

```
            %else                                          ;! get process into core
               change(mes_par1>>8-usminup,high prior queue)
               request to clock(time,0)
               %if runproc#idle %then adjust time and queue(run queue)
               update status processes
            %finish
         %finish
         %continue

called from(handler output):
      segtable==segmenttable(mes_par1-lowuser)
      %for blk=mes_par2,1,mes_par2+(mes_par3-1)//512 %cycle
         segtable_ioseg(1)_block(blk)=free          ;! blocks free
      %repeat
      blk=segtable_seg(1)_blockptr
      %if block(blk)_status=out core %start         ;! core can be released
         release core(block(blk)_stblk+mes_par2,1+(mes_par3-1)//512)
         %if candidate#0 %then load candidate
      %finish
      %continue

called from(kernel):
      request to clock(time,0)                      ;! get current time
      %if runproc#idle %start
         %if proctab(runproc)_asleep=connect service %or %c
           proctab(runproc)_asleep=disconnect service %or %c
           proctab(runproc)_asleep=recover service %or %c
           proctab(runproc)_asleep=logout service %start
            adjust time and queue(file system queue)
         %else
            %if proctab(runproc)_asleep#k'377' %start
               adjust time and queue(blocked queue)
            %else
               %continue
            %finish
         %finish
      %finish
      check processes(file system queue,run queue)
      check processes(blocked queue,high prior queue)
      update status processes
      %continue

called from(logout):
      process=mes_par1-usminup                      ;! process into
      change(process,login queue)                   ;! login queue
      userlist(process)_complete=false              ;! initialize
      userlist(process)_slices=small allocation     ;! variables
      userlist(process)_restslice=timeslice
      userlist(process)_slicebegin=0
      ->called from(disconnect)

called from(output request):
      process=mes_id
      segtable==segmenttable(process-lowuser)
      src=mes_source
      %if process#condes((src-10)>>1)_user %start ;! console
```

```
            mes=0                                   ;! not opened
            mes_id=process                          ;! send refusal
            mes_source=src
            mes_par1=1
            send(mes)
            %continue
         %finish
         %if error in blocks(1) %start              ;! illegal blocks
            mes=0                                   ;! send refusal
            mes_id=process
            mes_source=src
            mes_par1=2                               ;! error flag
         %else
            mes_id=src-1                            ;! pass on request
            mes_source=scheduler                    ;! to handler
            mes_par1=1
         %finish
         send(mes)
         %continue

%repeat


%routine adjust time and queue(%integer to queue)

   %integer t

   t=current time-userlist(runproc)_slicebegin      ;! 0<=time<1000
   %if t<0 %then t=t+1000
   %if to queue=blocked queue %start                ;! process heldup
      userlist(runproc)_slices=small allocation
      userlist(runproc)_restslice=timeslice
   %else
      t=userlist(runproc)_restslice-t
      %if t<min rest %start                         ;! time slice over
         t=timeslice
         head(run queue)=userlist(runproc)_succ     ;! next in run queue
         userlist(runproc)_slices=userlist(runproc)_slices-1
         %if userlist(runproc)_slices=0 %start      ;! slices over
            %if to queue=run queue %start
               userlist(runproc)_slices=large allocation
               to queue=low prior queue
            %else
               userlist(runproc)_slices=1
            %finish
         %finish
      %finish
      userlist(runproc)_restslice=t                 ;! rest of slice
   %finish
   %if to queue#run queue %start                    ;! no longer in
      change(runproc,to queue)                      ;! run queue
      run=run-1
   %finish

%end
```

```
%routine update status processes

   %if candidate=0 %then select candidate
   %if candidate#0 %start                                    ;! candidate present
      %if userlist(candidate)_complete=false %then load candidate
      %if userlist(candidate)_complete=true %and run<maxrun %start
         run=run+1                                           ;! candidate into
         change(candidate,run queue)                         ;! run queue
      %finish
   %finish
   select running process

%end


%routine check processes(%integer queue,to queue)

   %integer process,nextproc,stop

   nextproc=head(queue)
   %if nextproc#0 %start                                     ;! check all processes
      stop=false                                             ;! in queue
      %until stop=true %cycle
         process=nextproc
         %if userlist(process)_succ=head(queue) %then stop=true %c
         %else nextproc=userlist(process)_succ
         %if proctab(process)_asleep=k'377' %start           ;! no longer heldup
            %if to queue=run queue %then run=run+1
            change(process,to queue)
         %finish
      %repeat
   %finish

%end


%routine select running process

   %integer process

   %if head(run queue)=0 %start                              ;! no runnable process
      runproc=idle
   %else                                                     ;! else get first one
      process=head(run queue)                                ;! out of queue
      userlist(process)_slice begin=current time
      request to clock(interrupt,userlist(process)_restslice)
      runproc=process
   %finish

%end

%routine select candidate

   %if highlowcounter>0 %start                               ;! from high prior
```

```
      %if head(high prior queue)#0 %start              ;! queue if possible
         change(head(high prior queue),candidate queue)
         highlowcounter=highlowcounter-1
      %else                                            ;! else from low prior
         %if head(low prior queue)#0 %start            ;! queue if possible
            change(head(low prior queue),candidate queue)
            highlowcounter=highlowratio
         %finish
      %finish
   %else                                               ;! from low prior
      highlowcounter=highlowratio
      %if head(low prior queue)#0 %start               ;! queue if possible
         change(head(low prior queue),candidate queue)
      %else
         %if head(high prior queue)#0 %start           ;! else from high prior
            change(head(high prior queue),candidate queue) ;! queue
            highlowcounter=highlowcounter-1            ;! if possible
         %finish
      %finish
   %finish
   victim=0

%end


%routine load candidate

   %switch removed(1:2)

   %record(segmenttablef)%name segtable

   %integer s,stblk,blk,result,complete,mark

   segtable==segmenttable(candidate-lowuserproc)
   %for s=0,1,7 %cycle                                ;! files in core ?
