File created: 13-Jun-2021 11:52:17 **{DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>LLBFS.;2**

changes to: (FNS \ACTONDISKPAGES \WRITEDISKPAGES \VIRTUALDISKDA)

previous date: 17-Dec-92 01:31:53 {DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>LLBFS.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982-1987, 1990, 1992, 2021 by Venue & Xerox Corporation.

(RPAQQ LLBFSCOMS
       [(COMS                                                    ; Low-level subr calls
              (FNS \INITBFS \TESTPARTITION \ACTONDISKPAGES \WRITEDISKPAGES \DISKERROR M44.SIGNAL.DISK.ERROR)
              (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS .SETUPDISKBUFFERS. DISKWRITEACTION? DISKREADACTION?)
                     (PROP DOPVAL .DISKPARTINSTR.)
                     (CONSTANTS * DISKCOMMANDS)
                     (CONSTANTS * DISKERRORS)
                                                               ; Some of these are also used by MOD44IO
                     (RECORDS DISKREQUEST ALTODSKOBJ DDHEADER CB DISKLABEL REALDA SHORTCB FP DSKOBJ
                            \M44LeaderPage M44STREAM FID)
                     (CONSTANTS (\FILLINDA 65534)
                            (\EOFDA 65535)
                            (\LENFP 5)
                            (\FP.DIRECTORYP 32768)
                            (\INITPROPPTR 6866)
                            (\DDBITTABSTART 32)
                            (\NBYTES.DISKINFO 12)
                            (\OFFSET.DISKLASTSERIAL# 8)
                            (\NWORDS.DSKOBJ 36))
                     (GLOBALVARS \EMUDISKBUFEND \EMUDISKBUFFERS \EMUSCRATCH \EMUSWAPBUFFERS \EXTRAISFBUF \ISFMAP
                            \ISFMAXCHUNK \ISFSCRATCHCAS \ISFSCRATCHDAS \#DISKBUFFERS \MAXDISKDAs \#SWAPBUFFERS
                            \SYSDISK \ISFCHUNKSIZE \MAINDISK \DISKREQUESTBLOCK \SWAPREQUESTBLOCK \DISKDEBUG
                            \MAXSWAPBUFFERS \SPAREDISKWRITEBUFFER \FREEPAGEFID \#EMUBUFFERS \EMUBUFFERS
                            \LASTVMEMFILEPAGE \SWAPDSK1 \SWAPDSK2 \XVmemFmapBase \XVmemFmapHighBase
                            \XVmemDiskBase \XVmem \M44.READY)))
        [COMS                                                  ; Super low level
              (FNS \ACTONVMEMPAGES \WRITEVMEMPAGES \DOACTONDISKPAGES \DOWRITEDISKPAGES \CHECKFREEPAGE
                   \DODISKCOMMAND \GETDISKCB \CLEARCB \CLEANUPDISKQUEUE \VIRTUALDISKDA \REALDISKDA)
              (DECLARE%: EVAL@COMPILE DONTCOPY (CONSTANTS * CBSTATUSCONSTANTS)
                     (CONSTANTS * IDISKCOMMANDS)
                     (CONSTANTS (\EM.DISKCOMMAND 337)
                            (\EM.DISKADDRESS 339)
                            (\FIXEDLENDISKREQUEST 42)
                            (\DEFAULTDASTORAGELENGTH 60)
                            (\LENCB 6)
                            (\LENDSKOBJ 34)
                            (\LENSHORTCB 18))
                     (CONSTANTS (\CB.PENDING 1)
                            (\CB.FREE 0]
        [COMS                                                  ; At MAKEINIT time
              (FNS MAKEINITBFS)
              (DECLARE%: DONTCOPY (ADDVARS (INITPTRS (\MAINDISK)
                                                     (\SWAPDSK1)
                                                     (\SWAPDSK2)
                                                     (\SWAPREQUESTBLOCK)
                                                     (\DISKREQUESTBLOCK)
                                                     (\FREEPAGEFID))
                                           (INEWCOMS (FNS MAKEINITBFS)))
                     EVAL@COMPILE
                     (ADDVARS (DONTCOMPILEFNS MAKEINITBFS]
        (COMS                                                  ; Swap stuff
              (FNS \M44ACTONVMEMFILE \LOOKUPFMAP \M44EXTENDVMEMFILE \M44DOEXTENDVMEMFILE \EXTENDISFMAP)
                                                               ; Extended vmem stuff
              (FNS \EXTENDEDVMEMINIT \INITIALIZESWAPDISK \WHICHPART \SWAPDISKERROR)
                                                               ; For debugging and user info
              (FNS DESCRIBE-VIRTUAL-MEMORY \PRINTFMAP)
              (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS ISFMAP)
                     (CONSTANTS (\ISFMAPOFFSET 18)))
              (INITVARS (\DISKDEBUG)
                     (\MAXSWAPBUFFERS 1)
                     (\M44.READY))
              (ADDVARS (\SYSTEMCACHEVARS \M44.READY)))
        (DECLARE%: DONTCOPY
               (ADDVARS (INEWCOMS (ALLOCAL (ADDVARS (LOCKEDFNS ERROR RAID \M44ACTONVMEMFILE \ACTONVMEMFILESUBR
                                                            \ACTONVMEMPAGES \CLEANUPDISKQUEUE \CLEARCB \DISKERROR
                                                            \DOACTONDISKPAGES \DODISKCOMMAND \EXTENDISFMAP
                                                            \M44DOEXTENDVMEMFILE \GETDISKCB \INITBFS
                                                            \INSUREVMEMFILE \LISPERROR \LOOKUPFMAP \REALDISKDA
                                                            \VIRTUALDISKDA \CLEARWORDS \TESTPARTITION
                                                            \EXTENDEDVMEMINIT \WHICHPART \INITIALIZESWAPDISK
                                                            \SWAPDISKERROR)
```

```
                                                (LOCKEDVARS \DISKREQUESTBLOCK \SWAPREQUESTBLOCK \MAINDISK
                                                       \SWAPDSK1 \SWAPDSK2 \ISFCHUNKSIZE \EMUSCRATCH
                                                       \EMUDISKBUFFERS \EMUSWAPBUFFERS \EMUDISKBUFEND
                                                       \MAXSWAPBUFFERS \#DISKBUFFERS \InterfacePage \ISFMAP
                                                       \ISFSCRATCHCAS \ISFSCRATCHDAS \SYSDISK \#SWAPBUFFERS
                                                       \MAXDISKDAs \DISKDEBUG \SPAREDISKWRITEBUFFER
                                                       \#EMUBUFFERS \EMUBUFFERS \LASTVMEMFILEPAGE \XVmem
                                                       \XVmemFmapBase \XVmemFmapHighBase \XVmemDiskBase])
```

;; Low-level subr calls

```
(DEFINEQ
```

(\INITBFS

```
  [LAMBDA (BASE NWORDS AFTER)                                   ; Edited 16-Feb-87 10:26 by Briggs
    (PROG ((DSK \MAINDISK)
           CBSTART CB DD)
```

;; BASE is the start of a chunk of space running for NWORDS words for disk scratch use. Divvy it up as follows: first page as scratch buffer for
;; \WRITEDISKPAGES, then short CB's for disk ops, then miscellaneous scratch for copying DA and CA arrays

```
          (PROGN                                                ; For \WRITEDISKPAGES
                 (SETQ \SPAREDISKWRITEBUFFER BASE)
                 (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
                 (SETQ NWORDS (IDIFFERENCE NWORDS WORDSPERPAGE)))
          [PROGN                                                ; Fill in pieces of \MAINDISK
                 (OR (SETQ \SYSDISK (EMPOINTER (fetch (IFPAGE SYSDISK) of \InterfacePage)))
                     (RAID "Can't find sysDisk"))
                 (SETQ DD (fetch DDHEADER of \SYSDISK))
                 (SETQ CB (SETQ CBSTART (fetch CBQUEUE of DSK)))
                 (do                                            ; Allocate Short CB's for disk controller
                     (replace SHORTCB of CB with BASE)
                     (SETQ BASE (\ADDBASE BASE \LENSHORTCB))
                     (SETQ NWORDS (IDIFFERENCE NWORDS \LENSHORTCB))
                     (SETQ CB (fetch CBNEXT of CB)) repeatuntil (EQ CB CBSTART))
                 (replace ddPOINTER of DSK with (LOCF (fetch (ALTODSKOBJ DDLASTSERIAL#) of \SYSDISK)))
                                                                ; Make some fields indirect thru alto record for now
                 (replace ALTODSKOBJ of DSK with \SYSDISK)
                 (PROGN (replace NDISKS of DSK with (fetch DD#DISKS of DD))
                                                                ; Copy some constant fields from alto
                        (replace NTRACKS of DSK with (fetch DD#TRACKS of DD))
                        (replace NHEADS of DSK with (fetch DD#HEADS of DD))
                        (replace NSECTORS of DSK with (fetch DD#SECTORS of DD)))
                 (replace RETRYCOUNT of DSK with 8)
                 (AND AFTER (replace DDDIRTY of DSK with (replace DDVALID of DSK with NIL]
          (PROGN (SETQ \EMUSCRATCH BASE)
                 (SETQ \MAXDISKDAs (IQUOTIENT (IDIFFERENCE NWORDS (IPLUS \LENFP 8))
                                              2)))
          (COND
             ((SETQ \ISFMAP (EMPOINTER (fetch (IFPAGE ISFMAP) of \InterfacePage)))
              (SETQ \ISFSCRATCHCAS (\ADDBASE [SETQ \ISFSCRATCHDAS (\ADDBASE \ISFMAP (IPLUS (fetch ISFEND
                                                                                               of \ISFMAP)
                                                                                           (PROGN
                                                                ; Leave a little room for off-by-one error in BCPL code
                                                                                                 2]
                                                    (IPLUS (fetch ISFCHUNKSIZE of \ISFMAP)
                                                           2)))
              (SETQ \ISFCHUNKSIZE (fetch ISFCHUNKSIZE of \ISFMAP))
              (replace DISKVERSION of \SWAPREQUESTBLOCK with (fetch FPVERSION of \ISFMAP))
                                                                ; Fill in disk label info for all Lisp.virtualmem requests; this will be
                                                                ; changed by the disk io routine if we are running with extended
                                                                ; virtual memory
              (\BLT (LOCF (fetch DISKSERIAL# of \SWAPREQUESTBLOCK))
                    \ISFMAP WORDSPERCELL)
              (replace RETURNONCHECKERROR of \SWAPREQUESTBLOCK with NIL))
             (T (RAID "No ISF map")))
```

;;; initialization necessary for extended virtual memory feature n.h.briggs 6-feb-87

```
          (\EXTENDEDVMEMINIT])
```

(\TESTPARTITION

```
  [LAMBDA (NUM)                                                 (* bvm%: "13-Feb-85 19:41")
    (PROG ((HERE (.DISKPARTINSTR. 0)))
          (RETURN (COND
                     ((NEQ (.DISKPARTINSTR. (\DTEST NUM 'SMALLP))
                           0)                                   ; Partition switch succeeded, now restore original partition
                      (.DISKPARTINSTR. HERE)
                      T])
```

(\ACTONDISKPAGES

```
  [LAMBDA (DSK BUFFERS DAs DAorigin FID FIRSTPAGE LASTPAGE ACTION LASTNUMCHARSCONS LASTACTION ReturnOnCheckError
        HINTLASTPAGE CAs)                                       ; Edited 13-Jun-2021 11:47 by rmk:
```

;; performs indicated ACTION on pages FIRSTPAGE thru LASTPAGE of DSK (a disk object) for file whose alto id is FID. Returns page number of
;; last page acted on, which may be less than LASTPAGE if end of file was encountered. BUFFERS is either NIL (don't care about the data) or
;; else a buffer or list of buffers of data to be read/written. DAs is a vector of virtual disk addresses (words) for pages of the file, per alto

```
;; conventions.  DAorigin indicates which page (\GETBASE DAs 0) corresponds to;  it should be no greater than FIRSTPAGE.  LASTACTION, if
;; supplied, is performed instead of ACTION on LASTPAGE.  HINTLASTPAGE is hint of the last page of file, to avoid chaining beyond the end of
;; file.  NUMCHARSCONS, if supplied, is a list, car of which will be smashed with the NUMCHARS field of the last page acted on (this in lieu of
;; multiple value return).  If ReturnOnCheckError is true, returns (--- (I + 64)), where I was the last page successfully acted on

