File created: 3-Nov-2020 16:13:17 **{DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCLOCALDEVICE.;5**

previous date: 9-Jan-94 18:53:55 {DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCLOCALDEVICE.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCLOCALDEVICECOMS
       (
```

;;; Implements Local Single User Device

;;; The LOCALSINGLEUSER device vector.

```
        [COMS                                                    ; The basic device vector functions.
              (FNS NCLocalDevice.OpenNoteFile NCLocalDevice.CloseNoteFile NCLocalDevice.CreateNoteFile
                   NCLocalDevice.CompactNoteFile NCLocalDevice.InspectAndRepairNoteFile
                   NCLocalDevice.BuildHashArray NCLocalDevice.NoteFileOpenP NCLocalDevice.CheckpointNoteFile
                   NCLocalDevice.ConvertNoteFileFormat NCLocalDevice.TruncateNoteFile NCLocalDevice.NewCardUID
                   NCLocalDevice.MarkCardDeleted NCLocalDevice.GetCardInfo NCLocalDevice.PutCardPart
                   NCLocalDevice.GetCardPart NCLocalDevice.ObtainWritePermission
                   NCLocalDevice.ReleaseWritePermission NCLocalDevice.CancelCacheSubscription
                   NCLocalDevice.DeleteNoteFile)
                                                                 ; Install the device vector.
              (GLOBALVARS NC.DeviceVectorsHashArray)
              (ADDVARS (DEFAULTFILETYPELIST (NOTEFILE . BINARY)))
              (DECLARE%: DONTEVAL@LOAD (P (PUTHASH 'LOCALSINGLEUSER (create NoteFileDevice OpenNoteFileFn _
                                                                            (FUNCTION NCLocalDevice.OpenNoteFile)
                                                                            CloseNoteFileFn _
                                                                            (FUNCTION NCLocalDevice.CloseNoteFile
                                                                                  )
                                                                            CreateNoteFileFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.CreateNoteFile)
                                                                            DeleteNoteFileFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.DeleteNoteFile)
                                                                            CompactNoteFileFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.CompactNoteFile)
                                                                            RepairNoteFileFn _
                                                                            (FUNCTION
                                                                           NCLocalDevice.InspectAndRepairNoteFile
                                                                                  )
                                                                            BuildHashArrayFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.BuildHashArray)
                                                                            NoteFileOpenPFn _
                                                                            (FUNCTION NCLocalDevice.NoteFileOpenP
                                                                                  )
                                                                            CheckpointNoteFileFn _
                                                                            (FUNCTION
                                                                                 NCLocalDevice.CheckpointNoteFile
                                                                                  )
                                                                            ConvertNoteFileFormatFn _
                                                                            (FUNCTION
                                                                              NCLocalDevice.ConvertNoteFileFormat
                                                                                  )
                                                                            TruncateNoteFileFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.TruncateNoteFile
                                                                                  )
                                                                            NewCardUIDFn _ (FUNCTION
                                                                                        NCLocalDevice.NewCardUID
                                                                                           )
                                                                            MarkCardDeletedFn _
                                                                            (FUNCTION
                                                                                  NCLocalDevice.MarkCardDeleted)
                                                                            GetCardInfoFn _
                                                                            (FUNCTION NCLocalDevice.GetCardInfo)
                                                                            PutCardPartFn _
                                                                            (FUNCTION NCLocalDevice.PutCardPart)
                                                                            GetCardPartFn _
                                                                            (FUNCTION NCLocalDevice.GetCardPart)
                                                                            ObtainWritePermissionFn _
                                                                            (FUNCTION
```

```
                                                                    NCLocalDevice.ObtainWritePermission
                                                                          )
                                                                   ReleaseWritePermissionFn _
                                                                   (FUNCTION
                                                                    NCLocalDevice.ReleaseWritePermission
                                                                          )
                                                                   CancelCacheSubscriptionFn _
                                                                   (FUNCTION
                                                                   NCLocalDevice.CancelCacheSubscription
                                                                          ))
                                                NC.DeviceVectorsHashArray]
```

;;; Version information for LOCALSINGLEUSER device.

```
        [COMS (GLOBALVARS NCLocalDevice.NoteFileVersionsList NCLocalDevice.CurrentNoteFileVersionNumber)
              (FNS NCLocalDevice.CurrentVersion)
              (INITVARS (NCLocalDevice.CurrentNoteFileVersionNumber 3)
                     (NCLocalDevice.NoteFileVersionsList (LIST (create NoteFileVersion Version _ 3
                                                                       NumberOfReservedCards _ 20
                                                                       NoteFileIndexWidth _ 28 NoteFileHeaderSize
                                                                       _ 30]
```

;;; Supporting functions and variables

```
        (COMS (FNS NCLocalDevice.GetNoteFileHeader NCLocalDevice.GetHashArray NCLocalDevice.GetSpecialCardUIDs
                   NCLocalDevice.PutNoteFileHeader NCLocalDevice.PutHashArray NCLocalDevice.PutIndexEntry
                   NCLocalDevice.NoteFileVersionOkayP NCLocalDevice.CheckForPlausibleNoteFileHeader
                   NCLocalDevice.NoteFileNeedsTruncationP NCLocalDevice.OpenFailed NCLocalDevice.ReadIndexEntry
                   NCLocalDevice.SaveInformationPastCheckpoint NCLocalDevice.PutFromLinks
                   NCLocalDevice.TotalNoteFileIndexSize))
        [DECLARE%: DONTEVAL@LOAD (P (NC.StoreAutoloadFnFile (FUNCTION NC.ScavengerPhase1)
                                          'NCREPAIR
                                          'NOTECARDSDIRECTORIES)
                                    (NC.StoreAutoloadFnFile (FUNCTION NC.CompactNoteFileInPlace)
                                          'NCCOMPACT
                                          'NOTECARDSDIRECTORIES)
                                    (NC.StoreAutoloadFnFile (FUNCTION NC.CompactNoteFileToTarget)
                                          'NCCOMPACT
                                          'NOTECARDSDIRECTORIES]
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCLOCALDEVICE)))
```

;;; Implements Local Single User Device

;;; The LOCALSINGLEUSER device vector.

;; The basic device vector functions.

```
(DEFINEQ
```

(NCLocalDevice.OpenNoteFile

```
  [LAMBDA (NoteFile PromptWindow Don'tCheckForTruncationFlg)     (* rht%: " 7-Jul-87 18:08")
```

(* * OpenNoteFileFn for the local, single user device.)

(* * fgh |5/22/86| First created.)

(* * fgh |9/1/86| Reimplemented ReadOnly notefile.)

(* * rht 10/31/86%: Added Don'tCheckForTruncationFlg arg.)

(* * rht 6/8/87%: Switched order of checks of version num and plausibility.
It was also ignoring the error value returned by NCLocalDevice.CheckForPlausibleNoteFileHeader.)

(* * rht 7/7/87%: Undid first half of previous change. I.e. order of checks of version num and plausibility is back to the way it used to be.)

```
    (OR [CAR (ERSETQ (PROG ((FullName (INFILEP (fetch (NoteFile FullFileName) of NoteFile)))
                            EofPtr CriticalUIDs Stream ReturnVal)
```

(* * First check for possible errors.)

