File created: 4-Jun-87 18:33:02 **{ERIS}<DANIELS>LISP>LFHACKS.;14**

changes to: (FUNCTIONS DETERMINE-SYSTEM-VOLUME CHASE-BOOT-LINKS GET-BOOT-POINTER PDA-TO-VP
VOL-NUM-CONTAINING-PAGE VOL-INDEX-CONTAINING-PAGE DETERMINE-BOOT-FILE-RUNS-USING-POINTERS
PRINT-RUNS-ATTRACTIVELY READ-BAD-PAGE-TABLE MAKE-PAGE-BAD UNMAKE-PAGE-BAD
DETERMINE-FILE-RUNS VP-TO-DA DA-TO-VP SHOW-VMEM-RUN-TABLE DEFAULT-BFT FETCH-LONG-CARDINAL
BOOTFILE-FD WRITE-PV-ROOT-PAGE MAX-BAD-PAGES WRITE-BAD-PAGE-TABLE BAD-PAGE-COUNT
LIST-FROM-BPT BPT-REF LIST-BAD-PAGES)
(VARS LFHACKSCOMS)
(VARIABLES BPT BFT-PILOT-BOOT-FILE BFT-GERM BFT-EMULATOR-MICROCODE BFT-DIAGNOSTIC-MICROCODE
+BOOT-FILE-TYPES+)
(SETFS BAD-PAGE-COUNT BPT-REF)
(COMMANDS "EC")
(STRUCTURES FILE-RUN)
(RECORDS |PilotDiskAddress| BAD-PAGE-TABLE BPT-ENTRY)