(PROG (EMBLOCK EMBUFS EMCAs EMDAs EMFID EMFIXEDCA RESULT STREAM LASTNC)
      (COND
         ((EQ DSK \SYSDISK)
          (SETQ DSK \MAINDISK)))
      [COND
         ((type? STREAM FID)
          (SETQ STREAM FID)
          (SETQ FID (fetch (ARRAYP BASE) of (fetch (M44STREAM FID) of FID]
  RETRY
      (UNINTERRUPTABLY
          (\CLOCK0 (LOCF (fetch (MISCSTATS DISKTEMP0) of \MISCSTATS)))
                                                                  ; Note starting time
          (PROG ((REQUEST \DISKREQUESTBLOCK))
                (.SETUPDISKBUFFERS. ACTION)
                (replace (DISKREQUEST RETURNONCHECKERROR) of REQUEST with ReturnOnCheckError)
                (replace (DISKREQUEST DISKACTION) of REQUEST with ACTION)
                (replace (DISKREQUEST LASTDISKACTION) of REQUEST with (COND
                                                                      ((AND LASTACTION (NEQ LASTACTION 0))
                                                                       LASTACTION)
                                                                      (T ACTION)))
                (SETQ RESULT (\MISCAPPLY* (FUNCTION \DOACTONDISKPAGES)
                                          DSK REQUEST))
                (SETQ LASTNC (fetch (DISKREQUEST CURRENTNUMCHARS) of REQUEST)))
          [COND
             ((AND BUFFERS (NEQ LASTNC BYTESPERPAGE)
                   (IGEQ RESULT 0)
                   (DISKREADACTION? (OR (AND (EQ RESULT LASTPAGE)
                                             LASTACTION)
                                        ACTION)))          ; Zero out everything past the last byte
              (PROG [(BUF (OR EMFIXEDCA (EMPOINTER (\GETBASE EMCAs RESULT]
                    (\CLEARBYTES BUF LASTNC (IDIFFERENCE BYTESPERPAGE LASTNC]
          [COND
             ((NOT (EMADDRESSP DAs))                      ; Possibly update the user's DAs from the emulator copy
              (\BLT DAs EMDAs (IPLUS LASTPAGE 2 (IMINUS DAorigin]
                                                          ; If action was read, now copy from emu buffers into user buffers
          (COND
             ((LISTP BUFFERS)
              (for BUF in BUFFERS as (CA _ EMCAs) by (\ADDBASE CA 1) as N from FIRSTPAGE
                 when (AND BUF (NOT (EMADDRESSP BUF))
                           (DISKREADACTION? (OR (AND (EQ N LASTPAGE)
                                                     LASTACTION)
                                                ACTION)))
                 do (\BLT BUF (\VAG2 0 (\GETBASE CA 0))
                          WORDSPERPAGE)))
             ((AND BUFFERS (NOT (EMADDRESSP BUFFERS))
                   (DISKREADACTION? ACTION))
              (\BLT BUFFERS EMFIXEDCA WORDSPERPAGE)))
          [\BOXIPLUS (LOCF (fetch (MISCSTATS DISKIOTIME) of \MISCSTATS))
                 (\BOXIDIFFERENCE (\CLOCK0 (LOCF (fetch (MISCSTATS DISKTEMP1) of \MISCSTATS)))
                        (LOCF (fetch (MISCSTATS DISKTEMP0) of \MISCSTATS]
                                                          ; Note total time spent here

          )
      (COND
         ((ILESSP RESULT 0)
          (\DISKERROR (IMINUS RESULT)
                 STREAM LASTNC)
          (GO RETRY))
         (T [COND
               (LASTNUMCHARSCONS (COND
                                    ((LISTP LASTNUMCHARSCONS)
                                     (RPLACA LASTNUMCHARSCONS LASTNC))
                                    (T (\PUTBASE LASTNUMCHARSCONS 0 LASTNC]
            (RETURN (SIGNED RESULT BITSPERWORD])
```

(**WRITEDISKPAGES**

```
[LAMBDA (DSK BUFFERS DAs DAorigin FID FIRSTPAGE LASTPAGE LASTACTION LASTNUMCHARSCONS LASTNUMCHARS HINTLASTPAGE
         CAs)                                             ; Edited 13-Jun-2021 11:45 by rmk:

;; Write pages FIRSTPAGE thru LASTPAGE of DSK (a disk object) for file whose alto id is FID.  Returns page number of last page acted on.
;; BUFFERS is either NIL (don't care about the data) or else a buffer or list of buffers of data to be written.  DAs is a vector of virtual disk addresses
;; (words) for pages of the file, per alto conventions.  DAorigin indicates which page (\GETBASE DAs 0) corresponds to;  it should be no greater
;; than FIRSTPAGE.  LASTACTION, if supplied, is performed instead of ACTION on LASTPAGE.  HINTLASTPAGE is hint of the last page of file,
;; to avoid chaining beyond the end of file.  NUMCHARSCONS, if supplied, is a list, car of which will be smashed with the NUMCHARS field of the
;; last page acted on (this in lieu of multiple value return).  LASTNUMCHARS is the nchars field to be written for LASTPAGE

(PROG (EMBLOCK EMBUFS EMCAs EMDAs EMFID EMFIXEDCA RESULT STREAM)
      (COND
         ((EQ DSK \SYSDISK)
          (SETQ DSK \MAINDISK)))
      [COND
         ((type? STREAM FID)
          (SETQ STREAM FID)
```

```
            (SETQ FID (fetch (ARRAYP BASE) of (fetch (M44STREAM FID) of FID]
          (\OPENDISKDESCRIPTOR DSK)
      RETRY
          (UNINTERRUPTABLY
              (\CLOCK0 (LOCF (fetch (MISCSTATS DISKTEMP0) of \MISCSTATS)))
              [SETQ RESULT (PROG ((REQUEST \DISKREQUESTBLOCK))
                                 (.SETUPDISKBUFFERS. \DC.WRITED)
                                 (replace (DISKREQUEST DISKNOALLOC) of REQUEST with (EQ LASTACTION
                                                                                        (UNSIGNED -1 BITSPERWORD))
                                        )
                                 (replace (DISKREQUEST DISKWRITELASTNUMCHARS) of REQUEST with (OR LASTNUMCHARS
                                                                                                  BYTESPERPAGE))
                                 (RETURN (\MISCAPPLY* (FUNCTION \DOWRITEDISKPAGES)
                                                      DSK REQUEST]
              [COND
                 ((NOT (EMADDRESSP DAs))
                  (\BLT DAs EMDAs (IPLUS LASTPAGE 2 (IMINUS DAorigin]
              [\BOXIPLUS (LOCF (fetch DISKIOTIME of \MISCSTATS))
                     (\BOXIDIFFERENCE (\CLOCK0 (LOCF (fetch (MISCSTATS DISKTEMP1) of \MISCSTATS)))
                            (LOCF (fetch (MISCSTATS DISKTEMP0) of \MISCSTATS]
                                                                  (* Note total time spent)
              )
          (COND
             ((ILESSP RESULT 0)
              (\DISKERROR (IMINUS RESULT)
                     STREAM)
              (GO RETRY))
             (T (RETURN (SIGNED RESULT BITSPERWORD])
```

(\DISKERROR

```
  [LAMBDA (ERRCODE STREAM LASTNC)                                  (* bvm%: "12-Mar-85 03:44")
    (COND
       (STREAM (M44.SIGNAL.DISK.ERROR ERRCODE (fetch FULLFILENAME of STREAM)))
       (T (while T do (SELECTC ERRCODE
                           (\DSK.HARD.ERROR
                                (RAID "Hard Disk Error in Lisp.virtualmem.  Page = " (fetch (DSKOBJ CURRENTDISKPAGE
                                                                                                  )
                                                                                       of \MAINDISK)))
                           (\DSK.FULL.ERROR
                                (RAID "Disk Full"))
                           (RAID "Unknown disk error in Lisp.virtualmem" ERRCODE])
```

(M44.SIGNAL.DISK.ERROR

```
  [LAMBDA (ERRCODE FILENAME)                                       (* bvm%: "17-Jan-85 17:46")
    ;; This is a separate function, without a backslash, so that user can OK from the break, in which case we return OK from here, telling caller to try
    ;; again
    (COND
       [ERRCODE (bind (EC _ (PROG1 ERRCODE (SETQ ERRCODE NIL))) while T
                   do (SELECTC EC
                           (\DSK.HARD.ERROR
                                (LISPERROR "HARD DISK ERROR" FILENAME T))
                           (\DSK.FULL.ERROR
                                (LISPERROR "FILE SYSTEM RESOURCES EXCEEDED" FILENAME T))
                           (ERROR "Disk Error" FILENAME]
       (T 'OK])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS .SETUPDISKBUFFERS. MACRO
          ((ACTION)                                                (* bvm%: "15-OCT-82 17:28")
           (SETQ EMBUFS \EMUDISKBUFFERS)
           (SETQ EMBLOCK \EMUSCRATCH)
           [COND
              ((ILESSP DAorigin (SUB1 FIRSTPAGE))
               [SETQ DAs (\ADDBASE DAs (SUB1 (IDIFFERENCE FIRSTPAGE DAorigin]
               [AND CAs (SETQ CAs (\ADDBASE CAs (SUB1 (IDIFFERENCE FIRSTPAGE DAorigin]
               (SETQ DAorigin (SUB1 FIRSTPAGE]
           [SETQ EMDAs (COND
                          ((EMADDRESSP DAs)
                           DAs)
                          (T (\BLT EMBLOCK DAs (IPLUS LASTPAGE 2 (IMINUS DAorigin)))
                             (PROG1 EMBLOCK
                                 [SETQ EMBLOCK (\ADDBASE EMBLOCK (IPLUS LASTPAGE 2 (IMINUS DAorigin])]
           [COND
              [CAs (SETQ EMCAs (\ADDBASE CAs (IMINUS DAorigin]
              [(AND (LISTP BUFFERS)
                    (OR (CDR BUFFERS)
                        (PROGN (SETQ BUFFERS (CAR BUFFERS))      ; Treat singleton BUFFER as nonlist
                               NIL)))
               (SETQ EMCAs (\ADDBASE EMBLOCK (IMINUS FIRSTPAGE)))
```

```
               (for BUF in BUFFERS as N from FIRSTPAGE bind FIXEDBUF
                  do [\PUTBASE EMBLOCK 0 (COND
                                     ((AND BUF (EMADDRESSP BUF))
                                      (\LOLOC BUF))
                                     ((AND (NULL BUF)
                                           FIXEDBUF))
                                     (T (PROG1 (\LOLOC EMBUFS)
                                               [COND
                                                  ((DISKWRITEACTION? (OR (AND (EQ N LASTPAGE)
                                                                              LASTACTION)
                                                                         ACTION))
                                                   (COND
                                                      (BUF (\BLT EMBUFS BUF WORDSPERPAGE))
                                                      (T (\CLEARWORDS EMBUFS WORDSPERPAGE)
                                                         (SETQ FIXEDBUF (\LOLOC EMBUFS]
                                                  (SETQ EMBUFS (\ADDBASE EMBUFS WORDSPERPAGE))
                                                  (COND
                                                     ((PTRGTP EMBUFS \EMUDISKBUFEND)
                                                      (ERROR "Attempt to act on too many disk pages")))]
                     (SETQ EMBLOCK (\ADDBASE EMBLOCK 1]
               (T (SETQ EMFIXEDCA (COND
                                  ((AND BUFFERS (EMADDRESSP BUFFERS))
                                   BUFFERS)
                                  (T [COND
                                        ((DISKWRITEACTION? ACTION)
                                                                  ; If writing, copy data into buffer
                                         (COND
                                            (BUFFERS (\BLT EMBUFS BUFFERS WORDSPERPAGE))
                                            (T (\CLEARWORDS EMBUFS WORDSPERPAGE]
                                     EMBUFS]
            (replace DISKCAS of REQUEST with EMCAs)
            (replace FIXEDDISKBUFFER of REQUEST with EMFIXEDCA)
            (replace DISKDAS of REQUEST with (\ADDBASE EMDAs (IMINUS DAorigin)))
            (replace DISKVERSION of REQUEST with (fetch FPVERSION of FID))
            (\BLT (LOCF (fetch DISKSERIAL# of REQUEST))
                  FID WORDSPERCELL)                           ; Fill in serial number for label
            (replace DISKFIRSTPAGE of REQUEST with FIRSTPAGE)
            (replace DISKLASTPAGE of REQUEST with LASTPAGE)
            (replace DISKHINTLASTPAGE of REQUEST with (OR HINTLASTPAGE 0))))

(PUTPROPS DISKWRITEACTION? MACRO ((ACTION)                    (* bvm%: "15-OCT-82 17:06")
                                  (SELECTC ACTION
                                       ((LIST \DC.WRITEHLD \DC.WRITELD \DC.WRITED)
                                          T)
                                       NIL)))