```
                           (if (NULL FullName)
                               then
```

(* * Error%: NoteFileNotFound, Return)

```
                                   (RETURN 'NoteFileNotFound)
                               elseif (OPENP FullName)
                                 then
```

(* * Error%: NoteFile already open, Return)

```
                                   (RETURN 'NoteFileAlreadyOpen)
                               elseif [NULL (ERSETQ (SETQ Stream (OPENSTREAM FullName
```

```
                                                                    (if (fetch (NoteFile ReadOnlyFlg) of NoteFile)
                                                                        then 'INPUT
                                                                      else 'BOTH)
                                                                    'OLD
                                                                    '((TYPE BINARY]
                                then

        (* * Error%: File won't open)

                                    (RETURN 'NoteFileOpenFailed))

        (* * All is okay for now, create fill in the NoteFile object a bit.)

                                (replace (NoteFile Stream) of NoteFile with Stream)
                                (replace (NoteFile FullFileName) of NoteFile with FullName)
                                (NC.SetMonitor NoteFile (CREATE.MONITORLOCK FullName))

        (* * Make sure stream is not closed by CLOSEALLs)

                                (WHENCLOSE Stream 'CLOSEALL 'NO)

        (* * Get the header and check for correct version, correct checkpoint, & plausibility.)

                                (NCLocalDevice.GetNoteFileHeader NoteFile)
                                (SETQ EofPtr (GETEOFPTR (fetch (NoteFile Stream) of NoteFile)))
                                [COND
                                   [(NOT (NCLocalDevice.NoteFileVersionOkayP NoteFile))
                                    (RETURN (NCLocalDevice.OpenFailed NoteFile 'NoteFileNeedsConversion]
                                   ([NOT (type? NoteFile (SETQ ReturnVal (NCLocalDevice.CheckForPlausibleNoteFileHeader
                                                                          NoteFile EofPtr]
                                    (RETURN (NCLocalDevice.OpenFailed NoteFile ReturnVal)))
                                   ((AND (NULL Don'tCheckForTruncationFlg)
                                         (NCLocalDevice.NoteFileNeedsTruncationP NoteFile EofPtr))
                                    (RETURN (NCLocalDevice.OpenFailed NoteFile 'NoteFileNeedsTruncation]

        (* * Go get the first six card UIDs on the file.)

                                (SETQ CriticalUIDs (NCLocalDevice.GetSpecialCardUIDs NoteFile))
                                (replace (NoteFileCriticalUIDs NoteFile) of CriticalUIDs with (fetch (NoteFile UID)
                                                                                                  of NoteFile))

        (* * Return with list of special uids and NF-UID.)

                                (RETURN CriticalUIDs]
    (PROGN

        (* * Open must have failed. Return a message to that effect.)

            (NCLocalDevice.OpenFailed NoteFile])
```

(NCLocalDevice.CloseNoteFile

```
  [LAMBDA (NoteFile PromptWindow)                                   (* fgh%: "26-May-86 22:36")

        (* * Close the file for a local device NoteFile.)

        (* * fgh |5/26/86| First created.)

    (if (NOT (NCLocalDevice.NoteFileOpenP NoteFile))
        then 'NoteFileNotOpen
      elseif (ERSETQ (CLOSEF (fetch (NoteFile Stream) of NoteFile)))
        then (replace (NoteFile Stream) of NoteFile with NIL)
             NoteFile
      else 'CloseFailed])
```

(NCLocalDevice.CreateNoteFile

```
  [LAMBDA (NoteFile SizeInCards InterestedWindow OperationMsg QuietFlg)
                                                                    ; Edited 20-Oct-88 14:25 by RAR

        (* * Create an empty NoteFile on the local device.)

        (* * fgh |9/1/86| First created from NC.CloseDatabaseFile.)

    ;; RAR. 10/20/88 Remove the version number if creating a notefile on

    (DECLARE (GLOBALVARS NCLocalDevice.CurrentVersionNumber NC.DefaultIndexSizeInEntries))
    (LET ((HashArraySize (OR (FIXP SizeInCards)
                             NC.DefaultIndexSizeInEntries))
          (FileName (fetch (NoteFile FullFileName) of NoteFile))
          Stream)
         (OR [CAR (ERSETQ (PROG NIL

        (* * Open the file.)

                                 (if (EQ (FILENAMEFIELD FileName 'HOST)
                                         'CORE)
                                     then (SETQ FileName (PACKFILENAME 'VERSION NIL 'BODY FileName)))
```

```
                              [COND
                                 ([NULL (SETQ Stream (CAR (ERSETQ (OPENSTREAM FileName 'BOTH 'NEW
                                                                          '((TYPE BINARY]
                                  (RETURN 'FileWon'tOpen]
```

(* * Fix up the NoteFile object with the necessary information about the file about to be created.)

```
                              (replace (NoteFile Stream) of NoteFile with Stream)
                              (replace (NoteFile FullFileName) of NoteFile with (FULLNAME Stream))
                              (replace (NoteFile HashArraySize) of NoteFile with HashArraySize)
                              (replace (NoteFile Version) of NoteFile with
                                                                  NCLocalDevice.CurrentNoteFileVersionNumber
                                    )
                              (replace (NoteFile CheckptPtr) of NoteFile with (NCLocalDevice.TotalNoteFileIndexSize
                                                                                HashArraySize))
                              [replace (NoteFile NextIndexNum) of NoteFile
                                 with (CONSTANT (ADD1 (fetch (NoteFileVersion NumberOfReservedCards)
                                                         of (NCLocalDevice.CurrentVersion]
```

(* * Write the header down to the file.)

```
                              (NCLocalDevice.PutNoteFileHeader NoteFile)
```

(* * Write an empty index onto the file.)

```
                              [for CTR from 1 to HashArraySize eachtime (BLOCK)
                                 do [OR QuietFlg (AND (EQ 1 (IMOD CTR 500))
                                                      (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                             "Creating NoteFile."
                                                             (CHARACTER 13)
                                                             "Processing item " CTR " out of " HashArraySize
                                                             "." (CHARACTER 13]
                                    (NC.WriteStatus Stream 'FREE)
                                    (SETFILEPTR Stream (PLUS (GETFILEPTR Stream)
                                                             (CONSTANT (SUB1 (fetch (NoteFileVersion
                                                                                      NoteFileIndexWidth)
                                                                               of (NCLocalDevice.CurrentVersion]
```

(* Move NextIndexNum back to the beginning so that special cards will have the correct index nums.)

```
                              (RETURN NoteFile]
                 (PROGN
```

(* * Hmm. Create failed for some reason.)

```
                          (CLOSEF Stream (DELFILE (FULLNAME Stream)))
                          'CreateFailed])
```

(NCLocalDevice.CompactNoteFile

```
  [LAMBDA (FromNoteFile ToFileName InPlaceFlg PromptWindow)      ; Edited  8-Dec-88 18:06 by krivacic
```

;;; Compact a NoteFile. If InPlaceFlg is T calls NC.CompactNoteFileInPlace. Otherwise if ToFileName is NIL, asks for a new file name.

;;; fkr 11/8/85 Updated to handle new CardID scheme and NoteFile object.

;;; kirk 19Nov85: Created from NC.CompactDatabaseInPlace to handle new NoteFile format

;;; fgh 5/186 Totally rewritten to get rid of numerous bugs. Added new PromptWindow parameter.

;;; rht 7/2/86: Fixed bug in call to NC.CompactToTarget and NC.CompactInPlace. They were being called with FromNoteFile instead of (OR
;;; FromNoteFile FromFileName).

;;; kirk 3Jul86 Added SETQ NC.DatabaseFileNameSuggestion

;;; rht 10/16/86: Now autoloads NCREPAIR.

;;; rht 11/3/86: No longer reopens if was originally open. Also now passes PromptWindow along to called functions.

;;; pmi 5/27/87: Now returns the target notefile.

;;; pmi 6/24/87: No longer tries to calculate full file name for target file --- let create notefile worry about that later.