previous date: 3-Jun-87 18:31:15 {ERIS}<DANIELS>LISP>LFHACKS.;11

Read Table: XCL

Package: INTERLISP

Format: XCCS


```
(RPAQQ LFHACKSCOMS
       ((COMS (FUNCTIONS READ-LABEL)
              (VARIABLES BPT-LABEL PV-ROOT-PAGE-LABEL))
        (COMS (VARIABLES +BOOT-FILE-TYPES+ BFT-DIAGNOSTIC-MICROCODE BFT-EMULATOR-MICROCODE BFT-GERM
                     BFT-PILOT-BOOT-FILE)
              (DECLARE\: EVAL@COMPILE EVAL@LOAD DONTCOPY (RECORDS |PilotDiskAddress|))
              (FUNCTIONS VOL-NUM-CONTAINING-PAGE GET-BOOT-POINTER WRITE-PV-ROOT-PAGE BOOTFILE-FD DEFAULT-BFT
                     DETERMINE-SYSTEM-VOLUME FETCH-LONG-CARDINAL FILEDESC-FROM-NAME FIRST-VOLUME-PAGE VP-TO-DA
                     DA-TO-VP PDA-TO-VP))
        (COMS (DECLARE\: EVAL@COMPILE EVAL@LOAD DONTCOPY (RECORDS BAD-PAGE-TABLE BPT-ENTRY))
              (FUNCTIONS READ-BAD-PAGE-TABLE)
              (VARIABLES BPT)
              (FUNCTIONS BAD-PAGE-COUNT BPT-REF LIST-BAD-PAGES LIST-FROM-BPT MAKE-PAGE-BAD MAX-BAD-PAGES
                     UNMAKE-PAGE-BAD WRITE-BAD-PAGE-TABLE)
              (SETFS BAD-PAGE-COUNT BPT-REF))
        (COMS (STRUCTURES FILE-RUN)
              (FUNCTIONS DETERMINE-FILE-RUNS SHOW-VMEM-RUN-TABLE CHECK-PAGES-FREE PRINT-RUNS-ATTRACTIVELY)
              (FUNCTIONS CHASE-BOOT-LINKS DETERMINE-BOOT-FILE-RUNS-USING-POINTERS))
        (COMMANDS "EC")
        (VARIABLES DSKTW)
        (ADVICE |\\DoveDisk.HandleMajorError| |\\DoveDisk.TryRecalibrate| (\\DOVE.XFERDISK :IN \\DLDISK.EXECUTE)
               )
        (PROP FILETYPE LFHACKS)))


(CL:DEFUN READ-LABEL (PV-PAGE)
   (LET ((LABEL (CREATE |Label|)))
        (|\\PFTransferPage| PV-PAGE (NCREATE 'VMEMPAGEP)
              'VRR LABEL 1)
        LABEL))


(CL:DEFPARAMETER BPT-LABEL (READ-LABEL 1))


(CL:DEFPARAMETER PV-ROOT-PAGE-LABEL (READ-LABEL 0))


(CL:DEFCONSTANT +BOOT-FILE-TYPES+ '((BFT-DIAGNOSTIC-MICROCODE 0)
                                    (BFT-EMULATOR-MICROCODE 1)
                                    (BFT-GERM 2)
                                    (BFT-PILOT-BOOT-FILE 3)))


(CL:DEFCONSTANT BFT-DIAGNOSTIC-MICROCODE 0)


(CL:DEFCONSTANT BFT-EMULATOR-MICROCODE 1)


(CL:DEFCONSTANT BFT-GERM 2)


(CL:DEFCONSTANT BFT-PILOT-BOOT-FILE 3)

(DECLARE\: EVAL@COMPILE EVAL@LOAD DONTCOPY

(DECLARE\: EVAL@COMPILE

(BLOCKRECORD |PilotDiskAddress| ((HEAD BYTE)
                                 (SECTOR BYTE)
                                 (CYLINDER WORD)))
```

```
)
)


(CL:DEFUN VOL-NUM-CONTAINING-PAGE (PHYSICAL-PAGE-NUMBER)
   (FOR VOL-NUM FROM 0 TO (SUB1 (|fetch| (|PhysicalVolumeDescriptor| |subVolumeCount|) |of| |\\PhysVolumePage|))
      DO (LET ((SV-DESC (MESAELT (FETCH (|PhysicalVolumeDescriptor| |subVolumes|) OF |\\PhysVolumePage|)
                                 |SubVolumeArray| VOL-NUM)))
               (CL:WHEN (AND (IGEQ PHYSICAL-PAGE-NUMBER (FETCH (|SubVolumeDesc| |pvPage|) OF SV-DESC))
                             (ILESSP PHYSICAL-PAGE-NUMBER (IPLUS (FETCH (|SubVolumeDesc| |pvPage|) OF SV-DESC)
                                                                 (FETCH (|SubVolumeDesc| |nPages|) OF SV-DESC)))
                             )
                        (RETURN VOL-NUM)))))


(CL:DEFUN GET-BOOT-POINTER (VOL-NUM BFT)
   (CL:IF VOL-NUM
       (MESAELT (FETCH (|LogicalVolumeDescriptor| |bootingInfo|) OF (ELT |\\DFSLogicalVolumes| VOL-NUM))
              |LVBootFiles| BFT)
       (MESAELT (FETCH (|PhysicalVolumeDescriptor| |bootingInfo|) OF |\\PhysVolumePage|)
              |PVBootFiles| BFT)))


(CL:DEFUN WRITE-PV-ROOT-PAGE ()
   (|\\PFTransferPage| 0 |\\PhysVolumePage| 'VVW PV-ROOT-PAGE-LABEL 1))


(CL:DEFUN BOOTFILE-FD (&OPTIONAL VOLUME-NUM (BFT (DEFAULT-BFT)))
   (OR VOLUME-NUM
       ;; VOLUME-NUM = NIL means use the running sysout.
       (CL:SETF VOLUME-NUM (DETERMINE-SYSTEM-VOLUME)))
   (CREATE |FileDescriptor|
          |fileID| _ (FETCH-LONG-CARDINAL (FETCH (|DiskFileID| \fID) OF (MESAELT (FETCH (
                                                                                    |LogicalVolumeDescriptor|