      %if segtable_seg(s)_fileptr#0 %and segtable_seg(s)_status=out %start
         blk=segtable_seg(s)_blockptr
         %if block(blk)_status=in core %start
            segtable_seg(s)_status=in
            userapr(candidate-lowuserproc)_par(s)=block(blk)_par
            block(blk)_users in core=block(blk)_users in core+1
         %finish
      %finish
   %repeat
   complete=true
   s=0
   %until s=8 %cycle                                  ;! load files out core
    %if segtable_seg(s)_fileptr#0 %and segtable_seg(s)_status=out %start
         complete=false
         blk=segtable_seg(s)_blockptr
         %if block(blk)_status=out core %start
            get core(block(blk)_length,stblk,result)
            %if result=0 %start                       ;! got core
               mark=(s<<5!candidate)<<1
               call disk handler(read,block(blk)_diskad,stblk, %c
                                 block(blk)_length,mark)
```

```
            block(blk)_status=to core
            block(blk)_users in core=1
            block(blk)_stblk=stblk
            block(blk)_par=stblk*8
            userapr(candidate-lowuserproc)_par(s)=stblk*8
         %else                                          ;! no core got
            %if parts from core>0 %then %return
            %until result#0 %cycle                      ;! try to get core
               %if victim=0 %then choose victim         ;! from the victim
               %if victim=0 %then %return
               %if victim=candidate %start              ;! remove all core
                  %until result=0 %cycle                ;! from the candidate
                     remove core(result)                ;! itself
                     %repeat
                     %return
                  %finish
                  remove core(result)
               %repeat
               ->removed(result)
               removed(1): %continue                    ;! core free
               removed(2): %return                      ;! core becomes free
            %finish
         %finish
      %finish
      s=s+1
   %repeat
   userlist(candidate)_complete=complete

%end


%routine choose victim

   victim=0                                             ;! get victim from
   %if head(blocked queue)#0 %start                     ;! the blocked queue
      search(head(blocked queue))
      %if victim#0 %then %return
   %finish
   %if head(low prior queue)#0 %start                   ;! the low prior queue
      search(head(low prior queue))
      %if victim#0 %then %return
   %finish
   %if head(high prior queue)#0 %start                  ;! the high prior queue
      search(head(high prior queue))
      %if victim#0 %then %return
   %finish
   %if head(candidate queue)#userlist(head(candidate queue))_succ %or %c
       (run=0 %and head(file system queue)=0) %start    ;! the candidate queue
      search(head(candidate queue))
   %finish

%end


%routine search(%integer head)
```

```
      %integer process

      process=userlist(head)_pred                              ;! last process
      %until process=head %cycle
         %if segments in core(process) %start
            victim=process                                     ;! victim found
            %return
         %finishhelsestart
            process=userlist(process)_pred                     ;! previous process
         %finish
      %repeat

      %end


      %predicate segments in core(%integer process)

         %integer s

         segtable==segmenttable(process-lowuserproc)
         %for s=0,1,7 %cycle                                   ;! if any segment
            %if segtable_seg(s)_fileptr#0 %and %c
                segtable_seg(s)_status=in %then %true          ;! in core then true
         %repeat
         %false

      %end


      %routine remove core(%integername result)

      %record(segmenttablef)%name segtable

      %integer s,blk,mark

      userlist(victim)_complete=false
      segtable==segmenttable(victim-lowuserproc)
      %for s=2,1,7 %cycle                                      ;! normal segments
         %if segtable_seg(s)_fileptr#0 %and segtable_seg(s)_status=in %start
            blk=segtable_seg(s)_blockptr
            block(blk)_pdr=block(blk)_pdr!userapr(victim-lowuserproc)_pdr(s)
            userapr(victim-lowuserproc)_pdr(s)= %c
                     userapr(victim-lowuserproc)_pdr(s)&(\written bit)
            %if block(blk)_pdr&written bit=0 %start
               segtable_seg(s)_status=out
               block(blk)_users in core=block(blk)_users in core-1
               %if block(blk)_users in core=0 %start
                  release core(block(blk)_stblk,block(blk)_length)
                  block(blk)_status=out core
                  result=1                                     ;! core free
                  %return
               %finish
            %finish
         %finish
      %repeat
      %for s=0,1,1 %cycle                                      ;! i/o segments
```

```
      %if segtable_seg(s)_status=in %c
        %and %not (s=0 %and segtable_handler>0) %start
         blk=segtable_seg(s)_blockptr
         segtable_seg(s)_status=out
         %if (segtable_ioseg(s)_copy ok=true %and %c
           userapr(victim-lowuserproc)_pdr(s)&written bit=0) %start