```
    (DECLARE (GLOBALVARS NC.DatabaseFileNameSuggestion))
    (LET (FromFileName ToNoteFile)
```

;;; Get the name of the file to be compacted

```
         (SETQ FromFileName (COND
                               ((NULL FromNoteFile)
                                (PROG1 (NC.DatabaseFileName "Name of NoteFile to be compacted:" " -- " T NIL NIL
                                              PromptWindow)
                                       (NC.ClearMsg PromptWindow)))
                               ((type? NoteFile FromNoteFile)
                                (fetch (NoteFile FullFileName) of FromNoteFile))
                               (T FromNoteFile)))
```

```
;;; If compact to target, get the name of the target file

        [if (NULL InPlaceFlg)
            then (SETQ NC.DatabaseFileNameSuggestion (PACKFILENAME 'VERSION NIL 'BODY (FULLNAME FromFileName)))
                 (SETQ ToFileName (OR ToFileName (PROG1 (NC.DatabaseFileName "Name of target of compaction:"
                                                                  " -- " T NIL NIL PromptWindow)
                                                        (NC.ClearMsg PromptWindow]

;;; As long as you have file names, go ahead!

        (if (AND FromFileName (OR InPlaceFlg ToFileName))
            then

;;; Make full name for source file.

                (SETQ FromFileName (FULLNAME FromFileName 'OLD))
                                                                  ; SETQ ToFileName (FULLNAME ToFileName (QUOTE NEW))

;;; Close the file if its open

                (if (AND (SETQ FromNoteFile (NC.NoteFileFromFileName FromFileName))
                         (OPENP FromFileName))
                    then (NC.CloseDatabaseFile FromNoteFile))

;;; Compact the file and reopen if successfull and was previously open

                (NC.PrintMsg PromptWindow T "Compacting " FromFileName " ...")
                (if (SETQ ToNoteFile (if InPlaceFlg
                                         then (NC.AutoloadApply* (FUNCTION NC.CompactNoteFileInPlace)
                                                    (OR FromNoteFile FromFileName)
                                                    PromptWindow)
                                       else                       ; compact to target
                                            (NC.AutoloadApply* (FUNCTION NC.CompactNoteFileToTarget)
                                                   (OR FromNoteFile FromFileName)
                                                   ToFileName PromptWindow)))
                    then (NC.ClearMsg PromptWindow T)))
        ToNoteFile])
```

(NCLocalDevice.InspectAndRepairNoteFile

```
  [LAMBDA (NoteFileOrFileName ReadSubstancesFlg InterestedWindow)
                                                                  ; Edited  8-Dec-88 16:41 by krivacic

;;; Check to be sure file is closed before calling real inspect and repair.

;;; rht 7/16/86: Added InterestedWindow arg.  Removed call to NC.OpenDatabaseFile.

;;; rht 7/17/86: Now works with file name args as well as notefile args.  Took out reopen'ing of notefile, because you don't know how it was originally
;;; opened.

;;; rht 10/15/86: Now only calls NC.NoteFileOpenP on real notefile objects.

;;; rht 10/16/86: Now autoloads NCREPAIR.

;;; pmi 7/7/87: Now creates an event for use in signalling the end of I&R.

;;; rg 8/11/87 removed protection wrapper and AWAIT.EVENT, moved to lower level call

    (LET [(NoteFile (if (type? NoteFile NoteFileOrFileName)
                        then NoteFileOrFileName
                      else (NC.NoteFileFromFileName NoteFileOrFileName]
         (if (AND NoteFile (NC.NoteFileOpenP NoteFile))
             then (NC.CloseNoteFile NoteFile InterestedWindow))
         (NC.AutoloadApply* (FUNCTION NC.ScavengerPhase1)
                NoteFileOrFileName ReadSubstancesFlg NIL NIL InterestedWindow])
```

(NCLocalDevice.BuildHashArray

```
  [LAMBDA (NoteFile QuietFlg InterestedWindow OperationMsg)       ; Edited  3-Dec-87 18:59 by rht:

          (* * Build a hash array for a local device notefile)

          (* * fgh |5/23/86| First created.)

          (* * fgh |9/1/86| Reimplemented InterestedWindow, QuietFlg (\, &) OperationMsg args.)

    (OR [CAR (ERSETQ (PROGN (replace (NoteFile HashArray) of NoteFile with (NC.CreateUIDHashArray
                                                                              (fetch (NoteFile HashArraySize)
                                                                                 of NoteFile)))
                            (NCLocalDevice.GetHashArray NoteFile QuietFlg InterestedWindow OperationMsg]
        (PROGN

          (* * Build hash array failed for some reason.. This NF is definitely corrupted.
          Close it up and report the error.)

               (AND (STREAMP (fetch (NoteFile Stream) of NoteFile))
                    (CLOSEF? (fetch (NoteFile Stream) of NoteFile)))
```

```
            (replace (NoteFile Stream) of NoteFile with NIL)
            (CLRHASH (fetch (NoteFile HashArray) of NoteFile))
            (replace (NoteFile HashArray) of NoteFile with NIL)
            (NC.SetMonitor NoteFile NIL)
            'ErrorOfUnknownOrigin])
```

(NCLocalDevice.NoteFileOpenP

```
[LAMBDA (NoteFile)                                          (* fgh%: "26-May-86 22:27")
```

(* * Is thsi local device NoteFile Open?)

(* * fgh |5/26/86| First created.)

```
  (AND (STREAMP (fetch (NoteFile Stream) of NoteFile))
       (OPENP (fetch (NoteFile Stream) of NoteFile))
       NoteFile])
```

(NCLocalDevice.CheckpointNoteFile

```
[LAMBDA (NoteFile InterestedWindow OperationMsg QuietFlg)       (* Randy.Gobbel " 6-Jan-87 11:26")
```

(* * Checkpoint a local device notefile.)

(* * fgh |5/26/86| First created.)

(* * fgh |9/1/86| Reimplemented QuietFlg, etc.)

(* * rg |1/6/87| Added FORCEOUTPUT)

```
  (if (NOT (NCLocalDevice.NoteFileOpenP NoteFile))
      then 'NoteFileNotOpen
    elseif [NULL (ERSETQ (PROGN (NCLocalDevice.PutHashArray NoteFile InterestedWindow NIL OperationMsg QuietFlg)
                                (replace (NoteFile CheckptPtr) of NoteFile with (GETEOFPTR (fetch (NoteFile Stream)
                                                                                                  of NoteFile)))
                                (NCLocalDevice.PutNoteFileHeader NoteFile)
                                (FORCEOUTPUT (fetch (NoteFile Stream) of NoteFile)
                                       T]
      then 'CheckpointFailed
    else NoteFile])
```

(NCLocalDevice.ConvertNoteFileFormat

```
[LAMBDA (NoteFile PromptWindow)                                 (* rht%: " 6-Nov-86 12:00")
```

(* * Convert a NoteFile from old version to curretn version format.)

(* * fgh |5/25/86| First created on basis of old NC.CheckForNeededConversion.)

(* * rht 11/6/86%: Now passes an InterestedWindow argument to the converter.)