(PUTPROPS DISKREADACTION? MACRO ((ACTION)                     (* bvm%: "15-OCT-82 17:06")
                                 (ILESSP ACTION \DC.WRITEHLD)))
)

(PUTPROPS .DISKPARTINSTR. DOPVAL (1 SUBRCALL 8 1))

(RPAQQ DISKCOMMANDS ((\DC.READHLD 54784)
                     (\DC.READLD 54785)
                     (\DC.READD 54786)
                     (\DC.WRITEHLD 54787)
                     (\DC.WRITELD 54788)
                     (\DC.WRITED 54789)
                     (\DC.SEEKONLY 54790)
                     (\DC.NOOP 54791)
                     (\DC.RESTORE 54891)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \DC.READHLD 54784)

(RPAQQ \DC.READLD 54785)

(RPAQQ \DC.READD 54786)

(RPAQQ \DC.WRITEHLD 54787)

(RPAQQ \DC.WRITELD 54788)

(RPAQQ \DC.WRITED 54789)

(RPAQQ \DC.SEEKONLY 54790)

(RPAQQ \DC.NOOP 54791)

(RPAQQ \DC.RESTORE 54891)

(CONSTANTS (\DC.READHLD 54784)
       (\DC.READLD 54785)
       (\DC.READD 54786)
       (\DC.WRITEHLD 54787)
       (\DC.WRITELD 54788)
```

```
       (\DC.WRITED 54789)
       (\DC.SEEKONLY 54790)
       (\DC.NOOP 54791)
       (\DC.RESTORE 54891))
)

(RPAQQ DISKERRORS ((\DSK.HARD.ERROR 1101)
                   (\DSK.FULL.ERROR 1102)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \DSK.HARD.ERROR 1101)

(RPAQQ \DSK.FULL.ERROR 1102)

(CONSTANTS (\DSK.HARD.ERROR 1101)
       (\DSK.FULL.ERROR 1102))
)

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD DISKREQUEST ((DISKDAS FULLXPOINTER)                ; Vector of DAs to be acted on
                          (DISKCAS FULLXPOINTER)
                          (FIXEDDISKBUFFER FULLXPOINTER)
                          (DISKERRORCODE FULLXPOINTER)
                          (CBCLEANUPFN FULLXPOINTER)
                          (DISKFIRSTPAGE WORD)
                          (DISKLASTPAGE WORD)
                          (DISKHINTLASTPAGE WORD)
                          (DISKVERSION WORD)                    ; These 3 words are for the disk label
                          (DISKSERIAL# FIXP)
                          (DISKACTION WORD)
                          (LASTDISKACTION WORD)
                          (CURRENTNUMCHARS WORD)
                          (LASTPAGEACTEDON WORD)
                          (DISKWRITELASTNUMCHARS WORD)
                          (RETURNONCHECKERROR FLAG)
                          (DISKNOALLOC FLAG)
                          (NIL BITS 14)
                          (DISKCASTORAGE 20 WORD)
                          (DISKDASTORAGE 60 WORD)               ; Or as much as you want

                         )
       [ACCESSFNS ((DISKFID (LOCF (fetch DISKVERSION of DATUM]
       (CREATE (\ALLOCBLOCK (FOLDHI (IPLUS \FIXEDLENDISKREQUEST 60)
                                    WORDSPERCELL]

[BLOCKRECORD ALTODSKOBJ ((NIL 8 WORD)                           ; Alto functions to implement generic ops
                         (DSKFPSYSDIR WORD)                     ; Short pointer to SYSDIR FP
                         (NIL 2 WORD)                           ; More alto fns
                         (DSKFPWORKINGDIR WORD)                 ; Short pointer to FP of 'working' dir
                         (DSKNAMEWORKINGDIR WORD)               ; Short pointer to bcpl string
                         (DSKLNPAGESIZE WORD)                   ; ln[pagesize-in-words]
                         (NIL 3 WORD)                           ; More alto cruft
                         (DSKKD WORD)                           ; Short pointer to DDHEADER
                         (DSKFPDISKDESCRIPTOR WORD)             ; Short pointer to DiskDescriptor FP
                         (DSKDRIVE# WORD)
                         (DSKRETRYCOUNT WORD)
                         (DSKTOTALERRORS WORD)
                         (DSKLENCBZ WORD)
                         (DSKLENCB WORD)
                         (DSKDDHEADER 16 WORD)                  ; Overlays DDHEADER
                         (NIL 2 WORD)
                         (DSKDDMGR WORD)                        ; DD manager, for \FLUSHDISKDESCRIPTOR
                         (DSKLASTVDA WORD)                      ; VDA of last page allocated, for biasing search
                         (DSKSYSDIRBLK 5 WORD)
                         (DSKDDBLK 5 WORD)
                         (DSKWDBLK 5 WORD)
                         (NIL 20 WORD)                          ; WorkingDir name
                         (DSKDDVDAS 17 WORD)                    ; VDAs for the data part of DD

                        )
       (ACCESSFNS ALTODSKOBJ ((DDHEADER (LOCF (fetch DSKDDHEADER of DATUM]

(BLOCKRECORD DDHEADER ((DD#DISKS WORD)
                       (DD#TRACKS WORD)
                       (DD#HEADS WORD)
                       (DD#SECTORS WORD)
                       (DDLASTSERIAL# FIXP)
                       (NIL WORD)
                       (DDBTSIZE WORD)                          ; Size of bittable in words
                       (DDDEFAULTVERSIONSKEPT WORD)
                       (DDFREEPAGES WORD)
                       (NIL 6 WORD)))

(BLOCKRECORD CB ((CBQSTATUS BITS 4)
                 (CBNEXT POINTER)                               ; Link to next one
```

```
                (SHORTCB POINTER)                       ; In alto space, what disk actually uses
                (CBPAGENO WORD)                         ; The page number we intended to act on with this CB

                )
        (CREATE (\ALLOCBLOCK 3)))

(BLOCKRECORD DISKLABEL ((DLNEXT WORD)
                        (DLPREVIOUS WORD)
                        (NIL WORD)
                        (DLNUMCHARS WORD)
                        (DLPAGENO WORD)
                        (DLFID 3 WORD)                  ; Version followed by 2-word serial number

                        ))

[ACCESSFNS REALDA ((SECTOR (LRSH DATUM 12))
                   (TRACK (LOGAND (LRSH DATUM 3)
                                  511))
                   (HEAD (LOGAND (LRSH DATUM 2)
                                 1))
                   (DISK (LOGAND (LRSH DATUM 1)
                                 1))
                   (RESTORE (LOGAND DATUM 1]

[BLOCKRECORD SHORTCB ((CBLINK WORD)                     ; Short pointer to next in command chain, or zero
                      (CBSTATUS WORD)
                      (CBCOMMAND WORD)
                      (CBHEADERADDR WORD)               ; Short pointer to header record, normally CBHEADER
                      (CBLABELADDR WORD)                ; Short pointer to label record, normally either in this CB or in the
                                                        ; next cb in chain
                      (CBDATAADDR WORD)                 ; Short pointer to buffer of data
                      (CBWAKEUPS WORD)                  ; These two are always zero
                      (CBERRWAKEUPS WORD)
                      (CBHEADER WORD)
                      (CBDA WORD)                       ; Address of this disk block.  May be filled in by previous
                                                        ; access's label pointing at my CBHEADER
                      (CBLABNEXT WORD)                  ; Start of label field, if my CBLABELADDR points here
                      (CBLABPREV WORD)
                      (CBLABBLANK WORD)
                      (CBLABNUMCHARS WORD)
                      (CBLABPAGENO WORD)
                      (CBLABVERSION WORD)
                      (CBLABSN1 WORD)
                      (CBLABSN2 WORD)
                      (CBTRUEPAGENO WORD)               ; From here on is alto stuff that Lisp doesn't care about
                      (CBCBZ WORD)
                      (CBNEXTSHORTCB WORD))
       (BLOCKRECORD SHORTCB ((NIL WORD)
                             (CBSECTOR BITS 4)
                             (CBDONE BITS 4)
                             (CBSEEKFAIL BITS 1)
                             (CBSEEKING BITS 1)
                             (CBNOTREADY BITS 1)
                             (CBDATALATE BITS 1)
                             (CBNOTRANSFER BITS 1)
                             (CBCHECKSUMERR BITS 1)
                             (CBFINALSTATUS BITS 2)
                             (CBSEAL BYTE)
                             (CBACTION BYTE)))
       (BLOCKRECORD SHORTCB ((NIL WORD)
                             (NIL WORD)
                             (CBSHORTSEAL BITS 5)
                             (CBPARTITION BITS 3)
                             (NIL BYTE]

[BLOCKRECORD FP ((FPSERIAL# FIXP)
                 (FPVERSION WORD)
                 (NIL WORD)
                 (FPLEADERVDA WORD))
       (BLOCKRECORD FP ((FPSERIALHI WORD)
                        (FPSERIALLO WORD)
                        (NIL 3 WORD]

[BLOCKRECORD DSKOBJ ((ddPOINTER FULLXPOINTER)
                     ;; Either points at word 2 of this structure, or at DSKOBJ:LASTSERIAL#, so that we can maintain some fields in parallel with
                     ;; alto OS for awhile.  The next 6 words are arranged exactly as in the alto KDH structure, at least those fields we care about
                     (ddLASTSERIAL# FIXP)               ; Last serial number given a file
                     (NIL WORD)
                     (ddBITTABLESIZE WORD)              ; Size of disk descriptor's bit table in words
                     (NIL WORD)
                     (ddFREEPAGES WORD)
                     (DSKPARTITION BITS 4)              ; 0 or explicit partition pointer
                     (ALTODSKOBJ XPOINTER)              ; Pointer to alto BFSDSK structure, or NIL for disks other than
                                                        ; current partition
                     (SAWCHECKERROR FLAG)
                     (DISKERRORCNT BITS 3)
```

```
                    (SYSDIROFD POINTER)                         ; Stream onto SYSDIR.;1
                    (DDDIRTY FLAG)                              ; true if diskdescriptor needs writing
                    (DDVALID FLAG)                              ; True if DISKDESCRIPTOROFD field is ok.  Invalidated on
                                                                ; logout, etc
                    (DSKPASSWORDOK FLAG)                        ; True after password for this partition, if any, has been validated
                    (NIL BITS 1)
                    (DISKDESCRIPTOROFD POINTER)                 ; Stream onto DiskDescriptor.;1
                    (CBQUEUE POINTER)                           ; Stuff for management of command blocks.  No ref count
                                                                ; because must not fault
                    (CBFREEPTR FULLXPOINTER)
                    (CBPENDINGPTR FULLXPOINTER)
                    (CBLASTPTR FULLXPOINTER)
                    (CURRENTDAS FULLXPOINTER)                   ; Vector of DAs currently being acted on
                    (DISKREQUEST FULLXPOINTER)
                    (DISKDEVICENAME POINTER)                    ; For retrieving the FDEV
                    (DISKLASTPAGEALLOC WORD)                    ; Bias for new page search
                                                                ; Pointer to request subrecord
                    (CURRENTDISKPAGE WORD)
                    (TOTALDISKERRORS WORD)
                    (NDISKS WORD)                               ; Shape of disk.  Info taken from disk descriptor
                    (NTRACKS WORD)
                    (NHEADS WORD)
                    (NSECTORS WORD)
                    (RETRYCOUNT WORD))
      (CREATE (\ALLOCBLOCK 18))
      ddPOINTER _ NIL (BLOCKRECORD ddPOINTER ((DISKLASTSERIAL# FIXP)
                                              (NIL WORD)
                                              (DISKBITTABLESIZE WORD)
                                              (NIL WORD)
                                              (DISKFREEPAGES WORD]

[BLOCKRECORD \M44LeaderPage ((TimeCreate FIXP)
                             (TimeWrite FIXP)
                             (TimeRead FIXP)
                             (NameCharCount BYTE)
                             (NameChars 39 BYTE)
                             (LeaderProps 210 WORD)
                             (Spares 10 WORD)
                             (PropertyPtr WORD)
                             (ConsecutiveHint FLAG)
                             (NIL BITS 7)
                             (ChangeSerialNumber BYTE)
                             (FIDDirectoryHint 5 WORD)
                             (LastPageAddress WORD)
                             (LastPageNumber WORD)
                             (LastPageByteCount WORD))
      (BLOCKRECORD \M44LeaderPage ((NIL WORD)
                                   (TimeCreateLo WORD)
                                   (NIL WORD)
                                   (TimeWriteLo WORD)
                                   (NIL FIXP)
                                   (NIL 20 WORD)
                                   (NIL 210 WORD)
                                   (NIL 10 WORD)
                                   (PropertyBegin BYTE)
                                   (PropertyLength BYTE)))
      (CREATE (NCREATE 'VMEMPAGEP]