                                                                                       |bootingInfo|)
                                                                                 OF (ELT
                                                                                      |\\DFSLogicalVolumes|
                                                                                        VOLUME-NUM))
                                                                          |LVBootFiles| BFT)))
          |volNum| _ VOLUME-NUM
          |type| _ |tDiagnosticMicrocode|))


(CL:DEFUN DEFAULT-BFT ()
   (CASE (MACHINETYPE)
       (DOVE BFT-GERM)
       (CL:OTHERWISE BFT-DIAGNOSTIC-MICROCODE)))


(CL:DEFUN DETERMINE-SYSTEM-VOLUME ()
   (LET* ((FIRST-RUN (LOCF (FETCH DLVMEMFILEINFO OF \\IOCBPAGE)))
          (BOOT-FILE-PAGE (DA-TO-VP (FETCH DLVMCYL OF FIRST-RUN)
                                    (FETCH DLVMHEAD OF FIRST-RUN)
                                    (FETCH DLVMSECTOR OF FIRST-RUN))))
         (VOL-NUM-CONTAINING-PAGE BOOT-FILE-PAGE)))


(CL:DEFUN FETCH-LONG-CARDINAL (PTR)
   (\\MAKENUMBER (\\GETBASE PTR 1)
          (\\GETBASE PTR 0)))


(CL:DEFUN FILEDESC-FROM-NAME (NAME)
   (LET ((FILESPEC (|\\LFFileSpec| NAME 'OLD))
         |volNum|)
        (CREATE |FileDescriptor|
               |fileID| _ (|\\LFReadFileID| (|\\LFGetDirectory| (SETQ |volNum| (|fetch| (|ExpandedName| VOLNUM)
                                                                                   |of| (|fetch| (|DFSFileSpec|
                                                                                                  EXPANDEDNAME)
                                                                                          |of| FILESPEC))))
                                           (|fetch| (|DFSFileSpec| FSDIRPTR) |of| FILESPEC))
               |volNum| _ |volNum|
               |type| _ |tLispFile|)))


(CL:DEFUN FIRST-VOLUME-PAGE (VOL-INDEX)
   (FETCH (|SubVolumeDesc| |pvPage|) OF (MESAELT (FETCH (|PhysicalVolumeDescriptor| |subVolumes|) OF
                                                                                        |\\PhysVolumePage|
                                                       )
                                                 |SubVolumeArray| VOL-INDEX)))


(DEFMACRO VP-TO-DA (VP)
   `(CL:LOCALLY (DECLARE (GLOBALVARS \\DLDISKSHAPE.SECTORSPERCYLINDER \\DLDISKSHAPE.SECTORSPERHEAD))
           (CL:MULTIPLE-VALUE-BIND (CYLINDER REM)
               (CL:FLOOR ,VP \\DLDISKSHAPE.SECTORSPERCYLINDER)
```

```
            (CL:MULTIPLE-VALUE-CALL 'LIST CYLINDER (CL:FLOOR REM \\DLDISKSHAPE.SECTORSPERHEAD)))))


(DEFMACRO DA-TO-VP (CYL HD SEC)
   `(CL:LOCALLY (DECLARE (GLOBALVARS \\DLDISKSHAPE.SECTORSPERCYLINDER \\DLDISKSHAPE.SECTORSPERHEAD))
          (IPLUS (ITIMES ,CYL \\DLDISKSHAPE.SECTORSPERCYLINDER)
                 (ITIMES ,HD \\DLDISKSHAPE.SECTORSPERHEAD)
                 ,SEC)))


(DEFMACRO PDA-TO-VP (PDA)
   `(LET ((PDA ,PDA))
         (DA-TO-VP (FETCH (|PilotDiskAddress| CYLINDER) OF PDA)
               (FETCH (|PilotDiskAddress| HEAD) OF PDA)
               (FETCH (|PilotDiskAddress| SECTOR) OF PDA))))

(DECLARE\: EVAL@COMPILE EVAL@LOAD DONTCOPY

(DECLARE\: EVAL@COMPILE

(MESAARRAY BAD-PAGE-TABLE ((0 127))
       BPT-ENTRY)

(MESARECORD BPT-ENTRY ((PAGE SWAPPEDFIXP)))
)
)


(CL:DEFUN READ-BAD-PAGE-TABLE (&OPTIONAL (TABLE BPT))
   (|\\PFTransferPage| 1 TABLE 'VVR BPT-LABEL 1))


(CL:DEFPARAMETER BPT (LET ((TABLE (NCREATE 'VMEMPAGEP)))
                          (READ-BAD-PAGE-TABLE TABLE)
                          TABLE))


(DEFINLINE BAD-PAGE-COUNT ()
   (FETCH (|PhysicalVolumeDescriptor| |badPageCount|) OF |\\PhysVolumePage|))


(DEFMACRO BPT-REF (INDEX)
   `(FETCH (BPT-ENTRY PAGE) OF (MESAELT BPT BAD-PAGE-TABLE ,INDEX)))


(CL:DEFUN LIST-BAD-PAGES (&OPTIONAL READ?)
   (AND READ? (READ-BAD-PAGE-TABLE))
   (CL:DOTIMES (I (BAD-PAGE-COUNT)
                  (TERPRI))
       (CL:FORMAT T "~D " (BPT-REF I))))


(CL:DEFUN LIST-FROM-BPT ()