```
  (PROG ((Stream (fetch (NoteFile Stream) of NoteFile))
         (Version (fetch (NoteFile Version) of NoteFile))
         (FullFileName (fetch (NoteFile FullFileName) of NoteFile))
         NewFileName)
```

(* * Check if NoteFile is open. If so, its an error.)

```
        (if (OPENP FullFileName)
            then (RETURN 'NoteFileAlreadyOpen))
```

(* * Is Version a sensible version number? If not, then just report an error.)

```
        (if [OR (NOT (NUMBERP Version))
                (GEQ Version (CONSTANT (fetch (NoteFileVersion Version) of (NCLocalDevice.CurrentVersion]
            then (RETURN 'BadVersionNumber))
```

(* * Is this a really old version. If so then we can't handle it.)

```
        (if (LEQ Version 1)
            then (if (WINDOWP PromptWindow)
                     then (NC.PrintMsg PromptWindow T "Notefile " FullFileName " is  too old (version " Version
                                       ")." (CHARACTER 13)
                                       "  It must first be converted to version 2" "by opening in NoteCards release
                                       1.2i." (CHARACTER 13)
                                       " Then we convert to version 3."
                                       (CHARACTER 13)))
                 (RETURN 'NoteFileTooOld))
```

(* * Okay convert the NoteFile.)

```
        (if [CAR (ERSETQ (SETQ NewFileName (NC.AutoloadApply* (FUNCTION NC.ConvertNoteFileVersion2To3)
                                                              FullFileName PromptWindow]
            then
```

(* * Conversion went okay. Return the NoteFile with the new fullfilename.)

```
                (replace (NoteFile FullFileName) of NoteFile with NewFileName)
                (RETURN NoteFile)
          else
```

(* * something went wrong during the conversion. return an error indicator.)

```
        (CLOSEF? (fetch (NoteFile FullFileName) of NoteFile))
        (RETURN 'NoteFileConversionFailed])
```

(NCLocalDevice.TruncateNoteFile

```
[LAMBDA (NoteFile PromptWindow)                                    (* rht%: "19-Feb-87 12:23")
```

(* * Truncate a local device NoteFile back to the checkpoint pointer, offering the user the option of saving the changes in another file.)

(* * fgh |5/25/86| First created on basis of old NC.CheckForNeededtruncation.)

(* * rht 2/19/87%: Now properly binds FullFileName.)

(* * "sye: 2/Dec/88 Setfileinfo should take a stream instead of a filename as an input arg")

```
  (PROG ((FullFileName (fetch (NoteFile FullFileName) of NoteFile))
         (NoteFileStream (fetch (NoteFile Stream) of NoteFile))
         SaveStream ReturnValue)
```

(* * Make sure PromptWindow is a window)

```
        (OR (WINDOWP PromptWindow)
            (SETQ PromptWindow))
```

(* * Check to make NoteFile is not open. If it is, then error.)

```
        (if (AND NoteFileStream (OPENP NoteFileStream))
            then (RETURN 'NoteFileAlreadyOpen))
```

(* * First ask the user if they want to save the info past the checkpoint into a file.
If there's no PromptWindow, don't bother to ask.)

```
        [if (AND PromptWindow (NC.AskYesOrNo "Want to save info beyond checkpoint to a file? " "--" "Yes" NIL
                                    PromptWindow NIL T))
            then (if [NOT (type? NoteFile (SETQ ReturnValue (NCLocalDevice.SaveInformationPastCheckpoint NoteFile
                                                                    PromptWindow]
                     then                                    (* Save failed for some reason. Notify the user and return.)
                          (NC.PrintMsg PromptWindow T "Attempt to save information failed because " ReturnValue
                                "." (CHARACTER 13))
                          (RETURN 'NoteFileTruncationFailed]
```

(* * Now truncate. Confirming with the user first if there's a prompt window.)

```
        (if (OR (NOT PromptWindow)
                (NC.AskYesOrNo (CONCAT (CHARACTER 13)
                                       "Are you sure you want to truncate " FullFileName "? ")
                       "--" "No" NIL PromptWindow NIL T))
            then
```

(* * Notify the user if appropriate.)

```
                (AND PromptWindow (NC.PrintMsg PromptWindow T "Truncating file " FullFileName " ..."))
```

(* * Do the truncation.)

```
                (if [CAR (ERSETQ (PROG2 [SETQ NoteFileStream (OPENSTREAM FullFileName 'BOTH 'OLD
                                                                         '((TYPE BINARY]
                                        (SETFILEINFO NoteFileStream 'LENGTH (fetch (NoteFile CheckptPtr)
                                                                               of NoteFile))
                                        (CLOSEF NoteFileStream]
                    then
```

(* * truncation succeeded.)

```
                        (AND PromptWindow (NC.PrintMsg PromptWindow T "Done." (CHARACTER 13)))
                        (RETURN NoteFile)
                  else
```

(* * truncation failed. Notify user.)

```
                    (AND PromptWindow (NC.PrintMsg PromptWindow NIL "Couldn't truncate " FullFileName "."
                                                   (CHARACTER 13)
                                                   (CHARACTER 13)))
                    (RETURN 'NoteFileTruncationFailed))
          else
```

(* * user aborted truncation.)

```
        (RETURN 'NoteFileTruncationAborted])
```

(NCLocalDevice.NewCardUID

```
[LAMBDA (Card)                                                    (* rht%: "15-May-87 19:00")
```

(* * The local single user device vector function, which installs the UID and IndexLoc into the Card Object.)

(* * kef 8/6/86%: Incorporated kirk's change%: Kirk 7/25/86%: Fixed overflow check to catch GEQ, gave warning message a window to show message, changed warning message and changed break call to ERROR!.)

(* * rht 1/22/87%: Now only calls NC.MakeUID if Card doesn't already have one.
Made small change to calculation of PercentUsed. Prevented NextIndexNum from being incremented in the case of full index.)

(* * rht 5/15/87%: Now calls NC.CheckForExpandIndex to handle cases when index is full or nearly full.)

```
  (LET ((NoteFile (fetch (Card NoteFile) of Card))
        NextIndexNum IndexNumsFreeList IndexNum PercentUsed NumUsed)
       (NC.CheckForExpandIndex NoteFile)
       (SETQ NextIndexNum (fetch (NoteFile NextIndexNum) of NoteFile))
       (SETQ IndexNumsFreeList (fetch (NoteFile IndexNumsFreeList) of NoteFile))
       (if IndexNumsFreeList
           then (SETQ IndexNum (pop IndexNumsFreeList))
                (replace (NoteFile IndexNumsFreeList) of NoteFile with IndexNumsFreeList)
         else (SETQ IndexNum NextIndexNum)
              (replace (NoteFile NextIndexNum) of NoteFile with (ADD1 NextIndexNum)))
       (replace (Card IndexLoc) of Card with (NC.NoteFileLocFromIndexNum IndexNum))
       (replace (Card IndexDirtyFlg) of Card with T)
       (OR (type? UID (fetch (Card UID) of Card))
           (replace (Card UID) of Card with (NC.MakeUID)))
       Card])
```

(NCLocalDevice.MarkCardDeleted

```
[LAMBDA (Card)                                                    (* Feuerman "28-Jul-86 16:57")
```

(* * This is the local MarkCardDeletedFn. Simply sets the status field.)