(ACCESSFNS M44STREAM ((FID (fetch F1 of DATUM)
                           (replace F1 of DATUM with NEWVALUE))
                      (FILEPAGEMAP (fetch F2 of DATUM)
                             (replace F2 of DATUM with NEWVALUE))
                      (LASTMAPPEDPAGE (fetch F3 of DATUM)
                             (replace F3 of DATUM with NEWVALUE))
                      (DIRINFO (fetch F4 of DATUM)
                             (replace F4 of DATUM with NEWVALUE))
                      (LEADERPAGE (fetch F5 of DATUM)
                             (replace F5 of DATUM with NEWVALUE))
                      (LastPage (fetch FW6 of DATUM)
                             (replace FW6 of DATUM with NEWVALUE))
                      (LastOffset (fetch FW7 of DATUM)
                             (replace FW7 of DATUM with NEWVALUE)))
      (ACCESSFNS M44STREAM ((DIRHOLEPTR (fetch F4 of DATUM)
                                   (replace F4 of DATUM with NEWVALUE))
                                                                ; In dir stream only

                            ))
      (CREATE (create STREAM))
      LASTMAPPEDPAGE _ -1 LastPage _ 0 LastOffset _ 0)

(ACCESSFNS FID ((W0 (\WORDELT DATUM 0)
                    (SETA DATUM 0 NEWVALUE))
                (W1 (\WORDELT DATUM 1)
                    (SETA DATUM 1 NEWVALUE))
                (W2 (\WORDELT DATUM 2)
                    (SETA DATUM 2 NEWVALUE))
                (W3 (\WORDELT DATUM 3)
```

```
                    (SETA DATUM 3 NEWVALUE))
                (W4 (\WORDELT DATUM 4)
                    (SETA DATUM 4 NEWVALUE))
                (FIDBLOCK (fetch (ARRAYP BASE) of DATUM)))
        (CREATE (ARRAY 5 'SMALLPOSP 0 0)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \FILLINDA 65534)

(RPAQQ \EOFDA 65535)

(RPAQQ \LENFP 5)

(RPAQQ \FP.DIRECTORYP 32768)

(RPAQQ \INITPROPPTR 6866)

(RPAQQ \DDBITTABSTART 32)

(RPAQQ \NBYTES.DISKINFO 12)

(RPAQQ \OFFSET.DISKLASTSERIAL# 8)

(RPAQQ \NWORDS.DSKOBJ 36)

(CONSTANTS (\FILLINDA 65534)
       (\EOFDA 65535)
       (\LENFP 5)
       (\FP.DIRECTORYP 32768)
       (\INITPROPPTR 6866)
       (\DDBITTABSTART 32)
       (\NBYTES.DISKINFO 12)
       (\OFFSET.DISKLASTSERIAL# 8)
       (\NWORDS.DSKOBJ 36))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \EMUDISKBUFEND \EMUDISKBUFFERS \EMUSCRATCH \EMUSWAPBUFFERS \EXTRAISFBUF \ISFMAP \ISFMAXCHUNK
       \ISFSCRATCHCAS \ISFSCRATCHDAS \#DISKBUFFERS \MAXDISKDAs \#SWAPBUFFERS \SYSDISK \ISFCHUNKSIZE \MAINDISK
       \DISKREQUESTBLOCK \SWAPREQUESTBLOCK \DISKDEBUG \MAXSWAPBUFFERS \SPAREDISKWRITEBUFFER \FREEPAGEFID
       \#EMUBUFFERS \EMUBUFFERS \LASTVMEMFILEPAGE \SWAPDSK1 \SWAPDSK2 \XVmemFmapBase \XVmemFmapHighBase
       \XVmemDiskBase \XVmem \M44.READY)
)
)
```

;; Super low level

```
(DEFINEQ
```

(\ACTONVMEMPAGES

```
  [LAMBDA (DSK BUFFERS DAs DAorigin FIRSTPAGE LASTPAGE ACTION LASTACTION HINTLASTPAGE CAs)
                                                                (* bvm%: "13-Feb-85 19:00")
    (PROG ((REQUEST \SWAPREQUESTBLOCK))
          (replace FIXEDDISKBUFFER of REQUEST with (COND
                                                   (CAs (replace DISKCAS of REQUEST with (\ADDBASE CAs
                                                                                                (IMINUS DAorigin))
                                                                                         )
                                                        NIL)
                                                   (T BUFFERS)))
          (replace DISKDAS of REQUEST with (\ADDBASE DAs (IMINUS DAorigin)))
          (replace DISKFIRSTPAGE of REQUEST with FIRSTPAGE)
          (replace DISKLASTPAGE of REQUEST with LASTPAGE)
          (replace DISKHINTLASTPAGE of REQUEST with (OR HINTLASTPAGE 0))
          (replace DISKACTION of REQUEST with ACTION)
          (replace LASTDISKACTION of REQUEST with (COND
                                                  ((AND LASTACTION (NEQ LASTACTION 0))
                                                   LASTACTION)
                                                  (T ACTION)))
          (RETURN (\DOACTONDISKPAGES DSK REQUEST])
```

(\WRITEVMEMPAGES

```
  [LAMBDA (DSK BUFFERS DAs DAorigin FID FIRSTPAGE LASTPAGE LASTNUMCHARS HINTLASTPAGE CAs)
                                                                (* bvm%: "13-Feb-85 19:02")
```

;;; \WRITEDISKPAGES inside the junk context. Used by \M44DOEXTENDVMEMFILE

```
    (CHECK (NOT \INTERRUPTABLE))
    (PROG ((REQUEST \DISKREQUESTBLOCK))
          [COND
             (CAs (replace DISKCAS of REQUEST with (\ADDBASE CAs (IMINUS DAorigin]
          (replace FIXEDDISKBUFFER of REQUEST with BUFFERS)
          (replace DISKDAS of REQUEST with (\ADDBASE DAs (IMINUS DAorigin)))
          (replace DISKVERSION of REQUEST with (fetch FPVERSION of FID))
```

```
          (\BLT (LOCF (fetch DISKSERIAL# of REQUEST))
                FID WORDSPERCELL)                                    ; Fill in serial number for label
          (replace DISKFIRSTPAGE of REQUEST with FIRSTPAGE)
          (replace DISKLASTPAGE of REQUEST with LASTPAGE)
          (replace DISKHINTLASTPAGE of REQUEST with (OR HINTLASTPAGE 0))
          (replace DISKNOALLOC of REQUEST with NIL)
          (replace DISKWRITELASTNUMCHARS of REQUEST with (OR LASTNUMCHARS BYTESPERPAGE))
          (RETURN (\DOWRITEDISKPAGES DSK REQUEST])
```

(\DOACTONDISKPAGES

```
  [LAMBDA (DSK REQUEST CLEANUPFN)                                    (* bvm%: "30-DEC-82 13:08")
    (PROG ((CAS (fetch DISKCAS of REQUEST))
           (DAS (fetch DISKDAS of REQUEST))
           (FIRSTPAGE (fetch DISKFIRSTPAGE of REQUEST))
           (LASTPAGE (fetch DISKLASTPAGE of REQUEST))
           (BUFFER (fetch FIXEDDISKBUFFER of REQUEST))
           (HINTLASTPAGE (fetch DISKHINTLASTPAGE of REQUEST))
           (RETURNONCHECKERROR (fetch RETURNONCHECKERROR of REQUEST))
           CURRENTPAGE CB NEXTCB RESULT THISACTION)
          (replace DISKREQUEST of DSK with REQUEST)
          (replace CURRENTDAS of DSK with DAS)
          (replace DISKERRORCNT of DSK with 0)
          (replace SAWCHECKERROR of DSK with NIL)
          (replace CBCLEANUPFN of REQUEST with CLEANUPFN)
          (COND
             ((OR (NOT HINTLASTPAGE)
                  (ILESSP HINTLASTPAGE FIRSTPAGE)
                  (IGREATERP HINTLASTPAGE LASTPAGE))
              (SETQ HINTLASTPAGE LASTPAGE)))
          (SETQ CURRENTPAGE FIRSTPAGE)
```

;; HINTLASTPAGE is used, if reasonable, to terminate activity before LASTPAGE so that we do not chain off the end and seek to cylinder 0, a
;; typical BFS bug. If the hint is wrong, we just resume from there, having wasted a disk rotation

```
          RETRY
          (replace CBLASTPTR of DSK with NIL)
          (replace CBFREEPTR of DSK with (replace CBPENDINGPTR of DSK with (fetch CBQUEUE of DSK)))
          (SETQ CB (\GETDISKCB DSK))                                 ; Should never return NIL
          (SETQ RESULT HINTLASTPAGE)
          (for PAGENO from CURRENTPAGE to HINTLASTPAGE until (COND
                                                               ((EQ (\GETBASE DAS PAGENO)
                                                                    \EOFDA)
                                                                ; At end of file, do no more
                                                                (SETQ RESULT (SUB1 PAGENO))
                                                                T))
             do [SETQ THISACTION (COND
                                   ((EQ PAGENO LASTPAGE)
                                    (fetch LASTDISKACTION of REQUEST))
                                   (T (fetch DISKACTION of REQUEST]
                [COND
                   ((AND RETURNONCHECKERROR (fetch SAWCHECKERROR of DSK))
                                                                    ; No disk activity now, because cleanup waited for disk to stop
                    (replace LASTPAGEACTEDON of REQUEST with (SUB1 PAGENO))
                    (RETURN (SETQ RESULT (LOGAND (IMINUS (IPLUS PAGENO -1 64))
                                                 65535]
                (COND
                   ((NEQ THISACTION \DC.NOOP)
                    (COND
                       ((NULL (SETQ NEXTCB (\GETDISKCB DSK)))
                        (GO FAILURE)))
                    [replace (CB CBLABELADDR) of CB with (\LOLOC (COND
                                                                   ((EQ (\GETBASE DAS (ADD1 PAGENO))
                                                                        \FILLINDA)
                                                                    ;; Chain to next cb, so that the next field from the label of CB is written into
                                                                    ;; the diskaddress field of NEXTCB
                                                                    (LOCF (fetch (CB CBDA) of NEXTCB)))
                                                                   (T (LOCF (fetch (CB CBLABNEXT) of NEXTCB]
                    (\DODISKCOMMAND DSK CB (OR BUFFER (EMPOINTER (\GETBASE CAS PAGENO)))
                           (\GETBASE DAS PAGENO)
                           PAGENO THISACTION)
                    (SETQ CB NEXTCB)))
             finally (COND
                        ((AND (fetch CBFREEPTR of DSK)
                              (NEQ (fetch CBFREEPTR of DSK)
                                   (fetch CBNEXT of CB)))
                         (RAID "Inconsistency in CBFREEPTR" CB)))
                     (replace CBFREEPTR of DSK with CB)             ; 'Put back' the last CB, since it is never used for a command
                     (do                                            ; Wait for commands to complete
                         (SELECTQ (\CLEANUPDISKQUEUE DSK)
                              (NIL (GO FAILURE))
                              (T (RETURN))
                              NIL))
                     (COND
                        ((AND (NEQ RESULT LASTPAGE)
                              (NEQ (\GETBASE DAS (ADD1 RESULT))
                                   \EOFDA))                         ; Stopped before LASTPAGE because of a bad hint. Ignore hint
                                                                    ; and continue
```

```
                    (SETQ HINTLASTPAGE LASTPAGE)
                    (SETQ CURRENTPAGE (ADD1 RESULT))
                    (GO RETRY)))
                (replace LASTPAGEACTEDON of REQUEST with RESULT))
        (RETURN RESULT)
    FAILURE
        (COND
           ((ILEQ (fetch DISKERRORCNT of DSK)
                  (fetch RETRYCOUNT of DSK))
            (SETQ CURRENTPAGE (fetch CURRENTDISKPAGE of DSK))
            (GO RETRY)))
        (RETURN (IMINUS \DSK.HARD.ERROR])
```

(**DOWRITEDISKPAGES**