   (FOR I FROM 0 TO (CL:1- (BAD-PAGE-COUNT)) COLLECT (BPT-REF I)))


(CL:DEFUN MAKE-PAGE-BAD (PHYSICAL-PAGE-NUMBER &OPTIONAL READ?)
   (AND READ? (READ-BAD-PAGE-TABLE))
   (LET ((BP-LIST (LIST-FROM-BPT)))
        (COND
           ((IGEQ (CL:LIST-LENGTH BP-LIST)
                  (MAX-BAD-PAGES))
            (CL:ERROR "Too many bad pages"))
           ((MEMBER PHYSICAL-PAGE-NUMBER BP-LIST)
            (CL:FORMAT *ERROR-OUTPUT* "~D already marked bad~%" PHYSICAL-PAGE-NUMBER))
           (T (LET ((NEW-BP-LIST (CL:MERGE 'LIST (LIST PHYSICAL-PAGE-NUMBER)
                                           BP-LIST
                                           'ILESSP)))
                   (FOR PAGE IN NEW-BP-LIST AS INDEX FROM 0 DO (CL:SETF (BPT-REF INDEX)
                                                                        PAGE)
                     FINALLY (CL:SETF (BAD-PAGE-COUNT)
                                      (CL:LIST-LENGTH NEW-BP-LIST))
                             (UNINTERRUPTABLY
                                 (WRITE-BAD-PAGE-TABLE)
                                 (WRITE-PV-ROOT-PAGE)))))))))


(DEFMACRO MAX-BAD-PAGES ()
   (FETCH (|PhysicalVolumeDescriptor| |maxBadPages|) OF |\\PhysVolumePage|))


(CL:DEFUN UNMAKE-PAGE-BAD (PHYSICAL-PAGE-NUMBER &OPTIONAL READ?)
   (AND READ? (READ-BAD-PAGE-TABLE))
   (LET ((BP-LIST (LIST-FROM-BPT)))
        (COND
           ((MEMBER PHYSICAL-PAGE-NUMBER BP-LIST)
            (CL:SETF BP-LIST (REMOVE PHYSICAL-PAGE-NUMBER BP-LIST))
```

```
            (FOR PAGE IN BP-LIST AS INDEX FROM 0 DO (CL:SETF (BPT-REF INDEX)
                                                             PAGE)
               FINALLY (CL:SETF (BAD-PAGE-COUNT)
                                (CL:LIST-LENGTH BP-LIST))
                       (UNINTERRUPTABLY
                           (WRITE-BAD-PAGE-TABLE)
                           (WRITE-PV-ROOT-PAGE))))
           (T (CL:FORMAT *ERROR-OUTPUT* "~D not in bad page table~%" PHYSICAL-PAGE-NUMBER)))))


(CL:DEFUN WRITE-BAD-PAGE-TABLE ()
   (|\\PFTransferPage| 1 BPT 'VVW BPT-LABEL 1))


(CL:DEFSETF BAD-PAGE-COUNT () (NEW-COUNT)
   `(CL:IF (> ,NEW-COUNT (MAX-BAD-PAGES))
        (CL:ERROR "Too many bad pages")
        (REPLACE (|PhysicalVolumeDescriptor| |badPageCount|) OF |\\PhysVolumePage| WITH ,NEW-COUNT)))


(CL:DEFSETF BPT-REF (INDEX) (NEW-VAL)
   `(REPLACE (BPT-ENTRY PAGE) OF (\\ADDBASE BPT (IPLUS ((OPENLAMBDA (|index|)
                                                          (OR (AND (ILEQ 0 |index|)
                                                                   (ILEQ |index| 127))
                                                              (ERROR '|indexOutOfRange|))
                                                          (ITIMES 2 (IDIFFERENCE |index| 0)))
                                                       ,INDEX)))
       WITH ,NEW-VAL))


(CL:DEFSTRUCT (FILE-RUN (:TYPE LIST)
                        (:CONC-NAME "FR-"))
   FILE-PAGE
   VOL-PAGE
   LENGTH)


(CL:DEFUN DETERMINE-FILE-RUNS (FILE-DESC)
   (LET ((FILE-LENGTH (|\\PFFindFileSize| FILE-DESC))
         (PAGE-RUNS NIL)
         (FILE-PAGE 0))
        (CL:LOOP (CL:PUSH (MAKE-FILE-RUN :FILE-PAGE FILE-PAGE :VOL-PAGE (|\\PFFindPageAddr| FILE-DESC FILE-PAGE)
                                  :LENGTH
                                  (DIFFERENCE (FETCH (|PageGroup| |nextFilePage|) OF (FETCH (|FileDescriptor|
                                                                                             PAGEGROUP)
                                                                                       OF FILE-DESC))
                                              FILE-PAGE))
                          PAGE-RUNS)
                 (SETQ FILE-PAGE (FETCH (|PageGroup| |nextFilePage|) OF (FETCH (|FileDescriptor| PAGEGROUP)
                                                                                OF FILE-DESC)))
                 (CL:WHEN (>= FILE-PAGE FILE-LENGTH)
                     (RETURN (REVERSE PAGE-RUNS))))))


(CL:DEFUN SHOW-VMEM-RUN-TABLE ()