```
  (NC.SetStatus Card 'DELETED])
```

(NCLocalDevice.GetCardInfo

```
[LAMBDA (Card Aspects)                                            (* kirk%: "10-Sep-86 16:02")
```

(* * Returns an ALIST of all of the Aspects requested.)

(* * fgh |9/3/86| Added TITLE aspect to help spped up caching on server.)

(* * kirk |9/10/86| Deleted call on NC.FetchType testing before going to device because caused stack overflow in CacheTypesAndTitles)

```
  (for Aspect in [MKLIST (OR Aspects '(STATUS TYPE]
     collect (CONS Aspect (SELECTQ Aspect
                                   (TYPE (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                                             (LET [(Stream (fetch (NoteFile Stream) of (fetch (Card NoteFile)
                                                                                              of Card]
                                                   (SETFILEPTR Stream (fetch (Card MainLoc) of Card))
                                                   (NC.ReadCardPartHeader Card NC.ItemIdentifier Stream)
                                                   (NC.SetType Card (NC.ReadCardType Stream))
                                                   (NC.FetchType Card))))
                                   (STATUS (NC.FetchStatus Card))
                                   (TITLE (NC.RetrieveTitle Card))
                                   (SHOULDNT (CONCAT "Unknown card aspect:  " Aspect])
```

(NCLocalDevice.PutCardPart

```
[LAMBDA (Card CardPartName WhenFlg)                               (* rht%: "12-Sep-86 00:52")
```

(* * This is the local single user device put card part function.)

(* * The free variable PutSuccessfulLoc is bound in the calling function, either NC.PutTitle or NC.PutLinks, etc.
If NIL, then the Put wasn't successful, so report the error. If non-NIL, then it should be the position in the file where we put the card part, so now we can set the appropriate card part loc slot to be that number.)

(* * kef 7/28/86%: Added REGION possibility to CardPartName.)

(* * rht&fgh 9/12/86%: Now doesn't try to read links card parts if card is new in FROMLINKS/BEFORE case.)

```
  (DECLARE (GLOBALVARS NC.LinksIdentifier))
  (SELECTQ WhenFlg
       (BEFORE (SELECTQ CardPartName
                        (FROMLINKS
```

(* * In this case, let's grab the TOLINKS and GLOBALTOLINKS as they exist on the file still, and save them on the Card's UserProps, so that we can use them in the AFTER function to write them.)

```
                                (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                                    (LET (ToLinks GlobalLinks)
```

```
                                        [if (NOT (NC.FetchNewCardFlg Card))
                                            then (SETFILEPTR (fetch (NoteFile Stream) of (fetch (Card NoteFile)
                                                                                                of Card))
                                                             (fetch (Card LinksLoc) of Card))
                                                 (NC.ReadCardPartHeader Card NC.LinksIdentifier
                                                        (fetch (NoteFile Stream) of (fetch (Card NoteFile)
                                                                                       of Card)))
                                                 [SETQ ToLinks (NC.ReadListOfLinks (fetch (NoteFile Stream)
                                                                                      of (fetch (Card NoteFile)
                                                                                           of Card]
                                                 (NC.ReadListOfLinks (fetch (NoteFile Stream)
                                                                        of (fetch (Card NoteFile) of Card)))
                                                 (SETQ GlobalLinks (NC.ReadListOfLinks
                                                                    (fetch (NoteFile Stream)
                                                                       of (fetch (Card NoteFile) of Card]
                                             (NC.PutProp Card 'OldLinks (CONS ToLinks GlobalLinks)))
                                        (SETFILEPTR (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))
                                                    -1)))
                                ((TITLE SUBSTANCE LINKS PROPLIST)
                                    (SETFILEPTR (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))
                                                -1))
                                (REGION
```

(* In the region case, set the file pointer to the spot in the NoteFile just past the Card Type in the header information.)

```
                                        (SETFILEPTR (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))
                                                    (fetch (Card MainLoc) of Card))
                                        (NC.ReadCardPartHeader Card NC.ItemIdentifier (fetch (NoteFile Stream)
                                                                                         of (fetch (Card NoteFile)
                                                                                              of Card)))
                                        (NC.ReadCardType (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))))
                                (SHOULDNT (CONCAT "Bad card part name:  " CardPartName))))
        (AFTER [AND PutSuccessfulLoc (SELECTQ CardPartName
                                        (TITLE (NC.SetTitleLoc Card PutSuccessfulLoc))
                                        (SUBSTANCE (NC.SetMainLoc Card PutSuccessfulLoc))
                                        (LINKS (NC.SetLinksLoc Card PutSuccessfulLoc))
                                        (PROPLIST (NC.SetPropListLoc Card PutSuccessfulLoc))
                                        (FROMLINKS (NCLocalDevice.PutFromLinks Card PutSuccessfulLoc))
                                        (REGION NIL)
                                        (SHOULDNT (CONCAT "Bad card part name:  " CardPartName])
        (SHOULDNT (CONCAT "Don't understand WhenFlg argument:  " WhenFlg])
```

(NCLocalDevice.GetCardPart

```
  [LAMBDA (Card CardPartName WhenFlg)                    (* Feuerman "16-Jul-86 08:26")
```

(* * The local single user GetCardPartFn.)

```
    (COND
       ((EQ WhenFlg 'BEFORE)
        (SETFILEPTR (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))
               (SELECTQ CardPartName
                    (TITLE (fetch (Card TitleLoc) of Card))
                    (SUBSTANCE (fetch (Card MainLoc) of Card))
                    (TYPE (fetch (Card MainLoc) of Card))
                    (LINKS (fetch (Card LinksLoc) of Card))
                    (PROPLIST (fetch (Card PropListLoc) of Card))
                    (SHOULDNT (CONCAT "Unrecognized card part:  " CardPartName])
```

(NCLocalDevice.ObtainWritePermission

```
  [LAMBDA (Card CardPartName)                            (* Feuerman "16-Jul-86 13:45")
```

(* Should check here to make sure that this NoteFile is indeed a single user NoteFile, but we'll figure that out later. For now, just return T.)

```
    T])
```

(NCLocalDevice.ReleaseWritePermission

```
  [LAMBDA (Card CardPartName)                            (* Feuerman "16-Jul-86 13:45")
```

(* Should check here to make sure that this NoteFile is indeed a single user NoteFile, but we'll figure that out later. For now, just return T.)

```
    T])
```

(NCLocalDevice.CancelCacheSubscription

```
  [LAMBDA (Card CardPart)                                (* Feuerman "16-Jul-86 14:34")
```

(* Should check here to make sure that this NoteFile is indeed a single user NoteFile, but we'll figure that out later. For now, just return T.)

```
    T])
```

(NCLocalDevice.DeleteNoteFile

```
  [LAMBDA (FILENAME)                                                    (* rht%: "14-Nov-86 14:48")

          (* * The local device delete notefile. Contains a lot of the statements borrowed from the original delete notefile.)

          (* * rht & rg 11/14/86%: The print statement should be ripped out and perhaps moved to NC.DeleteDatabaseFile.
          This fn should return some sort of error code instead.)

    (COND
       ((OPENP FILENAME)
        (NC.PrintMsg NIL T FILENAME " is an open file." (CHARACTER 13)
               "Delete cancelled."
               (CHARACTER 13))
        NIL)
       (T (DELFILE FILENAME])

)

;; Install the device vector.