            %if s=1 %then release core from output segment(segtable) %c
            %else release core(block(blk)_stblk,block(blk)_length)
            block(blk)_status=out core
            result=1                                    ;! core free
            %return
         %finishelsestart                               ;! to disk first
            mark=(s<<5!victim)<<1!1
            call disk handler(write,block(blk)_diskad,block(blk)_stblk,%c
                                                  block(blk)_length,mark)
            userapr(victim-lowuserproc)_pdr(s)= %c
                     userapr(victim-lowuserproc)_pdr(s)&(\written bit)
            segtable_ioseg(s)_copy ok=true
            block(blk)_status=from core
            result=2                                    ;! core becomes free
           %return
        %finish
     %finish
  %repeat
  %for s=2,1,6 %cycle                                   ;! normal segments
     %if segtable_seg(s)_fileptr#0 %and segtable_seg(s)_status=in %start
        segtable_seg(s)_status=out
        blk=segtable_seg(s)_blockptr
        block(blk)_users in core=block(blk)_users in core-1
        %if block(blk)_users in core=0 %start
           mark=(s<<5!victim)<<1!1
           call disk handler(write,block(blk)_diskad,block(blk)_stblk,%c
                                                  block(blk)_length,mark)
           block(blk)_pdr=block(blk)_pdr&(\written bit)
           block(blk)_status=from core
           result=2                                     ;! core becomes free
           %return
        %finish
     %finish
  %repeat
  result=0                                              ;! no core found
  victim=0

%end


%routine request to clock(%integer service,interval)

  mes_id=clock
  mes_source=scheduler
  mes_par1=service
  %if service=time %start                               ;! request for time
     svc(mes)
     current time=mes_par2
  %else                                                 ;! reply after time
     %if clock reply coming=false %or previous interval>interval %start
```

```
            clock reply coming=true
            previous interval=interval
            mes_par2=interval
            mes_par3=0
            send(mes)
        %finish
    %finish

%end

%routine call disk handler(%integer service,diskad,stblk,length,mark)

    mes_id=disk handler
    mes_source=scheduler
    mes_par1=drive<<8!service
    mes_par2=diskad
    mes_par3=stblk
    mes_par4=length
    mes_par5=mark
    send(mes)
    %if service=write %then parts from core=parts from core+1

%end

%routine get core(%integer length,%integername stblk,result)

    mes_id=core manager
    mes_source=scheduler
    mes_par1=get
    mes_par3=length
    svc(mes)
    result=mes_par1
    stblk=mes_par2

%end

%routine release core(%integer stblk,length)

    mes_id=core manager
    mes_source=scheduler
    mes_par1=release
    mes_par2=stblk
    mes_par3=length
    svc(mes)

%end

%routine release core from output segment(%record (segmenttablef) %c
                                          %name segtable)

    %integer start,length,blk

    start=0                                          ;! release core which
    length=0                                         ;! is not used for
    %for blk=0,1,15 %cycle                           ;! output
```

```
      %if segtable_ioseg(1)_block(blk)#output %start
         length=length+1
      %else
         %if length>0 %start
            release core(block(segtable_seg(1)_blockptr)_stblk+start,length)
            start=start+length+1
            length=0
         %else
            start=start+1
         %finish
      %finish
   %repeat
   %if length>0 %then %c
      release core(block(segtable_seg(1)_blockptr)_stblk+start,length)

%end


%predicate error in blocks(%integer segment)

   %integer blks,i

   blks=(mes_par3-1)//512
   %if mes_par2&(\k'17')#0 %or mes_par3<=0 %c
      %or mes_par2+blks>16 %then %true                ;! illegal block
   %if segment=0 %then %false
   %for i=mes_par2,1,mes_par2+blks %cycle
      %if segtable_ioseg(1)_block(i)=file %then %true ;! block in use
   %repeat
   %for i=mes_par2,1,mes_par2+blks %cycle
      segtable_ioseg(1)_block(i)=output               ;! set output
   %repeat
   %false

%end

%routine change(%integer process,new queue)

   %integer successor,predecessor,old queue

   successor=userlist(process)_succ
   predecessor=userlist(process)_pred
   old queue=userlist(process)_queue
   %if successor=process %start                       ;! one element in old queue
      head(old queue)=0
   %else                                              ;! more elements in old queue
      %if head(old queue)=process %then head(old queue)=successor
                                                      ;! element at head
      userlist(predecessor)_succ=successor
      userlist(successor)_pred=predecessor
   %finish
   %if head(new queue)=0 %start                       ;! new queue is empty
      userlist(process)_succ=process
      userlist(process)_pred=process
      head(new queue)=process
   %else                                              ;! insert into new queue
```

```
      userlist(process)_succ=head(new queue)