```
[LAMBDA (DSK REQUEST)                                   (* bvm%: "17-Jan-85 17:06")
  (PROG ((CAS (fetch DISKCAS of REQUEST))
         (DAS (fetch DISKDAS of REQUEST))
         (FIRSTPAGE (fetch DISKFIRSTPAGE of REQUEST))
         (LASTPAGE (fetch DISKLASTPAGE of REQUEST))
         (BUFFER (fetch FIXEDDISKBUFFER of REQUEST))
         CURRENTPAGE CB FIRSTNEWPAGE NEXTNEWPAGE LASTVDA LAB)
        [COND
           ((NOT (fetch DISKNOALLOC of REQUEST))        ; First try \ACTONDISKPAGES for any existing pages
            [COND
               ((EQ (\GETBASE DAS FIRSTPAGE)
                    \FILLINDA)                          ; This happens from createfile, no pages exist yet
                (SETQ FIRSTNEWPAGE FIRSTPAGE))
               (T [COND
                     ((EQ (\GETBASE DAS (ADD1 FIRSTPAGE))
                          \EOFDA)
                      ;; FIRSTPAGE is the last existing page of the file, so we will have to rewrite its label anyway later on, so don't do
                      ;; anything now

                      )
                     (T ;; Some of these pages may not need to have their labels written, so see how far we can get with just
                        ;; \ACTONDISKPAGES ...
                        (replace DISKACTION of REQUEST with (replace LASTDISKACTION of REQUEST with \DC.WRITED))
                        (replace RETURNONCHECKERROR of REQUEST with NIL)
                        (SETQ FIRSTPAGE (\DOACTONDISKPAGES DSK REQUEST))
                        (COND
                           ((AND (EQ FIRSTPAGE LASTPAGE)
                                 (EQ (fetch DISKWRITELASTNUMCHARS of REQUEST)
                                     (fetch CURRENTNUMCHARS of REQUEST)))
                                                        ; All pages acted on, and byte count does not need to be
                                                        ; changed
                            (RETURN LASTPAGE))
                           ((ILESSP FIRSTPAGE 0)        ; Error
                            (RETURN FIRSTPAGE]
                  (SETQ FIRSTNEWPAGE (ADD1 FIRSTPAGE]
            (COND
               ((ILEQ FIRSTNEWPAGE LASTPAGE)
                ;; Need to allocate new pages.  For this, we need a spare buffer for reading the new pages to make sure they are free pages
                (replace FIXEDDISKBUFFER of REQUEST with \SPAREDISKWRITEBUFFER)
                (replace DISKACTION of REQUEST with (replace LASTDISKACTION of REQUEST with \DC.READLD))
                (SETQ NEXTNEWPAGE FIRSTNEWPAGE)
                (do (SETQ LASTVDA (\GETBASE DAS (SUB1 NEXTNEWPAGE)))
                    (for I from NEXTNEWPAGE to LASTPAGE do (COND
                                                              ((NOT (SETQ LASTVDA (\ASSIGNDISKPAGE DSK LASTVDA)
                                                                    ))
                                                               ; Disk full.  Back out the pages we have assigned so far
                                                               (for J from FIRSTNEWPAGE to (SUB1 I)
                                                                  do (\M44MARKPAGEFREE DSK (\GETBASE DAS J))
                                                                     (\PUTBASE DAS J \FILLINDA))
                                                               (GO DISKFULL)))
                                                            (\PUTBASE DAS I LASTVDA))
                                                               ; Now check that the pages are really free
                    (replace DISKFIRSTPAGE of REQUEST with NEXTNEWPAGE)
                    (\DOACTONDISKPAGES DSK REQUEST (FUNCTION \CHECKFREEPAGE))
                   ;; \CHECKFREEPAGE checks to make sure the page is really free, and if it is, stores its address in DAS.  We now march
                   ;; thru DAS, compacting toward the front all the pages that are really free, then iterate allocating more if necessary
                    (for I from NEXTNEWPAGE to LASTPAGE when (NEQ (SETQ LASTVDA (\GETBASE DAS I))
                                                                  \FILLINDA)
                       do (\PUTBASE DAS NEXTNEWPAGE LASTVDA)
                          (add NEXTNEWPAGE 1))
                  repeatuntil (IGREATERP NEXTNEWPAGE LASTPAGE]
        (replace DISKREQUEST of DSK with REQUEST)
        (replace CURRENTDAS of DSK with DAS)
        (replace DISKERRORCNT of DSK with 0)
        (replace SAWCHECKERROR of DSK with NIL)
        (replace CBCLEANUPFN of REQUEST with NIL)
        (SETQ CURRENTPAGE FIRSTPAGE)
    RETRY
        (replace CBLASTPTR of DSK with NIL)
```

```
        (replace CBFREEPTR of DSK with (replace CBPENDINGPTR of DSK with (fetch CBQUEUE of DSK)))
        (for PAGENO from CURRENTPAGE to LASTPAGE do (COND
                                                       ((NULL (SETQ CB (\GETDISKCB DSK)))
                                                        (GO FAILURE)))
                                                    (COND
                                                       ((OR (AND (EQ PAGENO LASTPAGE)
                                                                 (NEQ (fetch DISKWRITELASTNUMCHARS of REQUEST)
                                                                      BYTESPERPAGE))
                                                            (EQ (\GETBASE DAS (ADD1 PAGENO))
                                                                \FILLINDA))
                                                                ; Mark end of file after this page
                                                        (\PUTBASE DAS (ADD1 PAGENO)
                                                                \EOFDA)))
                                                                ; Set up label with next and previous disk addresses, numchars
                                                    (SETQ LAB (LOCF (fetch (CB CBLABNEXT) of CB)))
                                                    [replace DLNEXT of LAB with (\REALDISKDA DSK
                                                                                     (\GETBASE DAS (ADD1 PAGENO]
                                                    [replace DLPREVIOUS of LAB with (\REALDISKDA
                                                                                      DSK
                                                                                      (\GETBASE DAS (SUB1 PAGENO]
                                                    (replace DLNUMCHARS of LAB with (COND
                                                                                       ((EQ PAGENO LASTPAGE)
                                                                                        (fetch
                                                                                             DISKWRITELASTNUMCHARS
                                                                                           of REQUEST))
                                                                                       (T BYTESPERPAGE)))
                                                    (replace (CB CBLABELADDR) of CB with (\LOLOC LAB))
                                                    (\DODISKCOMMAND DSK CB (OR BUFFER (EMPOINTER (\GETBASE
                                                                                                     CAS PAGENO))
                                                                                  )
                                                          (\GETBASE DAS PAGENO)
                                                          PAGENO \DC.WRITELD)
                                                                ; Wait for commands to complete
           finally (do (SELECTQ (\CLEANUPDISKQUEUE DSK)
                               (NIL (GO FAILURE))
                               (T (RETURN))
                               NIL)))
        (replace LASTPAGEACTEDON of REQUEST with LASTPAGE)
        (RETURN LASTPAGE)
     FAILURE
        (COND
           ((ILEQ (fetch DISKERRORCNT of DSK)
                  (fetch RETRYCOUNT of DSK))
            (SETQ CURRENTPAGE (fetch CURRENTDISKPAGE of DSK))
            (GO RETRY)))
        (RETURN (IMINUS \DSK.HARD.ERROR))
     DISKFULL
        (RETURN (IMINUS \DSK.FULL.ERROR])
```

(**CHECKFREEPAGE**

```
  [LAMBDA (DSK CB)                                              (* bvm%: " 9-DEC-82 13:41")
    ;; Check that CB got a free page, i.e.  one whose file id is all -1
    (PROG [(FID (LOCF (fetch DLFID of (EMPOINTER (fetch (CB CBLABELADDR) of CB]
          [FRPTQ 3 (PROGN (COND
                             ((NEQ (\GETBASE FID 0)
                                   (UNSIGNED -1 BITSPERWORD))    ; Oops, bittable was wrong, so nullify this guy's address in caller
                              (\PUTBASE (fetch (DSKOBJ DISKDAS) of DSK)
                                     (fetch (CB CBPAGENO) of CB)
                                     \FILLINDA)
                              (RETURN)))
                          (SETQ FID (\ADDBASE FID 1]
          (RETURN T])
```

(**DODISKCOMMAND**

```
  [LAMBDA (DSK CB BUFFER VDA PAGENO ACTION NEXTCB)              (* bvm%: "13-Feb-85 19:42")
    (PROG ((SHORTCB (fetch SHORTCB of CB))
           LA NEXT LASTCB STATUS)
          [replace CBHEADERADDR of SHORTCB with (\LOLOC (LOCF (fetch CBHEADER of SHORTCB]
          (replace CBDATAADDR of SHORTCB with (\LOLOC BUFFER))
          [COND
             ((EQ (SETQ LA (fetch CBLABELADDR of SHORTCB))
                  0)                                            ; Fill this in only if caller hasn't
              (replace CBLABELADDR of SHORTCB with (SETQ LA (\LOLOC (COND
                                                                       (NEXTCB (LOCF (fetch (CB CBDA) of NEXTCB)))
                                                                       (T (LOCF (fetch (CB CBLABNEXT) of CB]
          (SETQ LA (EMPOINTER LA))
          (\BLT (LOCF (fetch DLFID of LA))
                (fetch (DSKOBJ DISKFID) of DSK)
                3)                                              ; Set serial number for label check
          (replace DLPAGENO of LA with PAGENO)
          (replace CBPAGENO of CB with PAGENO)
          [COND
             ((NEQ VDA \FILLINDA)
              (replace CBDA of SHORTCB with (\REALDISKDA DSK VDA]
```

```
          (replace CBCOMMAND of SHORTCB with (IPLUS (SELECTC ACTION
                                                           (\DC.READHLD \IDC.READHLD)
                                                           (\DC.READLD \IDC.READLD)
                                                           (\DC.READD \IDC.READD)
                                                           (\DC.WRITEHLD \IDC.WRITEHLD)
                                                           (\DC.WRITELD \IDC.WRITELD)
                                                           (\DC.WRITED \IDC.WRITED)
                                                           (\DC.SEEKONLY \IDC.SEEKONLY)
                                                           (\DC.RESTORE (replace CBDA of SHORTCB
                                                                           with (ADD1 (LOGAND (fetch CBDA of SHORTCB)
                                                                                              61440)))
                                                                   ; Track _ 0, Restore _ 1, so command is seek to track zero
                                                                      \IDC.SEEKONLY)
                                                           (RAID "Invalid disk action" ACTION))
                                                    (LLSH (fetch DSKPARTITION of DSK)
                                                          8)))
          (SETQ LASTCB (EMGETBASE \EM.DISKCOMMAND))
          [COND
             ((NEQ LASTCB 0)                                        ; Disk is busy, queue CB up at end
              (while (NEQ (SETQ NEXT (fetch CBLINK of (EMPOINTER LASTCB)))
                          0)
                 do (SETQ LASTCB NEXT))
              (replace CBLINK of (EMPOINTER LASTCB) with (\LOLOC SHORTCB]
          [COND
             ((AND (EQ (EMGETBASE \EM.DISKCOMMAND)
                       0)
                   (OR (EQ LASTCB 0)
                       (EQ (fetch CBDONE of SHORTCB)
                           0)))
              ;; No CB's queued, so ours is the only one, and it hasn't been done yet.  Careful here! If the last disk command got an error, we don't
              ;; want to do this.  Also true if last disk command was never executed, which means that some earlier command got an error
              (COND
                 ((OR (NOT (SETQ LASTCB (fetch CBLASTPTR of DSK)))
                      (AND (NEQ (SETQ STATUS (LOGAND (fetch (CB CBSTATUS) of LASTCB)
                                                     \CBS.GOODMASK))
                                0)
                           (EQ (LOGAND STATUS \CBS.ERRORBITS)
                               0)))
                  (EMPUTBASE \EM.DISKCOMMAND (\LOLOC SHORTCB]
          (replace CBQSTATUS of CB with \CB.PENDING)
          (replace CBLASTPTR of DSK with CB)
          (\BOXIPLUS (LOCF (fetch DISKOPS of \MISCSTATS))
                 1])
```

(**GETDISKCB**

```
[LAMBDA (DSK)                                                       (* bvm%: "24-NOV-82 18:12")
```

;; Gets a new CB, clearing it out, or returns NIL if there are errors. In latter case, caller should retry starting with CURRENTPAGE (set freely by ;; \CLEANUPDISKQUEUE)

```
(PROG (CB)
  LP  (RETURN (COND
                 ((SETQ CB (fetch CBFREEPTR of DSK))
                  [replace CBFREEPTR of DSK with (COND
                                                    ((EQ (fetch CBNEXT of CB)
                                                         (fetch CBPENDINGPTR of DSK))
                                                           ; Circular buffer;  when pointers are equal means everyone is
                                                           ; free.  Free = NIL means nobody is free
                                                     NIL)
                                                    (T (fetch CBNEXT of CB]
                  (\CLEARCB CB)
                  CB)
                 ((NOT (\CLEANUPDISKQUEUE DSK))                     ; an error occurred
                  NIL)
                 (T                                                 ; A CB was returned to the free queue
                    (GO LP])
```

(**CLEARCB**

```
[LAMBDA (CB)                                                        (* bvm%: "13-Feb-85 19:19")
  (\CLEARWORDS (fetch SHORTCB of CB)
         \LENSHORTCB)
  (replace CBQSTATUS of CB with \CB.FREE)
  CB])
```

(**CLEANUPDISKQUEUE**

```
[LAMBDA (DSK)                                                       (* bvm%: "13-Feb-85 19:42")
```

;; Called to process pending CB's. If queue is empty and all is quiet, returns T. Returns NIL on errors, after running error routine. Otherwise, ;; returns a finished CB, which has also been placed on \FREEDISKCBS by this action

```
(PROG ((CB (fetch CBPENDINGPTR of DSK))
       (FREE (fetch CBFREEPTR of DSK))
       SHORTCB LABEL LVDA)
      (COND
         ((EQ CB FREE)
```

```
            (RETURN T)))
      LP                                                    ; Wait for disk to finish something
          (SETQ SHORTCB (fetch SHORTCB of CB))
          (COND
             ((EQ (fetch CBDONE of SHORTCB)
                  0)                                        ; Command not done yet
              [COND
                 ((AND (EQ (EMGETBASE \EM.DISKCOMMAND)
                           0)
                       (EQ (fetch CBDONE of SHORTCB)
                           0))                              ; Disk queue was flushed for some reason.  Fake an error
                  (COND
                     ((NEQ (fetch CBQSTATUS of CB)
                           \CB.PENDING)
                      (RETURN (RAID "No free CB's")))
                     (T (replace CBSTATUS of SHORTCB with \CBS.FAKEERROR]
                                                            ; Here is where some day we could block to let another process
                                                            ; run
              (GO LP)))                                     ; We now have CB free from the disk controller
          (replace CBPENDINGPTR of DSK with (fetch CBNEXT of CB))
          (COND
             ((EQ CB (fetch CBLASTPTR of DSK))
              (replace CBLASTPTR of DSK with NIL)))         ; Remove from pending queue
          (replace CBSHORTSEAL of SHORTCB with 0)           ; Invalidate it as a disk command, just in case
          [COND
             ((NOT FREE)
              (replace CBFREEPTR of DSK with (SETQ FREE CB]         ; Now clean up the transfer
          (COND
             ((EQ (fetch RESTORE of (fetch CBDA of SHORTCB))
                  1)                                        ; This is our command, not user's, so nothing to cleanup
              (RETURN CB)))
          [COND
             ((NEQ (LOGAND (fetch CBSTATUS of SHORTCB)
                           \CBS.GOODMASK)
                   \CBS.GOOD)                               ; Error occurred
              (repeatuntil (EQ (EMGETBASE \EM.DISKCOMMAND)
                               0))                          ; Wait for disk to stop spinning
              (COND
                 ((NEQ (fetch TOTALDISKERRORS of DSK)
                       MAX.SMALL.INTEGER)                   ; Keep this count for debugging
                  (add (fetch TOTALDISKERRORS of DSK)
                       1)))
              (RETURN (COND
                         [(IGREATERP (add (fetch DISKERRORCNT of DSK)
                                          1)
                                     (fetch RETRYCOUNT of DSK))         ; Hard error
                          (COND
                             (\DISKDEBUG                    ; Error is normally fielded in a more benign place
                                    (RAID "Hard Disk Error.  ^N to continue" CB]
                         (T (COND
                               ((EQ (fetch CBFINALSTATUS of SHORTCB)
                                    \CBS.CHECKERROR)
                                (replace SAWCHECKERROR of DSK with T)))
                            (replace CURRENTDISKPAGE of DSK with (fetch CBPAGENO of CB))
                            (COND
                               ((IGREATERP (fetch DISKERRORCNT of DSK)
                                           (LRSH (fetch RETRYCOUNT of DSK)
                                                 1))        ; Half the tolerable errors.  Initiate a Restore to let disk
                                                            ; recalibrate
                                (EMPUTBASE \EM.DISKADDRESS (UNSIGNED -1 BITSPERWORD))
                                                            ; This forces a seek
                                (\DODISKCOMMAND DSK (\GETDISKCB DSK)
                                       NIL
                                       (\VIRTUALDISKDA DSK (fetch CBDA of SHORTCB))
                                       (fetch CURRENTDISKPAGE of DSK)
                                       \DC.RESTORE)))
                            NIL]
          (SETQ LABEL (EMPOINTER (fetch CBLABELADDR of SHORTCB)))
          (replace (DSKOBJ CURRENTNUMCHARS) of DSK with (fetch DLNUMCHARS of LABEL))
          (replace DISKERRORCNT of DSK with 0)
          (replace SAWCHECKERROR of DSK with NIL)
          [COND
             ((fetch (DSKOBJ CBCLEANUPFN) of DSK)
              (APPLY* (fetch (DSKOBJ CBCLEANUPFN) of DSK)
                      DSK CB))
             (T [SETQ LVDA (\ADDBASE (fetch CURRENTDAS of DSK)
                                     (SUB1 (fetch CBPAGENO of CB]
                [COND
                   ((EQ (\GETBASE LVDA 2)
                        \FILLINDA)                          ; Fill in Next address
                    (\PUTBASE LVDA 2 (\VIRTUALDISKDA DSK (fetch DLNEXT of LABEL]
                (COND
                   ((EQ (\GETBASE LVDA 0)
                        \FILLINDA)                          ; Fill in Previous address
                    (\PUTBASE LVDA 0 (\VIRTUALDISKDA DSK (fetch DLPREVIOUS of LABEL]
          (RETURN CB])
```

```
(\VIRTUALDISKDA
  [LAMBDA (DSK REALDA)                                    ; Edited 13-Jun-2021 11:51 by rmk:
                                                          ; Converts a real disk address into a virtual one
    (COND
       ((EQ REALDA 0)
        \EOFDA)
       (T (IPLUS (ITIMES (IPLUS (ITIMES (IPLUS (ITIMES (fetch (REALDA DISK) of REALDA)
                                                       (fetch (DSKOBJ NTRACKS) of DSK))
                                              (fetch (REALDA TRACK) of REALDA))
                                     (fetch (DSKOBJ NHEADS) of DSK))
                            (fetch (REALDA HEAD) of REALDA))
                   (fetch (DSKOBJ NSECTORS) of DSK))
          (fetch (REALDA SECTOR) of REALDA])

(\REALDISKDA
  [LAMBDA (DSK VDA)                                       (* bvm%: "18-NOV-82 21:16")
    ;; Returns a real disk address for given virtual address
    (COND
       ((EQ VDA \EOFDA)
        0)
       (T (PROG ((NSECTORS (fetch NSECTORS of DSK))
                 (NHEADS (fetch NHEADS of DSK))
                 (NTRACKS (fetch NTRACKS of DSK)))
                (RETURN (IPLUS (LLSH (IREMAINDER VDA NSECTORS)
                                     12)
                               (LLSH (IREMAINDER (SETQ VDA (IQUOTIENT VDA NSECTORS))
                                                 NHEADS)
                                     2)
                               (LLSH (IREMAINDER (SETQ VDA (IQUOTIENT VDA NHEADS))
                                                 NTRACKS)
                                     3)
                               (LLSH (IQUOTIENT VDA NTRACKS)
                                     1])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(RPAQQ CBSTATUSCONSTANTS ((\CBS.ERRORBITS 183)
                          (\CBS.GOODMASK 4023)
                          (\CBS.GOOD 3840)
                          (\CBS.FAKEERROR 3841)
                          (\CBS.CHECKERROR 2)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \CBS.ERRORBITS 183)

(RPAQQ \CBS.GOODMASK 4023)

(RPAQQ \CBS.GOOD 3840)

(RPAQQ \CBS.FAKEERROR 3841)

(RPAQQ \CBS.CHECKERROR 2)

(CONSTANTS (\CBS.ERRORBITS 183)
       (\CBS.GOODMASK 4023)
       (\CBS.GOOD 3840)
       (\CBS.FAKEERROR 3841)
       (\CBS.CHECKERROR 2))
)

(RPAQQ IDISKCOMMANDS ((\IDC.READHLD 18432)
                      (\IDC.READLD 18496)
                      (\IDC.READD 18512)
                      (\IDC.WRITEHLD 18600)
                      (\IDC.WRITELD 18536)
                      (\IDC.WRITED 18520)
                      (\IDC.SEEKONLY 18434)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \IDC.READHLD 18432)

(RPAQQ \IDC.READLD 18496)

(RPAQQ \IDC.READD 18512)

(RPAQQ \IDC.WRITEHLD 18600)

(RPAQQ \IDC.WRITELD 18536)

(RPAQQ \IDC.WRITED 18520)
```

```
(RPAQQ \IDC.SEEKONLY 18434)

(CONSTANTS (\IDC.READHLD 18432)
       (\IDC.READLD 18496)
       (\IDC.READD 18512)
       (\IDC.WRITEHLD 18600)
       (\IDC.WRITELD 18536)
       (\IDC.WRITED 18520)
       (\IDC.SEEKONLY 18434))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \EM.DISKCOMMAND 337)

(RPAQQ \EM.DISKADDRESS 339)

(RPAQQ \FIXEDLENDISKREQUEST 42)

(RPAQQ \DEFAULTDASTORAGELENGTH 60)

(RPAQQ \LENCB 6)

(RPAQQ \LENDSKOBJ 34)

(RPAQQ \LENSHORTCB 18)

(CONSTANTS (\EM.DISKCOMMAND 337)
       (\EM.DISKADDRESS 339)
       (\FIXEDLENDISKREQUEST 42)
       (\DEFAULTDASTORAGELENGTH 60)
       (\LENCB 6)
       (\LENDSKOBJ 34)
       (\LENSHORTCB 18))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \CB.PENDING 1)

(RPAQQ \CB.FREE 0)

(CONSTANTS (\CB.PENDING 1)
       (\CB.FREE 0))
)
)

;; At MAKEINIT time

(DEFINEQ

(MAKEINITBFS
  [LAMBDA NIL                                                    ; Edited  5-Nov-87 16:54 by bvm:
    ;; Called at MAKEINIT time to create bfs structures
    (PROGN                                                       ; Create disk structures for up to 3 virtual memory partitions.
           (\LOCKWORDS (SETQ \MAINDISK (create DSKOBJ))
                  \NWORDS.DSKOBJ)
           (replace DISKDEVICENAME of \MAINDISK with (EVQ 'DSK))
           (\LOCKWORDS (SETQ \SWAPDSK1 (create DSKOBJ))
                  \NWORDS.DSKOBJ)
           (\LOCKWORDS (SETQ \SWAPDSK2 (create DSKOBJ))
                  \NWORDS.DSKOBJ))
    (PROGN                                                       ; Disk request blocks for use by swapper and file system
           (\LOCKWORDS (SETQ \SWAPREQUESTBLOCK (create DISKREQUEST))
                  (+ \FIXEDLENDISKREQUEST \DEFAULTDASTORAGELENGTH))
           (\LOCKWORDS (SETQ \DISKREQUESTBLOCK (create DISKREQUEST))
                  (+ \FIXEDLENDISKREQUEST \DEFAULTDASTORAGELENGTH)))
    (to 3 bind PREV (FIRSTCB _ (create CB)) first (\LOCKWORDS (SETQ PREV FIRSTCB)
                                                        \LENCB)
       do (\LOCKWORDS (SETQ PREV (create CB
                                         CBNEXT _ PREV))
                 \LENCB)
       finally (replace CBNEXT of FIRSTCB with PREV)             ; note queue is now circular
              (replace CBQUEUE of \MAINDISK with FIRSTCB))
    (SETQ \FREEPAGEFID (\ALLOCBLOCK 3))                          ; FP or FID for free disk page is all -1
    (for I from 0 to 4 do (\PUTBASE \FREEPAGEFID I (UNSIGNED -1 BITSPERWORD])

)

(DECLARE%: DONTCOPY

(ADDTOVAR INITPTRS (\MAINDISK)
                   (\SWAPDSK1)
                   (\SWAPDSK2)
                   (\SWAPREQUESTBLOCK)
```

```
                (\DISKREQUESTBLOCK)
                (\FREEPAGEFID))

(ADDTOVAR INEWCOMS (FNS MAKEINITBFS))


(ADDTOVAR DONTCOMPILEFNS MAKEINITBFS)
)