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.DeviceVectorsHashArray)
)

(ADDTOVAR DEFAULTFILETYPELIST (NOTEFILE . BINARY))

(DECLARE%: DONTEVAL@LOAD

(PUTHASH 'LOCALSINGLEUSER (create NoteFileDevice
                                  OpenNoteFileFn _ (FUNCTION NCLocalDevice.OpenNoteFile)
                                  CloseNoteFileFn _ (FUNCTION NCLocalDevice.CloseNoteFile)
                                  CreateNoteFileFn _ (FUNCTION NCLocalDevice.CreateNoteFile)
                                  DeleteNoteFileFn _ (FUNCTION NCLocalDevice.DeleteNoteFile)
                                  CompactNoteFileFn _ (FUNCTION NCLocalDevice.CompactNoteFile)
                                  RepairNoteFileFn _ (FUNCTION NCLocalDevice.InspectAndRepairNoteFile)
                                  BuildHashArrayFn _ (FUNCTION NCLocalDevice.BuildHashArray)
                                  NoteFileOpenPFn _ (FUNCTION NCLocalDevice.NoteFileOpenP)
                                  CheckpointNoteFileFn _ (FUNCTION NCLocalDevice.CheckpointNoteFile)
                                  ConvertNoteFileFormatFn _ (FUNCTION NCLocalDevice.ConvertNoteFileFormat)
                                  TruncateNoteFileFn _ (FUNCTION NCLocalDevice.TruncateNoteFile)
                                  NewCardUIDFn _ (FUNCTION NCLocalDevice.NewCardUID)
                                  MarkCardDeletedFn _ (FUNCTION NCLocalDevice.MarkCardDeleted)
                                  GetCardInfoFn _ (FUNCTION NCLocalDevice.GetCardInfo)
                                  PutCardPartFn _ (FUNCTION NCLocalDevice.PutCardPart)
                                  GetCardPartFn _ (FUNCTION NCLocalDevice.GetCardPart)
                                  ObtainWritePermissionFn _ (FUNCTION NCLocalDevice.ObtainWritePermission)
                                  ReleaseWritePermissionFn _ (FUNCTION NCLocalDevice.ReleaseWritePermission)
                                  CancelCacheSubscriptionFn _ (FUNCTION NCLocalDevice.CancelCacheSubscription))
       NC.DeviceVectorsHashArray)
)


;;; Version information for LOCALSINGLEUSER device.

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NCLocalDevice.NoteFileVersionsList NCLocalDevice.CurrentNoteFileVersionNumber)
)

(DEFINEQ
```

(NCLocalDevice.CurrentVersion

```
  [LAMBDA NIL                                                           (* fgh%: "23-May-86 15:12")

          (* * Return the NoteFileVersion object corresponding to the latest release of notecards.)

    (for NoteFileVersion in NCLocalDevice.NoteFileVersionsList when (EQ (fetch (NoteFileVersion Version)
                                                                           of NoteFileVersion)

                                                                        NCLocalDevice.CurrentNoteFileVersionNumber
                                                                        )
       do (RETURN NoteFileVersion])

)

(RPAQ? NCLocalDevice.CurrentNoteFileVersionNumber 3)

(RPAQ? NCLocalDevice.NoteFileVersionsList (LIST (create NoteFileVersion Version _ 3 NumberOfReservedCards _ 20
                                                        NoteFileIndexWidth _ 28 NoteFileHeaderSize _ 30)))


;;; Supporting functions and variables

(DEFINEQ
```

(NCLocalDevice.GetNoteFileHeader

```
  [LAMBDA (NoteFile)                                                    (* kirk%: "21-Dec-86 16:57")

          (* * Fill in the NoteFile object with stuff from the file.)

    (LET ((Stream (fetch (NoteFile Stream) of NoteFile)))               (* Recover the 30 information bytes for the notefile.)
         (SETFILEPTR Stream 0)                                          (* 3 bytes for next card ID)
         (replace (NoteFile NextIndexNum) of NoteFile with (NC.ReadPtr Stream 3))
                                                                        (* 3 bytes for index size)
         (replace (NoteFile HashArraySize) of NoteFile with (NC.ReadPtr Stream 3))
                                                                        (* One dummy byte so that version number stays in favorite old
        place.)
         (NC.ReadPtr Stream 1)                                          (* 1 byte for notecards version number)
         (replace (NoteFile Version) of NoteFile with (NC.ReadPtr Stream 1))
                                                                        (* 3 bytes for next link ID)
         (replace (NoteFile NextLinkNum) of NoteFile with (NC.ReadPtr Stream 3))
                                                                        (* 3 bytes for pointer to current checkpt ptr.)
         (replace (NoteFile CheckptPtr) of NoteFile with (NC.ReadPtr Stream 3))
                                                                        (* 14 bytes for NoteFile UID.)
         (replace (NoteFile UID) of NoteFile with (NC.ReadUID NoteFile))
                                                                        (* 2 bytes for future needs)
         (NC.ReadPtr Stream 2)
        NoteFile])
```

(NCLocalDevice.GetHashArray

```
  [LAMBDA (NoteFile QuietFlg InterestedWindow OperationMsg)             (* fgh%: " 1-Sep-86 17:50")

          (* * Read the hash array off of a local device notefile, building the hash array along the way.)

          (* * fgh |5/23/86| First created on the basis of the old NC.GetHashArray.)

          (* * fgh |9/1/86| Reimplemented QuietFlg InterestedWindow OperationMsg args.)

          (* * Get the hash array.)

    (if (NOT (OPENP (fetch (NoteFile Stream) of NoteFile)))
        then

          (* * NoteFile isn't open, Error.)

            'NoteFileNotOpen
      else

          (* * Grab the monitor lock and read the index.)

          (WITH.MONITOR (NC.FetchMonitor NoteFile)
              (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
                    (CardTotal (SUB1 (fetch (NoteFile NextIndexNum) of NoteFile)))
                    IndexNumsFreeList)

          (* * Set fileptr to beginning of the index.)

                   [SETFILEPTR Stream (CONSTANT (fetch (NoteFileVersion NoteFileHeaderSize) of (
                                                                                  NCLocalDevice.CurrentVersion
                                                                                              ]

          (* * Read in the index entries, creating a card for each and installing it in the hash array.)

                   [for IndexNum from 1 to CardTotal bind Card
                      eachtime (if (AND (NULL QuietFlg)
                                        (EQ (IMOD IndexNum 100)
                                            1))
                                   then (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                "Building index array."
                                                (CHARACTER 13)
                                                "Processing number " IndexNum " of " CardTotal "." (CHARACTER
                                                                                                    13)))
                             (BLOCK)
                      do (SETQ Card (NCLocalDevice.ReadIndexEntry NoteFile))
                         (if (EQ (fetch (Card Status) of Card)
                                 'FREE)
                             then (push IndexNumsFreeList IndexNum)
                           else (NC.InstallCardInNoteFile Card NoteFile))
                      finally (AND (NULL QuietFlg)
                                   (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                          "Building index array."
                                          (CHARACTER 13)
                                          "Done."
                                          (CHARACTER 13]

          (* * Set the free list)

                   (replace (NoteFile IndexNumsFreeList) of NoteFile with IndexNumsFreeList)

          (* * Return the NoteFile NoteFile, indicating everything is okay.)

                   NoteFile))])
```

(NCLocalDevice.GetSpecialCardUIDs

```
[LAMBDA (NoteFile)                                                    (* fgh%: "25-May-86 11:58")

        (* * Get the UIDs for the first 6 cards from NoteFile)

        (* * fgh |5/23/86| First created.)