      userlist(process)_pred=userlist(head(new queue))_pred
      userlist(userlist(head(new queue))_pred)_succ=process
      userlist(head(new queue))_pred=process
      %if new queue=candidate queue %then head(candidate queue)=process
   %finish
   userlist(process)_queue=new queue
   %if new queue=candidate queue %then victim=0

%end

%routine initialize

   %integer i

   run=0
   victim=0
   highlowcounter=highlowratio
   parts from core=0
   clock reply coming=false
   %for i=run queue,1,login queue %cycle
      head(i)=0
   %repeat
   %for i=lowuserproc,1,highuserproc %cycle
      userlist(i)_succ=i+1
      userlist(i)_pred=i-1
      userlist(i)_queue=login queue
      userlist(i)_complete=false
      userlist(i)_slices=small allocation
      userlist(i)_restslice=timeslice
      userlist(i)_slicebegin=0
   %repeat
   userlist(lowuserproc)_pred=highuserproc
   userlist(highuserproc)_succ=lowuserproc
   head(login queue)=lowuserproc
   candidate==head(candidate queue)

%end

%endofprogram
```

```
!*****************************
!**** external routines ***
!*****************************

   %recordformat message (%integer r0,r1,r2,r3,r4,r5)

   %externalroutine receive (%record (message) %name mes)

      *k'010546'        ;! mov lnb,-(sp)
      *k'104003'        ;! emt 3
      *k'010546'        ;! mov r5,-(sp)
      *k'016605'        ;! mov 2(sp),lnb
      *k'000002'
      *k'010075'        ;! mov r0,@6(lnb)
      *k'000006'
      *k'016500'        ;! mov 6(lnb),r0
      *k'000006'
      *k'005720'        ;! tst (r0)+
      *k'010120'        ;! mov r1,(r0)+
      *k'010220'        ;! mov r2,(r0)+
      *k'010320'        ;! mov r3,(r0)+
      *k'010420'        ;! mov r4,(r0)+
      *k'012610'        ;! mov (sp)+,(r0)
      *k'005726'        ;! tst (sp)+

%end


%externalroutine send (%record (message) %name mes)

      *k'010546'        ;! mov lnb,-(sp)
      *k'016505'        ;! mov 6(lnb),r5
      *k'000006'
      *k'012500'        ;! mov (r5)+,r0
      *k'012501'        ;! mov (r5)+,r1
      *k'012502'        ;! mov (r5)+,r2
      *k'012503'        ;! mov (r5)+,r3
      *k'012504'        ;! mov (r5)+,r4
      *k'011505'        ;! mov (r5),r5
      *k'104002'        ;! emt 2
      *k'012605'        ;! mov (sp)+,lnb

%end


%externalroutine svc (%record (message) %name mes)

   *k'010546'        ;! mov lnb,-(sp)
   *k'016505'        ;! mov 6(lnb),r5
   *k'000006'
   *k'012500'        ;! mov (r5)+,r0
   *k'012501'        ;! mov (r5)+,r1
   *k'012502'        ;! mov (r5)+,r2
   *k'012503'        ;! mov (r5)+,r3
   *k'012504'        ;! mov (r5)+,r4
   *k'011505'        ;! mov (r5),r5
```

```
        *k'000004'      ;! iot
        *k'010546'      ;! mov r5,-(sp)
        *k'016605'      ;! mov 2(sp),lnb
        *k'000002'
        *k'010075'      ;! mov r0,@6(lnb)
        *k'000006'
        *k'016500'      ;! mov 6(lnb),r0
        *k'000006'
        *k'005720'      ;! tst (r0)+
        *k'010120'      ;! mov r1,(r0)+
        *k'010220'      ;! mov r2,(r0)+
        *k'010320'      ;! mov r3,(r0)+
        *k'010420'      ;! mov r4,(r0)+
        *k'012610'      ;! mov (sp)+,(r0)
        *k'005726'      ;! tst (sp)+

%end



%externalroutine waitsema (%integer sema)

        *k'016500'      ;! mov 6(lnb),r0
        *k'000006'
        *k'104001'      ;! emt 1

%end

%externalroutine signalsema (%integer sema)

        *k'016500'      ;! mov 6(lnb),r0
        *k'000006'
        *k'104000'      ;! emt 0

%end


%externalroutine printsymbol(%integer sym)

    %recordformat la (%integer status,buffer)
    %constrecord (la) %name printer=k'177564'

    %recordformat semaphore (%integer counter,waiting)
    %constrecord (semaphore) %name printsema=k'602'

    %while printer_status&k'200'=0 %cycle ; %repeat
    printer_buffer=sym
    printsema_counter=0                 ;! reset semaphore

%end

%externalroutine print (%string (63) s)

    %externalroutinespec printsymbol(%integer i)
    %integer i,j
```

```
        printsymbol(k'15');printsymbol(k'12')
        %cycle i=1,1,length(s)
           j=charno(s,i)
           printsymbol(j)
        %repeat
        printsymbol(k'15');printsymbol(k'12')

    %end

    %externalroutine poctal(%integer i)

        %externalroutinespec printsymbol(%integer i)
        %integer n
        printsymbol(' ')
        %cycle n=15,-3,0
           printsymbol('0'+i>>n&7)
        %repeat

    %end

    %endoffile
```

```
**********************
** initialization ***
**********************

control k'100001'

begin

  %constinteger available memory=k'340'
  %constinteger system size=k'110'
  %externalinteger start user memory
  %externalinteger available user memory

  ! interrupt and trap routines in kernel
  %externalintegerspec errorep              ;! error traps
  %externalintegerspec iotep                ;! iot
  %externalintegerspec powep                ;! power failure
  %externalintegerspec emtep                ;! emt
  %externalintegerspec clockep              ;! clock interrupt
  %externalintegerspec diskep               ;! disk interrupt
  %externalintegerspec cinep                ;! console input interrupt
  %externalintegerspec coutep               ;! console output interrupt