;; Swap stuff

(DEFINEQ

(\M44ACTONVMEMFILE
  [LAMBDA (FIRSTPAGE BUFFER NPAGES WRITEFLG)                    ; Edited  5-Nov-87 17:21 by bvm:
    (PROG ((LASTPAGE (IPLUS FIRSTPAGE NPAGES -1))
           (DAs \ISFSCRATCHDAS)
           (CAs \ISFSCRATCHCAS)
           (PAGE FIRSTPAGE)
           (BUF BUFFER)
           CHUNK RESULT)
          [if \XVmem
              then ;; extended virtual memory is in use
                   (while (IGREATERP NPAGES 0) bind DISKRELATIVEPAGE WHICHPART ISFMAP
                      do [SETQ ISFMAP (EMPOINTER (\GETBASE \XVmemFmapBase (SETQ WHICHPART (\WHICHPART PAGE]
                         (if (EQ NPAGES 1)
                             then                                ; common case
                                  (SETQ CHUNK 1)
                           else (SETQ CHUNK (IMIN NPAGES \ISFCHUNKSIZE))
                                (if (IGREATERP (IPLUS PAGE (SUB1 CHUNK))
                                               (\GETBASE \XVmemFmapHighBase WHICHPART))
                                    then                         ; oops -- trying to cross partition boundary in this chunk. Take the
                                                                 ; easy way out and step by ones until past the boundary.
                                         (SETQ CHUNK 1)))
                         [SETQ DISKRELATIVEPAGE (if (EQ WHICHPART 0)
                                                    then PAGE
                                                  else ;; need to adjust page number based on which partition.
                                                       (ADD1 (IDIFFERENCE PAGE (\GETBASE \XVmemFmapHighBase
                                                                                         (SUB1 WHICHPART]
                         (for I from 0 to (SUB1 CHUNK) do (\PUTBASE CAs I (\LOLOC BUF))
                                                          (\PUTBASE DAs I (\LOOKUPFMAP (IPLUS DISKRELATIVEPAGE I)
                                                                                       ISFMAP))
                                                          (SETQ BUF (\ADDBASE BUF WORDSPERPAGE)))
                         (\PUTBASE DAs CHUNK \FILLINDA)
                         (\BLT (LOCF (fetch DISKSERIAL# of \SWAPREQUESTBLOCK))
                               ISFMAP WORDSPERCELL)              ; Fill in disk label for this partition's vmem
                         [COND
                            ((ILESSP (SETQ RESULT (\ACTONVMEMPAGES (\GETBASEPTR \XVmemDiskBase
                                                                                (LLSH WHICHPART 1))
                                                                   NIL DAs DISKRELATIVEPAGE DISKRELATIVEPAGE
                                                                   (IPLUS DISKRELATIVEPAGE CHUNK -1)
                                                                   (COND
                                                                      (WRITEFLG \DC.WRITED)
                                                                      (T \DC.READD))
                                                                   NIL NIL CAs))
                                     0)
                             (\SWAPDISKERROR (IMINUS RESULT)
                                             (\GETBASEPTR \XVmemDiskBase (LLSH WHICHPART 1]
                         (SETQ NPAGES (IDIFFERENCE NPAGES CHUNK))
                         (SETQ PAGE (IPLUS PAGE CHUNK)))
            else ;; not using extended virtual memory
                 (AND (IGEQ LASTPAGE (\GETBASE \ISFMAP (fetch ISFLAST of \ISFMAP)))
                      (EQ (\LOOKUPFMAP FIRSTPAGE)
                          \FILLINDA)
                      (RETURN (RAID "Can't complete swap operation--page not in isf map")))
                 (while (IGREATERP NPAGES 0) do (SETQ CHUNK (IMIN NPAGES \ISFCHUNKSIZE))
                                                (for I from 0 to (SUB1 CHUNK)
                                                   do (\PUTBASE CAs I (\LOLOC BUF))
                                                      (\PUTBASE DAs I (\LOOKUPFMAP (IPLUS PAGE I)))
                                                      (SETQ BUF (\ADDBASE BUF WORDSPERPAGE)))
                                                (\PUTBASE DAs CHUNK \FILLINDA)
                                                [COND
                                                   ((ILESSP (SETQ RESULT (\ACTONVMEMPAGES \MAINDISK NIL DAs
                                                                                          PAGE PAGE (IPLUS PAGE CHUNK -1)
                                                                                          (COND
                                                                                             (WRITEFLG \DC.WRITED)
                                                                                             (T \DC.READD))
                                                                                          NIL NIL CAs))
                                                            0)
                                                    (\SWAPDISKERROR (IMINUS RESULT]
                                                (SETQ NPAGES (IDIFFERENCE NPAGES CHUNK))
                                                (SETQ PAGE (IPLUS PAGE CHUNK]
          (RETURN LASTPAGE])
```

(\LOOKUPFMAP

```
[LAMBDA (PAGE ISFMAP)                                   ; Edited  5-Nov-87 16:30 by bvm:
    ;; Return virtual DA for partition-relative page number PAGE by searching the run table in ISFMAP, which defaults to the global \isfmap in the
    ;; non-xvmem case.  With extended vmem, caller already figured out which partition the absolute page number belongs to.
    (PROG ((LO \ISFMAPOFFSET)
           HI MID)
          (if (NOT ISFMAP)
              then ;; non-xvmem case, worry about perversities.
                   (if (EQ PAGE (fetch ISFONEPAGE of \ISFMAP))
                       then                             ; This is in case runtable overflows
                            (RETURN (fetch ISFONEDA of \ISFMAP)))
                   (if (IGEQ PAGE (\GETBASE \ISFMAP (fetch ISFLAST of \ISFMAP)))
                       then                             ; Should never happen
                            (RETURN \FILLINDA))
                   (SETQ ISFMAP \ISFMAP))
          (SETQ HI (fetch ISFLAST of ISFMAP))
          (while (IGREATERP HI (IPLUS LO 2)) do (SETQ MID (FLOOR (FOLDLO (IPLUS LO HI)
                                                                         2)
                                                                 2))
                                                ; Do binary chop on map.  Page numbers are all at even offsets
                                                (if (IGEQ PAGE (\GETBASE ISFMAP MID))
                                                    then (SETQ LO MID)
                                                  else (SETQ HI MID)))
          (SETQ LO (\ADDBASE ISFMAP LO))
          (RETURN (IPLUS (\GETBASE LO 1)
                         (IDIFFERENCE PAGE (\GETBASE LO 0])
```

(\M44EXTENDVMEMFILE

```
[LAMBDA (LASTPAGE)                                      ; Edited  5-Nov-87 16:58 by bvm:
```

;;; Extends vmem to at least LASTPAGE in length, returns T unless it wants to defer doing this til later.

```
    ;; Extending the vmem seems like a bad thing to do these days, since people usually make the vmem the size they want in the first place.  Also,
    ;; with multiple-partition vmems, we never want to extend.  And finally, there is evidence that there is some buggy code in here somewhere.

    ;; If we ever revive this code, go back to Lyric sources.

    T])
```

(\M44DOEXTENDVMEMFILE

```
[LAMBDA (LASTPAGE)                                      ; Edited  5-Nov-87 17:11 by bvm:
    ;; Called when LASTPAGE is not in the ISF map.  Might simply need to look up some new pages;  if we have already scanned the whole file,
    ;; however, we will need to extend the file itself.  Returns error code on failure.  If LASTPAGE is NIL just reads to the end of the vmem file

    ;; This function is currently only used to find the end of the vmem, and even that function can be tossed if we require Bcpl version 26400.  If you
    ;; want to revive old functionality, look back to the Lyric sources.

    (AND LASTPAGE (SHOULDNT))
    (PROG ((SCRATCHBUF \SPAREDISKWRITEBUFFER)
           (DAs \ISFSCRATCHDAS)
           (LASTNEEDEDPAGE LASTPAGE)
           [LASTFULLPAGE (SUB1 (\GETBASE \ISFMAP (fetch ISFLAST of \ISFMAP]
           FIRSTDA NP LASTPAGEREAD LASTPAGEADDR LASTPAGEOFFSET EXTENDED)
     ;; Use \SPAREDISKWRITEBUFFER as a scratch buf, assuming this is not called while \DOWRITEDISKPAGES is happening (because
     ;; \DOWRITEDISKPAGES is locked).  If it turns out we actually need to invoke \DOWRITEDISKPAGES ourselves, then we better not have been
     ;; writing the disk anyway, in which case we steal a different disk buffer
          (while T do (SETQ NP (IDIFFERENCE \ISFCHUNKSIZE 2))
                      [\PUTBASE DAs 0 (COND
                                         ((EQ LASTFULLPAGE 0)
                                          \EOFDA)
                                         (T (\LOOKUPFMAP (SUB1 LASTFULLPAGE]
                      (\PUTBASE DAs 1 (\LOOKUPFMAP LASTFULLPAGE))
                                                ; Will operate on LASTFULLPAGE plus as many remaining
                                                ; pages as desired.  NP is number of new pages
                      (for I from 2 to (IPLUS NP 2) do (\PUTBASE DAs I \FILLINDA))
                                                ; \FILLINDA for NP starting at first unknown page.  Last one is
                                                ; bonus
                      [COND
                         ((ILESSP (SETQ LASTPAGEREAD (\ACTONVMEMPAGES \MAINDISK SCRATCHBUF DAs (SUB1
                                                                                                  LASTFULLPAGE
                                                                                                    )
                                                                                LASTFULLPAGE
                                                                                (IPLUS LASTFULLPAGE NP)
                                                                                \DC.READD NIL (fetch ISFHINTLASTPAGE of \ISFMAP)))
                                  0)
                          (\DISKERROR (IMINUS LASTPAGEREAD]
                      (SETQ LASTPAGEOFFSET (IPLUS (IDIFFERENCE LASTPAGEREAD LASTFULLPAGE)
                                                  2))   ; Offset in DAs of one past LASTPAGEREAD
                      [COND
                         ((OR (NEQ (fetch CURRENTNUMCHARS of \SWAPREQUESTBLOCK)
                                   BYTESPERPAGE)
                              (EQ (\GETBASE DAs LASTPAGEOFFSET)
                                  \EOFDA))
```

```
                        ;; Read to EOF.  The second condition should never be needed, but if file is malformed at the end we would be in
                        ;; trouble
                        (COND
                           ((ILEQ LASTPAGEOFFSET 3)                    ; No pages were acted on, so extend simply
                            (RETURN NIL))
                           (T ;; Too hard to do both.  Fill in part of map now, and extend the file on the next iteration.  Number of good
                              ;; pages read was LASTPAGEREAD-1-LASTFULLPAGE
                              (SETQ NP (IDIFFERENCE LASTPAGEOFFSET 3]
                     [for I from 1 to NP do (\EXTENDISFMAP (IPLUS LASTFULLPAGE I)
                                                           (\GETBASE DAs (ADD1 I]
                     (SETQ LASTFULLPAGE (IPLUS LASTFULLPAGE NP))
                     (SETQ EXTENDED T))
               [COND
                  ((AND EXTENDED (NEQ (fetch ISFREWRITE of \ISFMAP)
                                      0))                              ; Write map back onto file
                   (PROG ((CAs \ISFSCRATCHCAS))
                         (\PUTBASE DAs 0 \EOFDA)
                         (\PUTBASE DAs 1 (fetch ISFDA0 of \ISFMAP))
                         (\PUTBASE DAs 2 (fetch ISFDA1 of \ISFMAP))
                         (\PUTBASE DAs 3 (fetch ISFDA2 of \ISFMAP))
                         (\PUTBASE CAs 1 (\LOLOC SCRATCHBUF))
                         (\PUTBASE CAs 2 (\LOLOC \ISFMAP))             ; Set up to write first data page of file
                         (\ACTONVMEMPAGES \MAINDISK NIL DAs -1 1 1 \DC.WRITED NIL NIL CAs]
               [SETQ \LASTVMEMFILEPAGE (SUB1 (\GETBASE \ISFMAP (fetch ISFLAST of \ISFMAP]
                                                                       ; Update pointer to true end, return NIL to signal success
               (RETURN NIL])
```

(\EXTENDISFMAP