  (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
        (CriticalUIDs (create NoteFileCriticalUIDs)))

        (* * FOR Start of Index by Index Width, read the UID for the first 5 index entries.
        UID starts at the second byte of the index entry, hence the ADD1.)

       (for Index from 1 to 5 as FilePtr from [CONSTANT (ADD1 (fetch (NoteFileVersion NoteFileHeaderSize)
                                                                 of (NCLocalDevice.CurrentVersion]
         by (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NCLocalDevice.CurrentVersion)))
         do (SETFILEPTR Stream FilePtr)
            (SELECTQ Index
                 (1 (replace (NoteFileCriticalUIDs TableOfContents) of CriticalUIDs with (NC.ReadUID Stream)))
                 (2 (replace (NoteFileCriticalUIDs Orphans) of CriticalUIDs with (NC.ReadUID Stream)))
                 (3 (replace (NoteFileCriticalUIDs ToBeFiled) of CriticalUIDs with (NC.ReadUID Stream)))
                 (4 (replace (NoteFileCriticalUIDs LinkLabels) of CriticalUIDs with (NC.ReadUID Stream)))
                 (5 (replace (NoteFileCriticalUIDs Registry) of CriticalUIDs with (NC.ReadUID Stream)))
                 (SHOULDN'T)))
      CriticalUIDs])
```

(NCLocalDevice.PutNoteFileHeader

```
[LAMBDA (NoteFile)                                                    (* fgh%: "26-May-86 23:13")

        (* * Write down to the notefile the header information extracted from the NoteFile object.)

        (* * fgh |5/26/86| Renamed from NC.PutNoteFileHeader)

  (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
        (FullFileName (fetch (NoteFile FullFileName) of NoteFile))
        (UID (fetch (NoteFile UID) of NoteFile))
        (NextIndexNum (fetch (NoteFile NextIndexNum) of NoteFile))
        (HashArraySize (fetch (NoteFile HashArraySize) of NoteFile))
        (NextLinkNum (fetch (NoteFile NextLinkNum) of NoteFile))
        (CheckptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
        (Version (fetch (NoteFile Version) of NoteFile)))
       (WITH.MONITOR (NC.FetchMonitor NoteFile)
           (if (OPENP Stream)
               then                                                   (* Fill in the 30 information bytes for the notefile.)
                    (SETFILEPTR Stream 0)                             (* 3 bytes for next card ID)
                    (NC.WritePtr Stream NextIndexNum 3)               (* 3 bytes for index size)
                    (NC.WritePtr Stream HashArraySize 3)              (* One dummy byte so that version number stays in favorite old
                   place.)
                    (NC.WritePtr Stream -1 1)                         (* 1 byte for notecards version number)
                    (NC.WritePtr Stream Version 1)                    (* 3 bytes for next link ID)
                    (NC.WritePtr Stream NextLinkNum 3)                (* 3 bytes for pointer to current checkpt ptr.)
                    (NC.WritePtr Stream CheckptPtr 3)                 (* 14 bytes for NoteFile UID.)
                    (NC.WriteUID NoteFile UID)                        (* 1 bytes for future needs)
                    (NC.WritePtr Stream -1 1)
                   NoteFile
             else (NC.ReportError NIL "NC.PutNoteFileHeader: Stream not open!!!")))])
```

(NCLocalDevice.PutHashArray

```
[LAMBDA (NoteFile InterestedWindow AllCardsFlg OperationMsg QuietFlg)
                                                                      (* rht%: "24-May-87 00:32")

        (* * Write down the hash array's contents to the notefile.)

        (* * kirk 27Nov85 Added AllCardsFlg for use by the compactor.)

        (* * fgh |5/26/86| Adapted from NC.PutHashArray with minor changes.)

        (* * fgh |9/1/86| Reimplemented QuietFlg.)

        (* * fgh |9/5/86| Put in check for HARRAYP of NoteFile's HashArray becuase MAPHASH of NIL will map hash down an
        arbitrary array.)

        (* * rht 3/13/87%: Fixed the "TPutting" print msg. Changed AllActiveCardsFlg to be AllCardsFlg.
        This allows cards with Deleted status to be written down with Free status.)

        (* * rht 5/24/87%: Now sets InterestedWindow if was passed in nil.)

  (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
  (if (HARRAYP (fetch (NoteFile HashArray) of NoteFile))
      then (LET ((CardTotal (SUB1 (fetch (NoteFile NextIndexNum) of NoteFile)))
                 (Num 0))
                (OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                  "Putting index array."
```

```
                                    (CHARACTER 13)
                                    "Processing item number " 1 " out of " CardTotal "." (CHARACTER 13)))
                (NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                                  [OR QuietFlg (PROGN (SETQ Num (ADD1 Num))
                                                                      (AND (ZEROP (IREMAINDER Num 100))
                                                                           (NC.PrintMsg InterestedWindow T
                                                                                  (OR OperationMsg "")
                                                                                  "Putting index array."
                                                                                  (CHARACTER 13)
                                                                                  "Processing item number "
                                                                                  Num " out of " CardTotal
                                                                                  "." (CHARACTER 13]
                                                        (* Turn deleted slots into free ones.)
                                                  (if (AND AllCardsFlg (NEQ (fetch (Card Status) of Card)
                                                                            'ACTIVE))
                                                      then (replace (Card Status) of Card with 'FREE))
                                                  (if (OR AllCardsFlg (fetch (Card IndexDirtyFlg) of Card))
                                                      then (NCLocalDevice.PutIndexEntry Card])
```

(NCLocalDevice.PutIndexEntry

```
[LAMBDA (Card)                                                  (* fgh%: "26-May-86 23:20")

          (* * Write down to the file the index entry for this card.)

          (* * fgh |5/26/86| Renamed from NC.PutIndexEntry.)

    (LET ((NoteFile (fetch (Card NoteFile) of Card)))
         (WITH.MONITOR (NC.FetchMonitor NoteFile)
             (LET ((Stream (fetch (NoteFile Stream) of NoteFile)))
                  (SETFILEPTR Stream (fetch (Card IndexLoc) of Card))
                  (NC.WriteStatus Stream (fetch (Card Status) of Card))
                  (NC.WriteUID NoteFile (fetch (Card UID) of Card))
                  (NC.WritePtr Stream (fetch (Card MainLoc) of Card)
                         3)
                  (NC.WritePtr Stream (fetch (Card LinksLoc) of Card)
                         3)
                  (NC.WritePtr Stream (fetch (Card TitleLoc) of Card)
                         3)
                  (NC.WritePtr Stream (fetch (Card PropListLoc) of Card)
                         3)
                  (replace (Card IndexDirtyFlg) of Card with NIL)))])
```

(NCLocalDevice.NoteFileVersionOkayP

```
[LAMBDA (NoteFile)                                              (* fgh%: "23-May-86 15:46")

          (* * Is this a current version NF?)

    (EQP (fetch (NoteFile Version) of NoteFile)
         NCLocalDevice.CurrentNoteFileVersionNumber])
```

(NCLocalDevice.CheckForPlausibleNoteFileHeader

```
[LAMBDA (NoteFile EOFPtr)                                       (* rht%: " 8-Jun-87 14:48")

          (* * Return non-nil if notefile's header information seems reasonable.)

          (* * Changed to allow Checkpt = EOFPtr)

          (* * fgh |5/23/86| Changed to conform to Local Device NoteFile needs.)