  ! addresses processes
  %externalintegerspec clock                ;! clock handler
  %externalintegerspec clostk
  %externalintegerspec disk                 ;! disk handler
  %externalintegerspec disstk
  %externalintegerspec core                 ;! core manager
  %externalintegerspec corstk
  %externalintegerspec filsys               ;! file system
  %externalintegerspec filstk
  %externalintegerspec conhan               ;! console handler
  %externalintegerspec congla               ;! gla for console handlers
  %externalintegerspec schedu               ;! scheduler
  %externalintegerspec schstk

  %constinteger number of user processes=7

  %constintegername strtpr=k'165566'        ;! start user processes
  %constintegername userstack=k'156000'     ;! user stack

  %constinteger rubout=k'177'               ;! rubout character
  %constinteger backspace=k'10'             ;! backspace character

  %constinteger highest console=5

  %recordformat console information (%integer device address, %c
                                     vector address, %c
                                     max line width, %c
                                     delete char)

  %record (console information) %array console (0:highest console)

  %recordformat console descriptor (%integer address,ident, %c
                max line width,delete char,sema number,dmasignal, %c
```

```
                mode,output address,output map register, %c
                number of bytes,line width,state,dma output status,user)
%externalrecord (console descriptor) %array condes (0:highest console)

%constintegerarrayname kernel par=k'172340' ;! kernel page address registers
%constintegerarrayname kernel pdr=k'172300' ;! page descriptor registers
%constintegerarrayname user par=k'177640'   ;! user page address registers
%constintegerarrayname user pdr=k'177600'   ;! page descriptor registers

%constintegername mem man status 0=k'177572' ;! memory management status

%externalintegerarrayspec suppar (0:7)  ;! copy supervisor page descriptor
                                        ;! registers in kernel
%externalintegerarrayspec suppdr (0:7)  ;! copy supervisor page descriptor
                                        ;! registers in kernel

%recordformat processtable entry (%integer r0,r1,r2,r3,r4,r5, %c
                %integername sp,pc,%integer psw,received, %c
                %byteinteger unus1,heldup,%integer unus2,rsmoni, %c
                %byteinteger asleep,unus3,%integer unus4,unus5)
%constinteger idle=k'47'
%externalrecord (processtable entry) %array %spec proctab (0:idle)

%externalbyteintegerarrayspec serv (0:k'177')

%externalbyteintegerarrayspec readyq(0:k'37')    ;! ready queue
%externalintegerspec begrdy                      ;! pointers in
%externalintegerspec endrdy                      ;! ready queue

%externalintegerspec process                     ;! current process
%externalrecord(processtable entry)%namespec apd ;! pointer to entry current
                                                 ;! process in proctab

%constinteger intpsw=k'340'          ;! interrupt psw
%constinteger userps=k'140000'       ;! user psw
%constinteger notasleep=k'377'       ;! not asleep indicator in proctab

%constinteger login=4                ;! login service

%externalroutinespec kernex          ;! exit routine in kernel

%routinespec insert (%integer process,%integername program counter, %c
                                                  stack pointer)
%routinespec fill in page address registers (%integerarrayname par)
%routinespec fill in page descriptor registers (%integerarrayname pdr)

%constintegerarrayname memory=0      ;! used to address memory

%integer user,connum,glabase,loc,s


! fill in interrupt and trap vectors
memory(4>>1)=errorep                 ;! error trap 4
memory(6>>1)=intpsw
memory(k'10'>>1)=errorep             ;! error trap 10
memory(k'12'>>1)=intpsw!1
```

```
memory(k'14'>>1)=errorep      ;! break point trap
memory(k'16'>>1)=intpsw!2
memory(k'20'>>1)=iotep        ;! iot
memory(k'22'>>1)=intpsw
memory(k'24'>>1)=powep        ;! power interrupt
memory(k'26'>>1)=intpsw
memory(k'30'>>1)=emtep        ;! emt
memory(k'32'>>1)=intpsw
memory(k'34'>>1)=errorep      ;! trap
memory(k'36'>>1)=intpsw!3
memory(k'104'>>1)=clockep     ;! clock interrupt
memory(k'106'>>1)=intpsw
memory(k'220'>>1)=diskep      ;! disk interrupt
memory(k'222'>>1)=intpsw
memory(k'250'>>1)=errorep     ;! error trap 250
memory(k'252'>>1)=intpsw!4

begrdy=0
endrdy=0

insert(0,clock,clostk)        ;! insert processes
insert(1,disk,disstk)         ;! into processtable
insert(2,core,corstk)
insert(3,filsys,filstk)
insert(k'30',schedu,schstk)

%for s=0,1,k'177' %cycle
   serv(s)=k'377'
%repeat
serv(0)=0;serv(1)=1;serv(2)=2;serv(3)=k'30';serv(4)=3
%for s=40,1,49 %cycle
   serv(s)=3
%repeat
serv(60)=3
serv(61)=3

%for user=1,1,number of user processes %cycle
   insert(k'30'+user,strtpr,userstack)
   proctab(k'30'+user)_asleep=login
  serv(69+user)=k'30'+user
%repeat

console(0)_device address=k'177560'
console(0)_vector address=k'60'
console(0)_max line width=80
console(0)_delete char=rubout

console(1)_device address=k'175610'
console(1)_vector address=k'310'
console(1)_max line width=120