```
  [LAMBDA (PAGE DA)                                                    (* bvm%: "14-Feb-85 23:29")
    ;; extend map to include the knowledge that DA is address of PAGE
    (PROG ((LASTOFFSET (fetch ISFLAST of \ISFMAP))
           LASTPAGE LASTMAP)
          (replace ISFONEPAGE of \ISFMAP with PAGE)
          (replace ISFONEDA of \ISFMAP with DA)
          (SETQ LASTMAP (\ADDBASE \ISFMAP (IDIFFERENCE LASTOFFSET 2)))
                                                                       ; LASTMAP points at the last Page, DA pair in map
          (COND
             ((NEQ (SETQ LASTPAGE (\GETBASE LASTMAP 2))
                   PAGE)
              (RETURN)))
          [COND
             ([EQ DA (IPLUS (\GETBASE LASTMAP 1)
                            (IDIFFERENCE LASTPAGE (\GETBASE LASTMAP 0]
                                                                       ; Still in same chunk
              (\PUTBASE LASTMAP 2 (ADD1 LASTPAGE)))
             (T                                                        ; Start new chunk
                (COND
                   ((EQ LASTOFFSET (fetch ISFEND of \ISFMAP))          ; No more space in map
                    (RETURN))
                   (T (\PUTBASE LASTMAP 3 DA)                          ; DA corresponding to LASTPAGE=PAGE
                      (\PUTBASE LASTMAP 4 (ADD1 LASTPAGE))
                      (replace ISFLAST of \ISFMAP with (IPLUS LASTOFFSET 2]
          (RETURN T])

)

;; Extended vmem stuff

(DEFINEQ

(\EXTENDEDVMEMINIT
  [LAMBDA NIL                                                          ; Edited  5-Nov-87 16:50 by bvm:
    ;; Initialize structures used by extended Dorado virtual memory.  \XVmem is set to NIL if not present.
    (if (fetch (IFPAGE XVmemFmapBase) of \InterfacePage)
        then ;; check to see if Lisp.run had initialized the pointers to the ISFmaps, a good hint about whether we have the correct version of Lisp.run.
             ;; convenient references to items in the interface page
             (SETQ \XVmemFmapBase (LOCF (fetch (IFPAGE XVmemFmapBase) of \InterfacePage)))
             (SETQ \XVmemFmapHighBase (LOCF (fetch (IFPAGE XVmemFmapHighBase) of \InterfacePage)))
             (SETQ \XVmemDiskBase (LOCF (fetch (IFPAGE XVmemDiskBase) of \InterfacePage)))
             ;; record the last file page where the system expects to find it
             (SETQ \LASTVMEMFILEPAGE (IMAX (\GETBASE \XVmemFmapHighBase 0)
                                           (\GETBASE \XVmemFmapHighBase 1)
                                           (\GETBASE \XVmemFmapHighBase 2)))
             ;; fill in the Lisp disk objects for the extra swap partitions.  These must have been previously created and locked down.
             (\INITIALIZESWAPDISK \SWAPDSK1 1)
             (\INITIALIZESWAPDISK \SWAPDSK2 2)
             ;; store pointers to the disk devices in the disk array in the Interface page
```

```
            (\PUTBASEPTR \XVmemDiskBase 0 \MAINDISK)
            (\PUTBASEPTR \XVmemDiskBase 2 \SWAPDSK1)
            (\PUTBASEPTR \XVmemDiskBase 4 \SWAPDSK2)
            ;; check to see whether it is worth turning on the extended vmem -- there must be some extra files present.
            (SETQ \XVmem (NEQ (\GETBASE \XVmemFmapHighBase 1)
                              0))
      else (PROGN                                          ; Using old Lisp.run, have to figure out how big vmem file is
                  (\M44DOEXTENDVMEMFILE))
           (SETQ \XVmem NIL])
```

(\INITIALIZESWAPDISK

```
  [LAMBDA (DISK ISFindex)                                  ; Edited  5-Nov-87 15:48 by bvm:
    ;; called by \EXTENDEDVMEMINIT to initialize the various disk objects it uses.
    (\BLT DISK \MAINDISK \NWORDS.DSKOBJ)                   ; start with the main disk, but be careful--some of these fields are
                                                           ; ref-counted
    (\PUTBASEPTR (LOCF (fetch SYSDIROFD of DISK))
           0 NIL)
    (\PUTBASEPTR (LOCF (FETCH DISKDESCRIPTOROFD OF DISK))
           0 NIL)
    [replace ALTODSKOBJ of DISK with (EMPOINTER (fetch (ISFMAP ISFDISK) of (EMPOINTER (\GETBASE \XVmemFmapBase
                                                                                        ISFindex]
    [replace ddPOINTER of DISK with (LOCF (fetch (ALTODSKOBJ DDLASTSERIAL#) of (fetch ALTODSKOBJ of DISK]
    (replace DSKPARTITION of DISK with (LRSH (fetch DSKDRIVE# of (fetch ALTODSKOBJ of DISK))
                                             1))
    (replace DISKDEVICENAME of DISK with NIL])
```

(\WHICHPART

```
  [LAMBDA (PAGE)                                           ; Edited 16-Feb-87 10:28 by Briggs
    (if (ILEQ PAGE (\GETBASE \XVmemFmapHighBase 0))
        then 0
      elseif (ILEQ PAGE (\GETBASE \XVmemFmapHighBase 1))
        then 1
      elseif (ILEQ PAGE (\GETBASE \XVmemFmapHighBase 2))
        then 2
      else (RAID "Attempting to access a page not in ISFmap [extended virtual memory]" PAGE])
```

(\SWAPDISKERROR

```
  [LAMBDA (ERRCODE DISK)                                   ; Edited  5-Nov-87 15:50 by bvm:
    (while T do (SELECTC ERRCODE
                    (\DSK.HARD.ERROR
                        (RAID (if (EQ DISK \MAINDISK)
                                  then "Hard disk error in primary Lisp.VirtualMem"
                                elseif (EQ DISK \SWAPDSK1)
                                  then "Hard disk error in first Lisp.XVirtualMem"
                                else "Hard disk error in second Lisp.XVirtualMem")
                              (fetch (DSKOBJ CURRENTDISKPAGE) of DISK)))
                    (RAID (if (EQ DISK \MAINDISK)
                              then "Unknown disk error in primary Lisp.VirtualMem"
                            elseif (EQ DISK \SWAPDSK1)
                              then "Unknown disk error in first Lisp.XVirtualMem"
                              else "Unknown disk error in second Lisp.XVirtualMem")
                          ERRCODE])

)

;; For debugging and user info

(DEFINEQ

(DESCRIBE-VIRTUAL-MEMORY
  [LAMBDA NIL                                              ; Edited  6-Nov-87 16:45 by bvm:
    [if (NEQ \MACHINETYPE \DORADO)
        then (printout NIL "This is not a Dorado!")
      else (LET ((NOBCPL (NEQ (fetch (IFPAGE XVmemFmapBase) of \InterfacePage)
                              (\LOLOC \ISFMAP)))
                 (VMSIZE (VMEMSIZE)))
                (CL:FORMAT T "The extended virtual memory Lisp.run has ~@[not ~]been installed.~%" NOBCPL)
                (CL:FORMAT T "There are currently ~D of ~D pages of virtual memory in use.~%" VMSIZE
                       \LASTVMEMFILEPAGE)
                (CL:UNLESS NOBCPL
                    (for I from 0 to 2 bind ISFMAP when (NEQ 0 (\GETBASE (LOCF (fetch (IFPAGE XVmemFmapHighBase)
                                                                                      of \InterfacePage))
                                                                              I))
                       do (SETQ ISFMAP (EMPOINTER (\GETBASE (LOCF (fetch (IFPAGE XVmemFmapBase) of \InterfacePage)
                                                                  )
                                                            I)))
                          [if (EQ I 0)
                              then (PRIN1 "The primary partition")
                            else (CL:FORMAT T "Secondary partition ~D" (fetch (DSKOBJ DSKPARTITION)
                                                                         of (\GETBASEPTR
                                                                             (LOCF (fetch (IFPAGE XVmemDiskBase)
                                                                                      of \InterfacePage))
```

```
                                                                    (UNFOLD I WORDSPERCELL]
                          (CL:FORMAT T " has ~D pages in ~D segment~:P.~%" (- (\GETBASE ISFMAP
                                                                                (fetch (ISFMAP ISFLAST)
                                                                                   of ISFMAP))
                                                                          (CL:IF (EQ I 0)
                                                                              2
                                                                              1))
                                                  (IQUOTIENT (- (fetch ISFLAST of ISFMAP)
                                                                \ISFMAPOFFSET)
                                                             2)))
                                        (if (NEQ 0 (\WHICHPART VMSIZE))
                                            then (printout T "A sysout cannot be made because secondary partitions are in use" T))))]
    (CL:VALUES])
```

(\PRINTFMAP

```
  [LAMBDA (ISFMAP)                                                    ; Edited  6-Nov-87 16:32 by bvm:
    [if (SMALLP ISFMAP)
        then (SETQ ISFMAP (EMPOINTER (\GETBASE \XVmemFmapBase ISFMAP]
    (CL:FORMAT T "ISFmap of ~D segments:~%" (IQUOTIENT (- (fetch ISFLAST of ISFMAP)
                                                          \ISFMAPOFFSET)
                                                       2))
    (for OFF from \ISFMAPOFFSET to (- (fetch ISFLAST of ISFMAP)
                                      2)
       by 2 bind START END VDA do (CL:FORMAT T "Pages [~D..~D] at vda [~O..~O]B~%" (SETQ START (\GETBASE ISFMAP
                                                                                                 OFF))
                                        [SETQ END (SUB1 (\GETBASE ISFMAP (+ OFF 2]
                                        (SETQ VDA (\GETBASE ISFMAP (+ OFF 1)))
                                        (+ VDA (- END START])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD ISFMAP ((NIL 5 WORD)                                     ; First 5 words are a FP
                     (ISFDA0 WORD)                                    ; DA's of the first 3 pages of file
                     (ISFDA1 WORD)
                     (ISFDA2 WORD)
                     (ISFSEAL WORD)
                     (ISFDISK WORD)                                   ; points to a DSKOBJ for file
                     (NIL WORD)                                       ; ZONE
                     (ISFLAST WORD)                                   ; offset of last entry in map
                     (ISFEND WORD)                                    ; Offset of end of space for map
                     (ISFONEPAGE WORD)                                ; Last page# added to map
                     (ISFONEDA WORD)                                  ; its DA
                     (ISFREWRITE WORD)                                ; non-zero if map should be rewritten when file is extended
                     (ISFCHUNKSIZE WORD)                              ; if file needs to be extended, do so in this size unit
                     (ISFHINTLASTPAGE WORD)                           ; Hint of last page
                     (ISFMAPSTART WORD)
                     ;; Map entries follow.  Each is two words: the page number of the start of a run, followed by the vda of that first page

                     ))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \ISFMAPOFFSET 18)

(CONSTANTS (\ISFMAPOFFSET 18))
)
)

(RPAQ? \DISKDEBUG )

(RPAQ? \MAXSWAPBUFFERS 1)

(RPAQ? \M44.READY )

(ADDTOVAR \SYSTEMCACHEVARS \M44.READY)

(DECLARE%: DONTCOPY

(ADDTOVAR INEWCOMS
          (ALLOCAL (ADDVARS (LOCKEDFNS ERROR RAID \M44ACTONVMEMFILE \ACTONVMEMFILESUBR \ACTONVMEMPAGES
                                       \CLEANUPDISKQUEUE \CLEARCB \DISKERROR \DOACTONDISKPAGES \DODISKCOMMAND
                                       \EXTENDISFMAP \M44DOEXTENDVMEMFILE \GETDISKCB \INITBFS \INSUREVMEMFILE
                                       \LISPERROR \LOOKUPFMAP \REALDISKDA \VIRTUALDISKDA \CLEARWORDS \TESTPARTITION
                                       \EXTENDEDVMEMINIT \WHICHPART \INITIALIZESWAPDISK \SWAPDISKERROR)
                            (LOCKEDVARS \DISKREQUESTBLOCK \SWAPREQUESTBLOCK \MAINDISK \SWAPDSK1 \SWAPDSK2
                                   \ISFCHUNKSIZE \EMUSCRATCH \EMUDISKBUFFERS \EMUSWAPBUFFERS \EMUDISKBUFEND
                                   \MAXSWAPBUFFERS \#DISKBUFFERS \InterfacePage \ISFMAP \ISFSCRATCHCAS
                                   \ISFSCRATCHDAS \SYSDISK \#SWAPBUFFERS \MAXDISKDAs \DISKDEBUG
                                   \SPAREDISKWRITEBUFFER \#EMUBUFFERS \EMUBUFFERS \LASTVMEMFILEPAGE \XVmem
                                   \XVmemFmapBase \XVmemFmapHighBase \XVmemDiskBase))))
)
```

(PUTPROPS **LLBFS COPYRIGHT** ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1990 1992 2021))

FUNCTION INDEX

CONSTANT INDEX

RECORD INDEX

VARIABLE INDEX

MACRO INDEX

PROPERTY INDEX