   (LET ((LINKBASE (LOCF (FETCH (IOCBPAGE DLVMEMFILEINFO) OF \\IOCBPAGE))))
        (CL:FORMAT T "File Page Numbers  =>  Disk Page Numbers~%")
        (BIND (VP _ 0)
              END-OF-RUN-VP DA END-OF-RUN-DA RUN-LIST EACHTIME (CL:SETF DA (DA-TO-VP (FETCH (DLVMEMRUN DLVMCYL)
                                                                                        OF LINKBASE)
                                                                                  (FETCH (DLVMEMRUN DLVMHEAD)
                                                                                     OF LINKBASE)
                                                                                  (FETCH (DLVMEMRUN DLVMSECTOR)
                                                                                     OF LINKBASE)))
          WHILE (NEQ 0 (FETCH (DLVMEMRUN DLFIRSTFILEPAGE) OF (FETCH (DLVMEMRUN DLNEXTRUN) OF LINKBASE)))
          DO (CL:SETF END-OF-RUN-VP (CL:1- (FETCH (DLVMEMRUN DLFIRSTFILEPAGE) OF (FETCH (DLVMEMRUN DLNEXTRUN)
                                                                                     OF LINKBASE)))
                     END-OF-RUN-DA
                     (IPLUS DA (IDIFFERENCE END-OF-RUN-VP VP)))
             (CL:FORMAT T "[~D..~D]  =>  [~D..~D]~A~%" VP END-OF-RUN-VP DA END-OF-RUN-DA
                     (COND
                        ((SOME RUN-LIST #'(LAMBDA (PREV-ADDR-RANGE)
                                                  (AND (IGEQ DA (CAR PREV-ADDR-RANGE))
                                                       (ILEQ DA (CDR PREV-ADDR-RANGE)))))
                         " <= Entirely bogus VMem run!")
                        ((NOT (EQP (IDIFFERENCE END-OF-RUN-VP VP)
                                   (IDIFFERENCE END-OF-RUN-DA DA)))
                         " <= VMem run length doesn't match disk run length!")
                        (T "")))
             (PUSH RUN-LIST (CONS DA END-OF-RUN-DA))
             (CL:SETF VP (FETCH (DLVMEMRUN DLFIRSTFILEPAGE) OF (FETCH (DLVMEMRUN DLNEXTRUN) OF LINKBASE))
                     LINKBASE
                     (FETCH (DLVMEMRUN DLNEXTRUN) OF LINKBASE))
          FINALLY (CL:SETF END-OF-RUN-VP (FETCH (IFPAGE |DLLastVmemPage|) OF |\\InterfacePage|))
                (CL:FORMAT T "[~D..~D]  =>  [~D..~D]~%" VP END-OF-RUN-VP DA (IPLUS DA (IDIFFERENCE
                                                                                        END-OF-RUN-VP VP))
                        ))))
```

```
(CL:DEFUN CHECK-PAGES-FREE (VOL FILE-RUNS &OPTIONAL (ONE-AT-A-TIME? T))
    ;; Check that the labels for the given pages look good. Doesn't check the VAM yet.
    (FOR RUN IN FILE-RUNS DO (WITH-RESOURCE |\\DFSVAMjunkPage|
                                    (IF ONE-AT-A-TIME?
                                        THEN (FOR VOL-PAGE FROM (FR-VOL-PAGE RUN) AS COUNTER FROM 1
                                                  TO (FR-LENGTH RUN)
                                                  DO (PROCEED-CASE (|\\PFGetFreePage| VOL VOL-PAGE
                                                                                      |\\DFSVAMjunkPage| 1)
                                                                   (CONTINUE NIL :REPORT "Skip this page and continue"))
                                             )
                                        ELSE (|\\PFGetFreePage| VOL (FR-VOL-PAGE RUN)
                                                    |\\DFSVAMjunkPage|
                                                    (FR-LENGTH RUN))))))


(CL:DEFUN PRINT-RUNS-ATTRACTIVELY (FILE-RUNS &OPTIONAL VOL-NUM)
   (LET ((OFFSET (CL:IF VOL-NUM
                        (FIRST-VOLUME-PAGE VOL-NUM)
                        0)))
        (FOR RUN IN FILE-RUNS FIRST (CL:FORMAT T "File Page Numbers  =>  Disk Page Numbers~%")
           DO (CL:FORMAT T "[~D..~D]  =>  [~D..~D]~%" (FR-FILE-PAGE RUN)
                        (CL:1- (+ (FR-FILE-PAGE RUN)
                                  (FR-LENGTH RUN)))
                        (+ (FR-VOL-PAGE RUN)
                           OFFSET)
                        (CL:1- (+ (FR-VOL-PAGE RUN)
                                  OFFSET
                                  (FR-LENGTH RUN)))))))


(CL:DEFUN CHASE-BOOT-LINKS (FN &KEY VOL-NUM (BFT (DEFAULT-BFT))
                                    VERBOSE)
    ;; runs through the bootfile starting from the appropriate boot pointer, using the LV boot pointer is a particular volume is specified, following the boot
    ;; links. FN is called on each page with a physical page number, file page number, and file id. If verbose is true, will print something every 100 pages.
    (LET ((BOOT-POINTER (GET-BOOT-POINTER VOL-NUM BFT)))
         (CL:WHEN (CL:ZEROP (FETCH (|DiskFileID| |da|) OF BOOT-POINTER))
                (CL:ERROR "No boot pointer found."))
         (WITH-RESOURCE |label| (BIND (CORRECT-ID _ (FETCH-LONG-CARDINAL (FETCH (|DiskFileID| \fID)
                                                                              OF BOOT-POINTER)))
                                      (LAST-BOOT-FILE-PAGE _ (CL:1- (|\\PFFindFileSize| (BOOTFILE-FD VOL-NUM BFT)
                                                                                        )))
                                      (VP _ (PDA-TO-VP (FETCH (|DiskFileID| |da|) OF BOOT-POINTER)))
                                      (FP _ (CL:1- (FETCH (|DiskFileID| |firstPage|) OF BOOT-POINTER)))
                                      (BUFFER _ (NCREATE 'VMEMPAGEP))
                                     FILE-ID FIRST (CL:WHEN VERBOSE (CL:PRINC "Processing bootfile"
                                                                              *ERROR-OUTPUT*))
                                 FOR PAGE-NUM FROM 0
                                 DO ;; Read next page
                                    (CL:WHEN (EQL (CL:MOD FP 100)
                                                  99)
                                        (CL:WHEN VERBOSE (CL:PRINC "." *ERROR-OUTPUT*))
                                        (BLOCK))
                                    (LET ((STATUS (|\\PFTransferPage| VP BUFFER 'VRR |label| 1)))
                                         (CL:WHEN (NOT (EQ STATUS 'OK))
                                                (CL:CERROR "Continue processing the file" "Can't read page ~D:
                                                           status = ~S" VP STATUS)))
                                    (CL:WHEN (NOT (EQL (CL:1+ FP)
                                                       (FETCH (|Label| |filePage|) OF |label|)))
                                        (CL:CERROR "Continue processing the file" "Boot file pages not
                                               contiguous: prev = ~D, current = ~D" FP (FETCH (|Label|
                                                                                                 |filePage|
                                                                                                 )
                                                                                              OF |label|)))
                                    (CL:WHEN (NOT (EQL (CL:SETF FILE-ID (FETCH (|Label| |fileID|) OF |label|))
                                                       CORRECT-ID))
                                           (CL:CERROR "Continue processing the file" "File id in label (~D)
                                                  doesn't match boot pointer (~D)" FILE-ID CORRECT-ID))
                                    (CL:SETF FP (FETCH (|Label| |filePage|) OF |label|))
                                    (CL:FUNCALL FN VP FP PAGE-NUM FILE-ID)
                                    (COND
                                       ((AND (EQL -1 (FETCH (|PilotDiskLabel| |BootLinkA|) OF |label|))
                                             (EQL -1 (FETCH (|PilotDiskLabel| |BootLinkB|) OF |label|)))
                                        (CL:WHEN VERBOSE (CL:PRINC "<boot link all 1's> " *ERROR-OUTPUT*))
                                        (RETURN))
                                       ((IGEQ FP LAST-BOOT-FILE-PAGE)
                                        (CL:WHEN VERBOSE (CL:PRINC "<end of file> " *ERROR-OUTPUT*))
                                        (RETURN))
                                       ((AND (CL:ZEROP (FETCH (|PilotDiskLabel| |BootLinkA|) OF |label|))
                                             (CL:ZEROP (FETCH (|PilotDiskLabel| |BootLinkB|) OF |label|)))
                                        ;; No boot link - continue to next page
                                        (CL:INCF VP))
```

```
                                        (T ;; Have a real boot link - jump to new disk address
                                           (CL:SETF VP (PDA-TO-VP (\\MAKENUMBER (FETCH (|PilotDiskLabel|
                                                                                        |BootLinkB|)
                                                                                  OF |label|)
                                                                                (FETCH (|PilotDiskLabel| |BootLinkA|)
                                                                                  OF |label|))))
                                           (CL:WHEN VERBOSE (CL:FORMAT *ERROR-OUTPUT* "<Jump to ~D>" VP))))))
        (CL:WHEN VERBOSE
            (CL:PRINC "done." *ERROR-OUTPUT*)
            (CL:TERPRI *ERROR-OUTPUT*))))

(CL:DEFUN DETERMINE-BOOT-FILE-RUNS-USING-POINTERS (&REST KEY-ARGS &KEY VOL-NUM (BFT (DEFAULT-BFT))
                                                                      VERBOSE)
   (LET ((OFFSET (FIRST-VOLUME-PAGE (VOL-NUM-CONTAINING-PAGE (PDA-TO-VP (FETCH (|DiskFileID| |da|)
                                                                            OF (GET-BOOT-POINTER VOL-NUM BFT))
                                                                     ))))
         (RUN-LIST NIL)
         LAST-VP RUN)
        (CL:APPLY 'CHASE-BOOT-LINKS #'(CL:LAMBDA (VP FP PAGE-NUM FILE-ID)
                                         (DECLARE (IGNORE PAGE-NUM FILE-ID))
                                         (CL:FLET ((NEW-RUN (FP VP)
                                                            (CL:PUSH (CL:SETF RUN (MAKE-FILE-RUN :FILE-PAGE FP
                                                                                                 :VOL-PAGE
                                                                                                 (- VP OFFSET)
                                                                                                 :LENGTH 1))
                                                                     RUN-LIST)
                                                            (CL:SETF LAST-VP VP)))
                                                   (COND
                                                      ((NULL LAST-VP)
                                                       (NEW-RUN FP VP))
                                                      ((EQL VP (CL:INCF LAST-VP))
                                                       (CL:INCF (FR-LENGTH RUN)))
                                                      (T (NEW-RUN FP VP)))))
                KEY-ARGS)
        (REVERSE RUN-LIST)))

(DEFCOMMAND "EC" (EXPRESSION)
   ;; "eval compiled"
    (CL:FUNCALL (CL:COMPILE NIL `(CL:LAMBDA NIL ,EXPRESSION))))

(DEFGLOBALVAR DSKTW)

(XCL:REINSTALL-ADVICE '|\\DoveDisk.HandleMajorError| :BEFORE '((:LAST (PRIN2 'H DSKTW))))

(XCL:REINSTALL-ADVICE '|\\DoveDisk.TryRecalibrate| :BEFORE '((:LAST (PRIN2 'R DSKTW))))

(XCL:REINSTALL-ADVICE '(\\DOVE.XFERDISK :IN \\DLDISK.EXECUTE)
       :AFTER
       '((:LAST (IF (EQ !VALUE 'OK)
                    THEN (PRIN2 '+ DSKTW)
                  ELSE (PRIN2 '- DSKTW)))))

(PUTPROPS LFHACKS FILETYPE :COMPILE-FILE)

(PUTPROPS LFHACKS COPYRIGHT ("Xerox Corporation" 1987))
```

FUNCTION INDEX

CONSTANT INDEX

ADVICE INDEX

MACRO INDEX

VARIABLE INDEX

SETF INDEX

PROPERTY INDEX

COMMAND INDEX

STRUCTURE INDEX

RECORD INDEX