console(1)_delete char=rubout

console(2)_device address=k'175630'
console(2)_vector address=k'330'
console(2)_max line width=80
console(2)_delete char=backspace
```

```
console(3)_device address=k'175650'
console(3)_vector address=k'350'
console(3)_max line width=80
console(3)_delete char=backspace

console(4)_device address=k'175700'
console(4)_vector address=k'400'
console(4)_max line width=80
console(4)_delete char=backspace

console(5)_device address=k'175730'
console(5)_vector address=k'430'
console(5)_max line width=120
console(5)_delete char=rubout

%for connum=0,1,highest console %cycle
   memory(console(connum)_vector address>>1)=cinep
   memory((console(connum)_vector address+2)>>1)=intpsw!connum
   memory((console(connum)_vector address+4)>>1)=coutep
   memory((console(connum)_vector address+6)>>1)=intpsw!connum
   condes(connum)_address=console(connum)_device address
   condes(connum)_ident=10+2*connum
   condes(connum)_max line width=console(connum)_max line width
   condes(connum)_delete char=console(connum)_delete char
   condes(connum)_sema number=8+connum
   condes(connum)_dmasignal=0
   condes(connum)_mode=0
   condes(connum)_output address=0
   condes(connum)_output map register=0
   condes(connum)_number of bytes=0
   condes(connum)_line width=0
   condes(connum)_state=0
   condes(connum)_dma output status=0
   condes(connum)_user=0
   glabase=(system size+connum*2)<<9              ;! glabase for this handler
   proctab(8+connum)_r4=glabase                   ;! into processtable
   %for loc=0,2,476 %cycle
      memory((glabase+loc)>>1)=memory((congla+loc)>>1) ;! copy glap to gla
   %repeat
   insert(8+connum,conhan,memory((system size+(connum+1)*2)<<8))
  serv(10+2*connum)=8+connum
  serv(10+2*connum+1)=k'30'
  memory(console(connum)_device address>>1)=k'103'      ;! input status
%repeat

start user memory=system size+(highest console+1)*2
available user memory=available memory-start user memory

fill in page address registers(kernel par)
fill in page descriptor registers(kernel pdr)
fill in page address registers(user par)
fill in page descriptor registers(user pdr)
fill in page address registers(suppar)
fill in page descriptor registers(suppdr)
mem man status 0=1                            ;! enable memory management unit
```

```
   process=0                                ;! select first process to run
   apd==proctab(0)                          ;! set pointer entry in process table
   begrdy=1                                 ;! get it out of ready queue
   kernex                                   ;! exit routine in kernel


   %routine insert(%integer process,%integername program counter,stack pointer)

      proctab(process)_sp==stack pointer
      proctab(process)_pc==program counter
      proctab(process)_psw=userps
      proctab(process)_asleep=notasleep
      %if process<=k'30' %start
         readyq(endrdy)=process             ;! supervisor processes
         endrdy=endrdy+1                    ;! into ready queue
      %finish

   %end

   %routine fill in page address registers (%integerarrayname par)

      %integer paf,reg

      paf=0                                 ;! page address field
      %for reg=0,1,6 %cycle                 ;! virtual addresses in
         par(reg)=paf                       ;! in segments 0-6
         paf=paf+k'200'                     ;! mapped on the same
      %repeat                               ;! physical addresses
      par(7)=k'7600'                        ;! segment 7 mapped on the
                                            ;! peripheral and register addresses
   %end

   %routine fill in page descriptor registers (%integerarrayname pdr)

      %constinteger read and write=6

      %integer reg
      %for reg=0,1,7 %cycle
         pdr(reg)=k'177'<<8!read and write      ;! whole segment mapped
      %repeat

   %end


%endofprogram
```

```
                        .sbttl read symbol
;
001424 013700 readsym:mov     @#instrm,r0     ; address input stream descriptor
       157174
001430 001416         beq     rseoi           ; null stream
001432 021060         cmp     (r0),max(r0)    ; reached then end of the buffer ?
       000002
001436 103403         blo     1$
001440 004770         jsr     pc,@getbuf(r0)  ; yes: get a buffer
       000006
001444 001010         bne     rseoi           ; end of input
001446 117001 1$:     movb    @0(r0),r1       ; next symbol
       000000
01452 042701          bic     #177400,r1      ; ensure only 8 bits
       177400
01456 005210          inc     (r0)            ; step the pointer
01460 012600          mov     (sp)+,r0        ; return address
01462 010136          mov     r1,@(sp)+       ; plant character through pointer
01464 010007          mov     r0,pc
;
01466 012600 rseoi:   mov     (sp)+,r0        ; return address
01470 012736          mov     #100004,@(sp)+  ; plant end of input marker
       100004
01474 010007          mov     r0,pc
;
                        .sbttl next symbol
;
01476 013700 nextsym:mov      @#instrm,r0     ; address input stream descriptor
       157174
01502 001413          beq     nseoi           ; null stream
01504 021060          cmp     (r0),max(r0)    ; reached the end of the buffer ?
       000002
01510 103403          blo     1$
01512 004770          jsr     pc,@getbuf(r0)  ; yes: get a buffer
       000006
01516 001005          bne     nseoi           ; end of input
01520 117001 1$:      movb    @0(r0),r1       ; next symbol
       000000
01524 042701          bic     #177400,r1      ; ensure only 8 bits
       177400
01530 000207          rts     pc
;
01532 012701 nseoi:   mov     #100004,r1      ; end of input marker
       100004
01536 000207          rts     pc
;
```

```
                        .sbttl get buffer from device

001540 010346 getdbuf:mov     r3,-(sp)
001542 010446         mov     r4,-(sp)
001544 010546         mov     r5,-(sp)
001546 012701 1$:     mov     #getinp,r1      ; get input service
       000001
001552 016002         mov     bufblk(r0),r2   ; buffer start block
       000020
001556 016003         mov     buflen(r0),r3   ; buffer length in blocks
       000024
001562 072327         ash     #11,r3          ; buffer length in bytes
       000011
001566 016000         mov     serv(r0),r0     ; service number
       000012
001572 000004         iot
001574 013700         mov     @#instrm,r0     ; address stream descriptor
       157174
001600 005701         tst     r1              ; flag
001602 001424         beq     2$              ; got a buffer
001604 020127         cmp     r1,#1
       000001
001610 001036         bne     deveoi          ; error
001612 012701         mov     #opend,r1       ; flag=1: device not opened
       000000
001616 116002         movb    mode(r0),r2     ; ascii/raw mode
       000014
001622 116003         movb    echo(r0),r3     ; echo/no echo
       000016
001626 016000         mov     serv(r0),r0     ; service number
       000012
001632 000004         iot
001634 013700         mov     @#instrm,r0
       157174
001640 005701         tst     r1              ; if success
001642 001741         beq     1$              ; then try get input again
001644 012700         mov     #fopen,r0       ; else open fault
       000010
001650 000167         jmp     dostop
       001452

001654 005702 2$:     tst     r2              ; end of information ?
001656 001413         beq     deveoi
001660 016010         mov     bufst(r0),(r0)  ; pointer to beginning of buffer
       000022
001664 066002         add     bufst(r0),r2    ; pointer to end of buffer
       000022
001670 010260         mov     r2,max(r0)
       000002
001674 012605         mov     (sp)+,r5
001676 012604         mov     (sp)+,r4
001700 012603         mov     (sp)+,r3
001702 000264         sez                     ; indicate success
001704 000207         rts     pc

001706 012605 deveoi: mov     (sp)+,r5
001710 012604         mov     (sp)+,r4
```

```
41 01712 012603          mov     (sp)+,r3
42 01714 000244          clz                     ; indicate end of information
43 01716 000207          rts     pc
44
```

GET BUFFER FROM FILE

```
                        .sbttl get buffer from file
                ;
001720 010346 getfbuf:mov     r3,-(sp)
001722 010446         mov     r4,-(sp)
001724 010546         mov     r5,-(sp)
001726 066060         add     buflen(r0),block(r0) ; block number
       000024
       000016
001734 012704         mov     #400,r4          ; read shared mode / segment 0
       000400
001740 004767         jsr     pc,connect       ; connect file
       003514
001744 005701         tst     r1               ; if flag=0
01746 001416          beq     2$               ; then got a buffer
01750 020127          cmp     r1,#2            ; if flag=2
       000002
01754 001426          beq     fileoi           ; then end of file
01756 013700          mov     @#instrm,r0      ; try
       157174
01762 005004          clr     r4               ; read unshared mode / segment 0
01764 004767          jsr     pc,connect       ; connect file
       003470
01770 005701          tst     r1               ; if flag=0
01772 001404          beq     2$               ; then got a buffer
01774 012700          mov     #fconnect,r0     ; else connect fault
       000011
02000 000167          jmp     dostop           ; stop the program
       001322
                ;
02004 013700 2$:      mov     @#instrm,r0      ; address stream descriptor
       157174
02010 010210          mov     r2,(r0)          ; pointer to beginning of buffer
02012 060302          add     r3,r2            ; pointer to end of buffer
02014 010260          mov     r2,max(r0)
       000002
02020 012605          mov     (sp)+,r5
02022 012604          mov     (sp)+,r4
02024 012603          mov     (sp)+,r3
02026 000264          sez                      ; indicate success
02030 000207          rts     pc
                ;
02032 013700 fileoi:  mov     @#instrm,r0      ; reset block number
       157174
02036 166060          sub     buflen(r0),block(r0)
       000024
       000016
02044 012605          mov     (sp)+,r5
02046 012604          mov     (sp)+,r4
02050 012603          mov     (sp)+,r3
02052 000244          clz                      ; indicate end of information
02054 000207          rts     pc
                ;
```

```
                        .sbttl loader
                
007000 013765 loader: mov     @#40002,glength(r5); glap length
       040002
       000034
007006 012700         mov     #156000,r0
       156000
007012 163700         sub     @#40002,r0
       040002
007016 163700         sub     @#40004,r0      ; stk length
       040004
007022 010065         mov     r0,glabot(r5)
       000026
007026 013700         mov     @#40000,r0      ; code length
       040000
007032 010001         mov     r0,r1
07034 010065          mov     r0,glapst(r5)   ; glap start in file
       000030
07040 005300          dec     r0
07042 072027          ash     #-11,r0
       177767
07046 005200          inc     r0              ; code length in blocks
07050 010065          mov     r0,codel(r5)
       000032
07054 042701          bic     #177000,r1      ; code length in first glap block
       177000
07060 063701          add     @#40002,r1
       040002
07064 005301          dec     r1
07066 072127          ash     #-11,r1
       177767
07072 005201          inc     r1              ; glap length in blocks
07074 012765          mov     #400+2,par4(r5) ; read shared segment 2
       000402
       000022
07102 072127          ash     #11,r1
       000011
07106 013700          mov     @#40000,r0      ; code length
       040000
07112 072027          ash     #-11,r0         ; glap start in blocks
       177767
07116 050001          bis     r0,r1
07120 010165          mov     r1,par5(r5)     ; glap area in file
       000024
07124 012700          mov     #conct,r0
       000050
07130 004767          jsr     pc,callsup      ; connect it
       003552
07134 001121          bne     ldex            ; if error then exit
07136 016501 4$:      mov     glabot(r5),r1
       000026
07142 006201          asr     r1
07144 042701          bic     #100000,r1
       100000
07150 072127          ash     #-14,r1         ; lowest scratch segment number
       177764
07154 005065          clr     scrcon(r5)
```

```
                000502
34  07160 020127        cmp     r1,#6           ; if scratch file must be connected
          000006
35  07164 001450        beq     1$
36  07166 005265        inc     scrcon(r5)      ; then set flag
          000502
37  07172 010104        mov     r1,r4
38  07174 052704        bis     #1000,r4        ; write unshared mode
          001000
39  07200 010465        mov     r4,par4(r5)
          000022
40  07204 162701        sub     #3,r1           ; lowest possible scratch segment=3
          000003
41  07210 072127        ash     #4,r1           ; offset from begin of scratch file
          000004
42  07214 010165        mov     r1,par5(r5)     ; length = rest
          000024
43  07220 016765        mov     scratch,par1(r5) ; scratch file name
          000156
          000014
44  07226 016765        mov     scratch+2,par2(r5)
          000152
          000016
45  07234 016765        mov     scratch+4,par3(r5)
          000146
          000020
46  07242 066565        add     userid(r5),par3(r5)
          000002
          000020
47  07250 012700        mov     #scrfil,r0
          000044
48  07254 004767        jsr     pc,savnam       ; save file name
          003362
49  07260 012700        mov     #conct,r0
          000050
50  07264 004767        jsr     pc,callsup      ; connect scratch file
          003416
51  07270 001043        bne     ldex            ; if error then exit
52  07272 010704        mov     pc,r4           ; set recovery
53  07274 062704        add     #ldrec-.,r4     ; for loading
          000152
54  07300 010603        mov     sp,r3
55  07302 004767        jsr     pc,setrs
          176732
56  07306 016500 1$:    mov     glabot(r5),r0
          000026
57  07312 005020 2$:    clr     (r0)+           ; clean out gla to sp
58  07314 020006        cmp     r0,sp
59  07316 103775        blo     2$
60  07320 004767        jsr     pc,copgla       ; copy glap to gla
          000064
61  07324 012704        mov     #progfil,r4
          000036
62  07330 060504        add     r5,r4
63  07332 012465        mov     (r4)+,par1(r5)  ; reset code file name
          000014
64  07336 012465        mov     (r4)+,par2(r5)
```

```
                000016
65  07342 011465        mov     (r4),par3(r5)
                000020
66  07346 012765        mov     #400+2,par4(r5) ; read shared in segment 2
                000402
                000022
67  07354 016500        mov     code1(r5),r0
                000032
68  07360 072027        ash     #11,r0
                000011
69  07364 010065        mov     r0,par5(r5)     ; code area in file
                000024
70  07370 012700        mov     #conct,r0
                000050
71  07374 004767        jsr     pc,callsup      ; connect code
                003306
72  07400 000207 ldex:  rts     pc
73
74  07402 073512 scratch:.rad50 /scrfil/        ; scratch file name
    07404 023364
75  07406 073300        .rad50 /s  /
76
77  07410 016500 copgla: mov    glabot(r5),r0   ; virtual gla start
                000026
78  07414 016501        mov     glapst(r5),r1   ; glap start in file
                000030
79  07420 042701        bic     #177000,r1
                177000
80  07424 062701        add     #40000,r1       ; address glap
                040000
81  07430 016502        mov     glength(r5),r2  ; glap length
                000034
82  07434 012120 1$:    mov     (r1)+,(r0)+
83  07436 162702        sub     #2,r2
                000002
84  07442 001374        bne     1$
85  07444 000207        rts     pc
86
87  07446 000244 ldrec: clz                     ; indicate load error
88  07450 000207        rts     pc
```