    (LET [(NextIndexNum (fetch (NoteFile NextIndexNum) of NoteFile))
          (HashArraySize (fetch (NoteFile HashArraySize) of NoteFile))
          (CheckptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
          (NextLinkNum (fetch (NoteFile NextLinkNum) of NoteFile))
          (IndexWidth (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NCLocalDevice.CurrentVersion]
         (COND
            ((OR (LEQ NextIndexNum 0)
                 (GEQ (TIMES IndexWidth NextIndexNum)
                      EOFPtr))
             'BadNextIndexNum)
            ((OR (LEQ HashArraySize 0)
                 (GEQ (TIMES IndexWidth HashArraySize)
                      EOFPtr))
             'BadHashArraySize)
            ((OR (LEQ CheckptPtr 0)
                 (IGREATERP CheckptPtr EOFPtr))
             'BadCheckptPtr)
            ((LESSP NextLinkNum 0)
             'BadNextLinkNum)
            (T NoteFile])
```

(NCLocalDevice.NoteFileNeedsTruncationP

```
[LAMBDA (NoteFile EofPtr)                                       (* fgh%: "23-May-86 15:44")
```

(* * Is the notefile bigger than its checkpoint pointer thinks?)

```
(LESSP (fetch (NoteFile CheckptPtr) of NoteFile)
       EofPtr])
```

(NCLocalDevice.OpenFailed

```
[LAMBDA (NoteFile ErrorType)                                    (* fgh%: " 1-Sep-86 22:35")
```

(* * Open failed, clean up and return an error)

(* * fgh |5/2/386| First created.)

```
(AND (STREAMP (fetch (NoteFile Stream) of NoteFile))
     (CLOSEF? (fetch (NoteFile Stream) of NoteFile)))
(replace (NoteFile Stream) of NoteFile with NIL)
(NC.SetMonitor NoteFile NIL)
(OR ErrorType 'NoteFileOpenFailed])
```

(NCLocalDevice.ReadIndexEntry

```
[LAMBDA (NoteFile)                                              (* fgh%: "25-May-86 18:36")
```

(* * Get a card from current position of file and install it into notefile.)

(* * fgh |5/23/86| First created on the basis of the old NC.ReadIndexEntry)

```
(LET ((Stream (fetch (NoteFile Stream) of NoteFile)))
     (create Card
             IndexLoc _ (GETFILEPTR Stream)
             Status _ (NC.ReadStatus Stream)
             UID _ (NC.ReadUID NoteFile)
             MainLoc _ (NC.ReadPtr Stream 3)
             LinksLoc _ (NC.ReadPtr Stream 3)
             TitleLoc _ (NC.ReadPtr Stream 3)
             PropListLoc _ (NC.ReadPtr Stream 3)
             IndexDirtyFlg _ NIL
             NoteFile _ NoteFile])
```

(NCLocalDevice.SaveInformationPastCheckpoint

```
[LAMBDA (NoteFile PromptWindow)                                 (* rht%: "19-Feb-87 17:39")
```

(* * Save the information on NoteFile that is past the checkpoint pointer.
Assume NoteFile is closed.)

(* * fgh |5/25/86| First created.)

(* * rht 2/19/87%: Now properly binds FullFileName.)

```
(PROG ((FullFileName (fetch (NoteFile FullFileName) of NoteFile))
       NoteFileStream SaveStream SaveFile)
```

(* * Open the NoteFile)

```
(if [NULL (CAR (ERSETQ (SETQ NoteFileStream (OPENSTREAM FullFileName 'INPUT 'OLD '((TYPE BINARY]
    then (RETURN 'CouldNotOpenNoteFile))
```

(* * Open the save file after asking the user to specify it.)

```
(SETQ SaveFile (NC.AskUser (CONCAT (CHARACTER 13)
                                   "File to save info in: ")
                           NIL NIL NIL PromptWindow T))
(if [NULL (CAR (ERSETQ (SETQ SaveStream (AND SaveFile (OPENSTREAM SaveFile 'OUTPUT NIL
                                                                  '((TYPE BINARY]
    then (RETURN 'CouldNotOpenSaveFile))
```

(* * Copy the bytes from the NoteFile to the save file.)

```
(if [NULL (ERSETQ (PROGN (NC.PrintMsg PromptWindow T "Saving extra info to " SaveFile " ...")
                         (COPYBYTES NoteFileStream SaveStream (fetch (NoteFile CheckptPtr) of NoteFile))
                         (CLOSEF SaveStream)
                         (CLOSEF NoteFileStream)
                         (NC.PrintMsg PromptWindow NIL "Done." (CHARACTER 13]
    then (CLOSEF? SaveStream)
         (DELFILE SaveStream)
         (CLOSEF? NoteFileStream)
         (RETURN 'CopyToSaveFileFailed)
  else (RETURN NoteFile])
```

(NCLocalDevice.PutFromLinks

```
[LAMBDA (Card StreamLoc)                                        (* rht%: " 4-Nov-86 19:32")
```

(* * Assuming that Card has on it UserProps the old TOLINKS and GLOBALLINKS, this function rewrites over the end of the NoteFile stream starting at StreamLoc the Links Info merged with the new FromLinks, which are already on the Stream.)

(* * rht&rg 11/4/86%: Now sets length field properly after writing down the links.)

```
    (DECLARE (GLOBALVARS NC.LinksIdentifier))
    (LET ((STREAM (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card)))
          FromLinks EndLoc DataLoc)
         (SETFILEPTR STREAM StreamLoc)
         (NC.ReadCardPartHeader Card NC.LinksIdentifier STREAM)
         (SETQ DataLoc (GETFILEPTR STREAM))
         (SETQ FromLinks (NC.ReadListOfLinks STREAM))
         (SETFILEPTR STREAM DataLoc)
         [NC.WriteListOfLinks STREAM (CAR (NC.GetProp Card 'OldLinks]
         (NC.WriteListOfLinks STREAM FromLinks)
         [NC.WriteListOfLinks STREAM (CDR (NC.GetProp Card 'OldLinks]
         (SETQ EndLoc (GETFILEPTR STREAM))
         (SETFILEPTR STREAM StreamLoc)
         (NC.WritePtr STREAM (DIFFERENCE EndLoc StreamLoc)
                3)
         (SETFILEPTR STREAM EndLoc)
         (NC.SetLinksLoc Card StreamLoc])
```

(NCLocalDevice.TotalNoteFileIndexSize

```
  [LAMBDA (HashArraySize)                                    (* fgh%: " 1-Sep-86 17:36")
```

(* * Return the length of the index part of the notefile including header.)

```
    (PLUS (CONSTANT (fetch (NoteFileVersion NoteFileHeaderSize) of (NCLocalDevice.CurrentVersion)))
          (TIMES (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NCLocalDevice.CurrentVersion)))
                 HashArraySize])

)

(DECLARE%: DONTEVAL@LOAD

(NC.StoreAutoloadFnFile (FUNCTION NC.ScavengerPhase1)
       'NCREPAIR
       'NOTECARDSDIRECTORIES)

(NC.StoreAutoloadFnFile (FUNCTION NC.CompactNoteFileInPlace)
       'NCCOMPACT
       'NOTECARDSDIRECTORIES)

(NC.StoreAutoloadFnFile (FUNCTION NC.CompactNoteFileToTarget)
       'NCCOMPACT
       'NOTECARDSDIRECTORIES)
)

(PUTPROPS NCLOCALDEVICE FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS NCLOCALDEVICE MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(PUTPROPS NCLOCALDEVICE COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1989 1990 1993 1994 2020))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX