pDOSD

COLLABORATORS			
	TITLE : pDOSD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 29, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

pDOSD

1.1 pDOSD.doc

pdos.library

```
DosAllgemeines()              pOS_AddDosAssign()
pOS_AddDosDefDevice()         pOS_AddDosDevice()
pOS_AddDosDeviceBuffers()     pOS_AddPart()
pOS_AddSegment()              pOS_AllocDosObject()
pOS_AllocDosProcessNum()      pOS_ChangeDosFileAccess()
pOS_ChangeDosObjectAccess()   pOS_CloneProcessPath()
pOS_CloneVars()               pOS_CloseDosDevice()
pOS_CloseDOSVolume()          pOS_CloseFile()
pOS_CloseSegment()            pOS_CompareDates()
pOS_ConstructDosFH()          pOS_ConstructDosLk()
pOS_ConstructDosMsgNotify()   pOS_ConstructDosSigNotify()
pOS_ConstructParse()          pOS_ConstructPattern()
pOS_ConstructProcess()        pOS_ConstructShell()
pOS_ConstructShellScript()    pOS_CopyObjectName()
pOS_CreateDirectory()         pOS_CreateDosArgs()
pOS_CreateDosAssign()         pOS_CreateDosDevFromMount()
pOS_CreateDosMount()          pOS_CreateDOSTemplate()
pOS_CreateDosTokenList()      pOS_CreateDosTokenString()
pOS_CreateDOSVolume()         pOS_CreateParse()
pOS_CreatePath()              pOS_CreateProcessA()
pOS_DateToStr()               pOS_DeleteDosArgs()
pOS_DeleteDosAssign()         pOS_DeleteDosMount()
pOS_DeleteObjectLk()          pOS_DeleteObjectName()
pOS_DeleteParse()             pOS_DeleteProcess()
pOS_DeleteVar()               pOS_DestructDosFH()
pOS_DestructDosLk()           pOS_DestructDosNotify()
pOS_DestructParse()           pOS_DestructPattern()
pOS_DestructProcess()         pOS_DestructShell()
pOS_DestructShellScript()     pOS_DosCheckSignal()
pOS_DosDelay()                pOS_DosEndNotify()
pOS_DosErrorReportA()         pOS_DosFindToken()
pOS_DosGetAliasCom()          pOS_DosGetResString1()
pOS_DosStartNotify()          pOS_DupFileHandle()
pOS_DupObjectLock()           pOS_ExamineFH()
pOS_ExamineObject()           pOS_ExamineObjectName()
pOS_ExitProcess()             pOS_ExNextObject()
```

pOS_FileEOF()
pOS_FileFlush()
pOS_FileFRead()
pOS_FileFWrite()
pOS_FileGetC()
pOS_FileGets()
pOS_FilePart()
pOS_FilePutC()
pOS_FilePuts()
pOS_FileUnGetC()
pOS_FindVar()
pOS_FreeDosObject()
pOS_FreeDosProcessNum()
pOS_GetBootLock()
pOS_GetCurrentDirName()
pOS_GetDateStamp()
pOS_GetDosDevice()
pOS_GetDosDeviceName()
pOS_GetDosErrText()
pOS_GetDosHandler()
pOS_GetDosInfoData()
pOS_GetDosInfoDataName()
pOS_GetDosMountName()
pOS_GetIoErr()
pOS_GetNextDosDevice()
pOS_GetProgDir()
pOS_GetPrompt()
pOS_GetStdErrOutput()
pOS_GetStdInput()
pOS_GetStdOutput()
pOS_GetVar()
pOS_InhibitDosDevice()
pOS_InitDosDevice()
pOS_InitDosIOReq()
pOS_InternalLoadSegment()
pOS_InternalUnloadSegment()
pOS_InterpretDosTokenList()
pOS_IsFileInteractive()
pOS_IsFileSystem()
pOS_IsFileSystemName()
pOS_IsInfinite()
pOS_LoadSegmentA()
pOS_LockDosDevList()
pOS_LockObject()
pOS_LockProcessList()
pOS_MatchPattern()
pOS_MountDosDevice()
pOS_MoveObjectName()
pOS_NameFromFH()
pOS_NameFromObjectLock()
pOS_OpenDosDevice()
pOS_OpenDOSVolume()
pOS_OpenFile()
pOS_OpenFileFromLock()
pOS_OpenSegment()
pOS_ParentObjectDir()
pOS_ParsePatternA()
pOS_Parsing()
pOS_PathMatchEnd()
pOS_PathMatchFirst()
pOS_PathMatchNext()
pOS_PathPart()
pOS_PrintDosArgList()
pOS_PrintDosErr()
pOS_ReadDosArgsA()
pOS_ReadFile()
pOS_RelabelDosDevice()
pOS_RemDosDefDevice()
pOS_RemDosDevice()
pOS_RemSegment()
pOS_RenameObject()
pOS_RenameObjectName()
pOS_RunCommand()
pOS_SameDosDevice()
pOS_SameDosObject()
pOS_SeekFile()
pOS_SetConsoleFH()
pOS_SetCurrentDirLock()
pOS_SetCurrentDirName()
pOS_SetDosScreenMode()
pOS_SetFileDate()
pOS_SetFileHandleBuf()
pOS_SetFileSize()
pOS_SetIoErr()
pOS_SetObjectComment()
pOS_SetObjectCommentName()
pOS_SetObjectDate()
pOS_SetObjectDateName()
pOS_SetObjectProtection()
pOS_SetObjectProtectionName()
pOS_SetObjectSize()
pOS_SetObjectSizeName()
pOS_SetProgDir()
pOS_SetPrompt()
pOS_SetShellFail()
pOS_SetVar()
pOS_SetVarLoc()
pOS_SplitName()
pOS_StrToDate()
pOS_SystemA()
pOS_UnloadSegment()
pOS_UnlockDosDevList()
pOS_UnlockObject()
pOS_UnlockProcessList()
pOS_VFWritef()
pOS_WaitForChar()
pOS_WriteDosArgsA()
pOS_WriteFile()

1.2 pdos.library/DosAllgemeines()

```
STRUKTUREN

        struct pOS_AnchorPath
        struct pOS_AnchorPathObj
        struct pOS_AssignNode
        struct pOS_DateStamp
        struct pOS_DateTime
        struct pOS_DosArgs
        struct pOS_DosBase
        struct pOS_DosDefDevice
        struct pOS_DosDevice
        struct pOS_DosDevPathInfo
        struct pOS_DosInfoData
        struct pOS_DosIOReq
        struct pOS_DosMountDevice
        struct pOS_DosNotifyMessage
        struct pOS_DosNotifyReq
        struct pOS_DosResidentLibInit
        struct pOS_DosScriptList
        struct pOS_DosScriptNode
        struct pOS_DosToken
        struct pOS_FileHandle
        struct pOS_FileInfoBlock
        struct pOS_FileLock
        struct pOS_FHManage
        struct pOS_GetDateStamp
        struct pOS_HunkStcHeader
        struct pOS_ISegmentData
        struct pOS_LocalVar
        struct pOS_MemPool
        struct pOS_MsgPort
        struct pOS_Parse
        struct pOS_ParseObject
        struct pOS_PatternObj
        struct pOS_PatternMatching
        struct pOS_Process
        struct pOS_Segment
        struct pOS_SegmentLst
        struct pOS_Shell
        struct pOS_ShellScript
        struct pOS_StdDosDevFunction
        struct pOS_TagItem
        struct pOS_TimeVal
        struct pOS_PatternMatching
        struct pOS_PatternObj

ENUMERATION

        enum pOS_AnchorPathFlags
        enum pOS_ArgTags
        enum pOS_DateTimeFlags
        enum pOS_DateTimeFormat
```

```
        enum pOS_DosArgsFlags
        enum pOS_DosBaseFlags
        enum pOS_DosDeviceTypes
        enum pOS_DosErrors
        enum pOS_DosFails
        enum pOS_DosInfoDataFlags
        enum pOS_DosInfoDataState
        enum pOS_DosInfoDataTyp
        enum pOS_DosIOReqCommands
        enum pOS_DosMountDevFlags
        enum pOS_DosMountDevType
        enum pOS_DosNotifyMsgClass
        enum pOS_DosNotifyMsgCode
        enum pOS_DosNotifyReqFlags
        enum pOS_DosObjects
        enum pOS_DosReportTags
        enum pOS_DosSignals
        enum pOS_DosTags
        enum pOS_DosTokenKind
        enum pOS_FileInfoEntryType
        enum pOS_FileHandleBufFlags
        enum pOS_FileHandleBufType
        enum pOS_FileHandleFlags
        enum pOS_FileHandleMode
        enum pOS_FileHandleSeek
        enum pOS_FileLockAccess
        enum pOS_FileLockFlags
        enum pOS_FileProtectionBits
        enum pOS_HunkTypes
        enum pOS_LocalVarFlags
        enum pOS_LocalVarType
        enum pOS_ProcessFlags
        enum pOS_SameFileLockType
        enum pOS_SegmLstFlags

NAMENLOSE ENUMERATION

        pOS_DosFileName_MAX
        pOS_DosPathName_MAX
        pOS_DosComment_MAX

DATENTYPEN

        dossize_t;
        dos2size_t;
        dospos_t;
        dosname_t;

DEFINES

        pDOS_TICKS_PER_SECOND
        pOS_DATETIMESTR_MAX
        pOS_DOSERR

INCLUDES

        pDOS/ArgTags.h
```

```
pDOS/Date.h
pDOS/DateTime.h
pDOS/DosArgs.h
pDOS/DosBase.h
pDOS/DosDev.h
pDOS/DosErrors.h
pDOS/DosSig.h
pDOS/DosTags.h
pDOS/DosTypes.h
pDOS/FIB.h
pDOS/Files.h
pDOS/Hunk.h
pDOS/InfoData.h
pDOS/Lock.h
pDOS/Macros.h
pDOS/Notify.h
pDOS/Parsing.h
pDOS/Pattern.h
pDOS/Process.h
pDOS/Report.h
pDOS/ScanDir.h
pDOS/Segment.h
pDOS/Var.h
proto/pDOS.h
```

1.3 pdos.library/pOS_AllocDosObject()

```
PROTOTYP
      __ARID__ APTR dosobj = pOS_AllocDosObject
        (
          pOS_DosBase *dosbase,
          ULONG typ,
          ULONG reserved
        );

FUNKTION
      Reservieren einer Dos-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      typ (_R_D0)
        Typ des anzulegenden Objekts (enum pOS_DosObjects)
        DOSOBJ_FIB     : struct pOS_FileInfoData,
                         pOS_ExamineObject(), pOS_ExNextObject()
        DOSOBJ_Shell   : struct pOS_Shell
        DOSOBJ_InfoData: struct pOS_DosInfoData
                         pOS_GetDosInfoData()
      reserved (_R_D1)
        Reserviert für zukünftige Erweiterungen, muß auf 0 gesetzt werden

ERGEBNIS
      dosobj (_R_D0)
        Zeiger auf angelegtes DosObject oder NULL im Fehlerfall
```

```
        Die Struktur muß mit pOS_FreeDosObject() freigegeben werden.

SIEHE AUCH
      pOS_FreeDosObject()

AMIGA FUNKTION
      void *AllocDosObject(ULONG typ,struct TagItem *tag)
```

1.4 pdos.library/pOS_FreeDosObject()

```
PROTOTYP
      VOID pOS_FreeDosObject
        (
          pOS_DosBase *dosbase,
          ULONG typ,
          __ARID__ APTR dosobj
        );

FUNKTION
      Freigeben einer Dos-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      typ (_R_D0)
        Typ des freizugebenden Objekts
      dosobj (_R_A0)
        Zeiger auf ein von pOS_AllocDosObject() erzeugte Struktur,
        die freigegeben werden soll.
        Achtung: auf dosobj darf nach diesem Aufruf nicht mehr
                 zugegriffen werden.

SIEHE AUCH
      pOS_AllocDosObject()

AMIGA FUNKTION
      void FreeDosObject(ULONG typ,APTR dosobj)
```

1.5 pdos.library/pOS_AllocDosProcessNum()

```
PROTOTYP
      ULONG res = pOS_AllocDosProcessNum
        (
          pOS_DosBase *dosbase,
          ULONG num
        );

FUNKTION
      Reservieren einer Process-Nummer.
```

```
PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      num (_R_D0)
        gewünschte Nummer oder
        -1 für nächste freie Nummer

ERGEBNIS
      res (_R_D0)
        resererviertе Nummer oder (-1) im Fehlerfall

SIEHE AUCH
      pOS_FreeDosProcessNum()

AMIGA FUNKTION
      ähnlich ULONG MaxCli();
```

1.6 pdos.library/pOS_FreeDosProcessNum()

```
PROTOTYP
      VOID pOS_FreeDosProcessNum
        (
          pOS_DosBase *dosbase,
          ULONG num
        );

FUNKTION
      Freigeben einer reservierten Process-Nummer.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      num (_R_D0)
        Nummer, die mit pOS_AllocDosProcessNum() reserviert wurde.

SIEHE AUCH
      pOS_AllocDosProcessNum()

AMIGA FUNKTION
```

1.7 pdos.library/pOS_LockProcessList()

```
PROTOTYP
      VOID pOS_LockProcessList
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Sperren der Process-Liste gegen Zugriffe anderer Tasks.
```

```
PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

SIEHE AUCH
      pOS_UnlockProcessList()

AMIGA FUNKTION
      struct DosList *LockDosList(ULONG)
```

1.8 pdos.library/pOS_UnlockProcessList()

```
PROTOTYP
      VOID pOS_UnlockProcessList
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Aufheben der Process-List-Sperre.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

SIEHE AUCH
      pOS_LockProcessList()

AMIGA FUNKTION
      void UnLockDosList(ULONG)
```

1.9 pdos.library/pOS_CreateDosMount()

```
PROTOTYP
      pOS_DosMountDevice *dosdev = pOS_CreateDosMount
        (
          pOS_DosBase *dosbase,
          ULONG typ
        );

FUNKTION
      Erzeugen einer neuen Gerätebeschreibung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      typ (_R_D0) (enum pOS_DosMountDevType)
        DMDTYP_Pico : Minimal-Device
        DMDTYP_BOD  : Blockorientiertes Device
```

```
        DMDTYP_Net  : Netzwerk-Device

ERGEBNIS
      res (_R_D0)
        Erzeugte Datenstruktur oder NULL im Fehlerfall.

SIEHE AUCH
      pOS_DeleteDosMount(), pOS_CreateDosDevFromMount()

AMIGA FUNKTION
```

1.10 pdos.library/pOS_DeleteDosMount()

```
PROTOTYP
      VOID pOS_DeleteDosMount
        (
          pOS_DosBase *dosbase,
          pOS_DosMountDevice *dosdev
        );

FUNKTION
      Freigeben einer Gerätebeschreibung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdev (_R_A0)
        Zeiger auf die Gerätestruktur, die freigegeben werden soll.

SIEHE AUCH
      pOS_CreateDosMount()

AMIGA FUNKTION
```

1.11 pdos.library/pOS_LockDosDevList()

```
PROTOTYP
      VOID pOS_LockDosDevList
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Sperren der Dos-Device-Liste gegen Zugriffe anderer Tasks.
      Danach darf auf die Strukturen dos_Device, dos_DevMount,
      dos_DefDev und dos_FHManage der pOS_DosBase zugegriffen werden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
```

```
HINWEIS
      Das Programm steht solange, bis die Semaphore gelockt werden
      konnte. Die Sperre soll deshalb so kurz wie möglich bestehen,
      damit andere Processe, die ebenfalls auf die Daten zugreifen
      wollen, nicht zu lange warten müssen.

SIEHE AUCH
      pOS_UnlockDosDevList()

AMIGA FUNKTION
      ähnlich void UnLockDosList(ULONG)
```

1.12 pdos.library/pOS_UnlockDosDevList()

```
PROTOTYP
      VOID pOS_UnlockDosDevList
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Aufheben einer Zugriffssperre auf die Dos-Device-Liste.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

SIEHE AUCH
      pOS_LockDosDevList()

AMIGA FUNKTION
      ähnlich void UnLockDosList(ULONG)
```

1.13 pdos.library/pOS_GetBootLock()

```
PROTOTYP
      pOS_FileLock *res = pOS_GetBootLock
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Ermittelt den Lock des Boot-Laufwerks ("SYS:").

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
```

```
      res (_R_D0)
        pOS_FileLock-Zeiger auf das Boot-Laufwerk.

HINWEIS
      Kann auch direkt über pOS_DosBase->dos_BootLock
      ausgelesen werden.

SIEHE AUCH
      pOS_LockDosDevList()

AMIGA FUNKTION
```

1.14 pdos.library/pOS_OpenFile()

```
PROTOTYP
      __ARID__ pOS_FileHandle *res = pOS_OpenFile
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *filename,
          ULONG mode
        );

FUNKTION
      Öffnen einer Datei für Schreib/Leseaktionen.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        wird ein filelock angegeben, kann der filename relativ
        dazu angegeben werden.
        bei NULL wird pr_CurrentDir verwendet
      filename (_R_A1)
        Name der Datei, die geöffnet werden soll
      mode (_R_D0) (enum pOS_FileHandleMode)
        FILEHDMOD_Read      : zum Lesen
        FILEHDMOD_Write     : zum Schreiben
                              löscht eine bestehende Datei
        FILEHDMOD_ReadWrite : zum Lesen und Schreiben
        FILEHDMOD_Append    : zum Schreiben ab Dateiende
                              erzeugt die Datei wenn sie noch nicht existiert
        Zusätzlich kann noch angegeben werden
        FILEHDMOD_GBuffer   : Schreib/Lese-Pufferung
        FILEHDMOD_DobBuf    : Doppelte Pufferung
        FILEHDMOD_Unique    : MultiAssigns werden NICHT berücksichtigt
        FILEHDMOD_NoReq     : Es wird im Fehlerfall kein Report-Requester ↩
           geöffnet

ERGEBNIS
      res (_R_D0)
        Zeiger auf FileHandle der geöffneten Datei die mit
        pOS_CloseFile() wieder geschlossen werden muß.
        NULL wenn die Datei nicht geöffnet werden konnte.
```

```
        Dann kann über pOS_GetIoErr() die Fehlerursache abgefragt werden.

HINWEIS
      Wird die Datei zum Lesen geöffnet, kann die Dateigröße direkt
      aus filehandle->fh_Size gelesen werden.

SIEHE AUCH
      pOS_CloseFile(), pOS_SetFileHandleBuf()

AMIGA FUNKTION
      BPTR Open(STRPTR filename, LONG mode);
```

1.15 pdos.library/pOS_CloseFile()

```
PROTOTYP
      VOID pOS_CloseFile
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileHandle *filehandle
        );

FUNKTION
      Datei schließen.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle der Datei, die geschlossen werden soll.
        ACHTUNG: danach darf auf filehandle nicht mehr zugegriffen
                 werden.

SIEHE AUCH
      pOS_OpenFile()

AMIGA FUNKTION
      void Close();
```

1.16 pdos.library/pOS_ReadFile()

```
PROTOTYP
      dossize_t res = pOS_ReadFile
        (
           pOS_DosBase *dosbase,
           pOS_FileHandle *filehandle,
           APTR buffer,
           dossize_t size
        );

FUNKTION
```

```
      Lesen von Daten.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die geöffnete Datei
      buffer (_R_A1)
        Speicherbereich, der die gelesenen Daten aufnehmen soll
      size (_R_D0)
        Größe von buffer

ERGEBNIS
      res (_R_D0)
        Anzahl gelesener Zeichen, 0 am Dateiende

SIEHE AUCH
      pOS_OpenFile(), pOS_WriteFile(), pOS_SeekFile

AMIGA FUNKTION
      LONG Read(BPTR fh, VOID *buffer, LONG size);
```

1.17 pdos.library/pOS_WriteFile()

```
PROTOTYP
      dossize_t res = pOS_WriteFile
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const APTR buffer,
          dossize_t size
        );

FUNKTION
      Schreiben von Daten.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die geöffnete Datei
      buffer (_R_A1)
        Daten die geschrieben werden sollen
      size (_R_D0)
        Anzahl der Zeichen die geschrieben werden sollen

ERGEBNIS
      res (_R_D0)
        Anzahl der geschriebenen Zeichen, in der Regel size.
        Im Fehlerfall weniger, dann kann mit pOS_GetIoErr() die
        Fehlerursache erfragt werden.

SIEHE AUCH
      pOS_OpenFile(), pOS_ReadFile(), pOS_SeekFile
```

```
AMIGA FUNKTION
      LONG Write(BPTR fh, VOID *buffer, LONG size);
```

1.18 pdos.library/pOS_SeekFile()

```
PROTOTYP
      dossize_t res = pOS_SeekFile
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          dospos_t distanz,
          SLONG mode
        );

FUNKTION
      Schreib/Leseposition der Datei ändern.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die geöffnete Datei
      distanz (_R_D0)
        neue Position abhängig von mode
      mode (_R_D1) (enum pOS_FileHandleSeek)
        FILEHDSEK_Begin   : ab Dateianfang
        FILEHDSEK_Current : ab aktueller Position
        FILEHDSEK_End     : ab Dateiende

ERGEBNIS
      res (_R_D0)
        alte Position in der Datei
        pOS_DOSERR im Fehlerfall, dann kann mit pOS_GetIoErr()
        die Fehlerursache erfragt werden.

BEISPIEL
      /* von beliebieger Position an das Ende der Datei positionieren */
      pOS_SeekFile(...,...,0,FILEHDSEK_End)
      /* jetzt ohne eine Änderung, nur die Position erfragen */
      pOS_SeekFile(...,...,0,FILEHDSEK_End) => liefert Dateigröße

      pOS_SeekFile(...,...,-10,FILEHDSEK_Current)
      => die Position Richtung Dateianfang ändern
      pOS_SeekFile(...,...,10,FILEHDSEK_Current)
      => die Position Richtung Dateiende ändern

SIEHE AUCH
      pOS_OpenFile(), pOS_ReadFile(), pOS_WriteFile()

AMIGA FUNKTION
      LONG Seek(BPTR fh, LONG distanz, LONG mode);
```

1.19 pdos.library/pOS_DupFileHandle()

```
PROTOTYP
      __ARID__ pOS_FileHandle *res = pOS_DupFileHandle
        (
          pOS_DosBase *dosbase,
          const pOS_FileHandle *filehandle
        );

FUNKTION
      Duplizieren eines FileHandels

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle der dupliziert werden soll.

ERGEBNIS
      res (_R_D0)
        Duplizierter FileHandle der mit pOS_CloseFile() wieder
        geschlossen werden muß.
        NULL im Fehlerfall. Dann gibt pOS_GetIoErr() die Fehlerursache zurück.

HINWEIS
      Beide FileHandle sind weiterhin identisch. Ihre Positionen stimmen
      immer überein, egal mit welchen eine Aktion durchgeführt wird.

SIEHE AUCH
      pOS_DupObjectLock(), pOS_CloseFile

AMIGA FUNKTION
      ähnlich BPTR DupLockFromFH(BPTR filehandle);
```

1.20 pdos.library/pOS_IsFileInteractive()

```
PROTOTYP
      BOOL res = pOS_IsFileInteractive
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle
        );

FUNKTION
      Ermitteln, ob sich bei der Datei um eine interaktive Datei
      (z.B. CON: oder RAW:) handelt.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die zu prüfende Datei
```

```
ERGEBNIS
      res (_R_D0)
        FALSE wenn keine interaktive Datei vorliegt,
        sonst TRUE.

SIEHE AUCH

AMIGA FUNKTION
      LONG IsInteractive(BPTR filehandle);
```

1.21 pdos.library/pOS_WaitForChar()

```
PROTOTYP
      dossize_t res = pOS_WaitForChar
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const pOS_TimeVal *timeval
        );

FUNKTION
      Ermitteln, ob Daten gelesen werden können.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      timeval (_R_A1)
        maximale Wartezeit

ERGEBNIS
      res (_R_D0)
        Anzahl der Zeichen im Lesepuffer,
        0 für keine Zeichen

SIEHE AUCH
      pOS_ReadFile(), pOS_FGetC()

AMIGA FUNKTION
      BOOL WaitForChar(BPTR fh, LONG timeout);
```

1.22 pdos.library/pOS_SetFileHandleBuf()

```
PROTOTYP
      BOOL res = pOS_SetFileHandleBuf
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
```

```
          UBYTE *buffer,
          ULONG mode,
          dossize_t size
        );

FUNKTION
      Ändern des Puffermodus und der Puffergröße.
      Der Aufruf sollte direkt hinter pOS_OpenFile() erfolgen.

PARAMETER
          dosbase (_R_LB)
            Zeiger auf pOS_DosLibrary
          filehandle (_R_A0)
            FileHandle auf die Datei
          buffer (_R_A1)
            Zeiger auf den Pufferbereich.
          mode (_R_D0) (enum pOS_FileHandleBufType)
            Puffermodus
            FILEHDBUFTYP_None : keine Pufferung
      ~     FILEHDBUFTYP_Std  : volle Pufferung
      ~     FILEHDBUFTYP_LF   : zeilenweise Pufferung
          size (_R_D1)
            Größe des Pufferbereichs

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neue Puffergröße verwendet wird,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_FileGetC(), pOS_FileGets(), pOS_FilePutC(), pOS_FilePuts(),
      pOS_FileFRead(), pOS_FileFWrite(), pOS_FileFlush()

AMIGA FUNKTION
      LONG SetVBuf(BPTR fh, STRPTR, LONG, LONG);
```

1.23 pdos.library/pOS_FileFlush()

```
PROTOTYP
      BOOL res = pOS_FileFlush
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle
        );

FUNKTION
      Ausgabestrom: schreiben aller Daten
      Eingabestrom: löschen des Puffers, sodaß keine Daten mehr ausstehen

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
```

```
ERGEBNIS
      res (_R_D0)
        TRUE für alles ok,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_FileGetC(), pOS_FileGets(), pOS_FilePutC(), pOS_FilePuts(),
      pOS_FileFRead(), pOS_FileFWrite()

AMIGA FUNKTION
      LONG Flush(BPTR fh);
```

1.24 pdos.library/pOS_FileGetC()

```
PROTOTYP
      ULONG res = pOS_FileGetC
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle
        );

FUNKTION
      Lesen eines Zeichens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei

ERGEBNIS
      res (_R_D0)
        Gelesenes Zeichen im Bereich von 0-255,
        im Fehlerfall pOS_DOSERR. Dann kann mit pOS_GetIoErr()
        die Fehlerursache ermittelt werden.

SIEHE AUCH
      pOS_FilePutC(), pOS_FileGets(), pOS_FilePuts()

AMIGA FUNKTION
      LONG FGetC(BPTR fh);
```

1.25 pdos.library/pOS_FileGets()

```
PROTOTYP
      UBYTE *res = pOS_FileGets
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
```

```
          UBYTE *buffer,
          size_t size
        );

FUNKTION
      Lesen einer Zeile. (LF)

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      buffer (_R_A1)
        Speicherbereich, der die Daten aufnehmen soll
      size (_R_D0)
        Größe des Speicherbereichs

ERGEBNIS
        res (_R_D0)
          buffer wenn die Daten gelesen werden konnten,
          sonst NULL. Dann kann mit pOS_GetIoErr() die Fehlerursache
          ermittelt werden.

      HINWEIS:
        Es wird bis zum Auftreten des Zeilenvorschubs, des Datiendes
        oder size-1 Zeichen gelesen.
        Ist der Puffer zu klein, wird nicht die vollständige
        Zeile gelesen. Der Rest wird bei der nächsten Read-Anfrage
        zurückgegeben.
        Das gelesene LineFeed wird in ein '\0' umgewandelt.

SIEHE AUCH
      pOS_FilePuts(), pOS_FileGetC(), pOS_FilePutC()

AMIGA FUNKTION
      STRPTR FGets(BPTR fh, STRPTR buffer, ULONG size);
```

1.26 pdos.library/pOS_FilePutC()

```
PROTOTYP
      ULONG res = pOS_FilePutC
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          ULONG character
        );

FUNKTION
      Schreiben eines Zeichens

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
```

```
        FileHandle auf die Datei
      character (_R_D0)
        Zeichen das geschrieben werden soll
        (im Bereich von 0-255)

ERGEBNIS
      res (_R_D0)
        geschriebenes Zeichen,
        oder pOS_DOSERR im Fehlerfall. Dann kann mit pOS_GetIoErr()
        die Fehlerursache ermittelt werden.

SIEHE AUCH
      pOS_FileGetC(), pOS_FileGets(), pOS_FilePuts()

AMIGA FUNKTION
      LONG FPutC(BPTR fh, LONG character);
```

1.27 pdos.library/pOS_FilePuts()

```
PROTOTYP
      ULONG res = pOS_FilePuts
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const UBYTE *str
        );

FUNKTION
      Schreiben einer Zeile

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      str (_R_A1)
        Zeichenkette die geschrieben werden soll

ERGEBNIS
      res (_R_D0)
        Anzahl geschriebener Zeichen,
        im Fehlerfall pOS_DOSERR. Dann kann mit pOS_GetIoErr()
        die Fehlerursache ermittelt werden.

HINWEIS
      Das abschließende '\0' wird in ein LineFeed umgewandelt.

SIEHE AUCH
      pOS_FileGets(), pOS_FileGetC(), pOS_FilePutC()

AMIGA FUNKTION
      LONG FPuts(BPTR fh, STRPTR str);
```

1.28 pdos.library/pOS_VFWritef()

```
PROTOTYP
      dossize_t res = pOS_VFWritef
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const CHAR *format,
          const ULONG *args
        );

FUNKTION
      Schreiben einer formatierten Zeichenkette.
      Die Platzhalter des format-String werden mit den Argumenten
      aus args gefüllt.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      format (_R_A1)
        Zeiger auf den Formatstring
        siehe pOS_RawDoFmt() für die Platzhalterzeichen
      args (_R_A2)
        Zeiger auf die einzelnen Argumente, die im Formatstring
        aufgefüllt werden.

ERGEBNIS
      res (_R_D0)
        Anzahl geschriebener Zeichen,
        pOS_DOSERR im Fehlerfall. Dann kann mit pOS_GetIoErr() die
        Fehlerursache ermittelt werden.

SIEHE AUCH
      pOS_FilePuts()

AMIGA FUNKTION
      LONG VFWritef(BPTR fh, STRPTR format, LONG *args);
```

1.29 pdos.library/pOS_ExamineFH()

```
PROTOTYP
      BOOL res = pOS_ExamineFH
        (
          pOS_DosBase *dosbase,
          const pOS_FileHandle *filehandle,
          pOS_FileInfoBlock *fileinfoblock
        );

FUNKTION
      Ermitteln von Informationen über eine Datei oder ein Verzeichnis.
```

```
PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      fileinfoblock (_R_A1)
        Zeiger auf den FileInfoBlock, der die ermittelten Daten
        aufnehmen soll.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten ermittelt werden konnten
        sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      fileinfoblock muß mit pOS_AllocDosObject() allociert werden.

SIEHE AUCH
      pOS_ExamineObject(), pOS_ExNextObject()

AMIGA FUNKTION
      BOOL ExamineFH(BPTR fh, struct FileInfoBlock *fib);
```

1.30 pdos.library/pOS_FileFRead()

```
PROTOTYP
      ULONG res = pOS_FileFRead
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          APTR buffer,
          size_t blocksize,
          ULONG blocks
        );

FUNKTION
      Lesen einer Anzahl von Blöcken.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der die Daten aufnehmen
        soll
      blocksize (_R_D0)
        Größe eines Datenblocks (muß größer 0 sein)
      blocks (_R_D1)
        Anzahl von Blöcken, die gelesen werden sollen (muß größer 0 sein)

ERGEBNIS
      res (_R_D0)
```

```
        Anzahl der gelesenen Datenblöcke, in der Regel blocks.
        Im Fehlerfall ein kleinerer Wert. Dann liefert pOS_GetIoErr()
        die Fehlerursache.

SIEHE AUCH
      pOS_FileFWrite(), pOS_FileGets(), pOS_FileGetC()

AMIGA FUNKTION
      LONG FRead(BPTR fh, STRPTR buffer, ULONG blocksize, ULONG blocks);
```

1.31 pdos.library/pOS_FileFWrite()

```
PROTOTYP
      ULONG res = pOS_FileFWrite
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const APTR buffer,
          size_t blocksize,
          ULONG blocks
        );

FUNKTION
      Schreiben einer Anzahl von Blöcken.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der die Daten enthält
      blocksize (_R_D0)
        Größe eines Datenblocks (muß größer 0 sein)
      blocks (_R_D1)
        Anzahl von Blöcken, die geschrieben werden sollen  (muß größer 0 sein)

ERGEBNIS
      res (_R_D0)
        Anzahl der geschriebenen Datenblöcke, in der Regel blocks.
        Im Fehlerfall ein kleinerer Wert. Dann liefert pOS_GetIoErr()
        die Fehlerursache.

SIEHE AUCH
      pOS_FileFRead(), pOS_FilePuts(), pOS_FilePutC()

AMIGA FUNKTION
      LONG FWrite(BPTR fh, STRPTR buffer, ULONG blocksize, ULONG blocks);
```

1.32 pdos.library/pOS_FileEOF()

```
PROTOTYP
      BOOL res = pOS_FileEOF
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle
        );

FUNKTION
      Prüfen auf Dateiende

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei

ERGEBNIS
      res (_R_D0)
        liefert TRUE wenn eine Lesedatei am Dateiende steht,
        bei einer Schreibdatei undefiniert.

SIEHE AUCH
      pOS_FileRead(), pOS_FileGetC(), pOS_FileGets()

AMIGA FUNKTION
```

1.33 pdos.library/pOS_FileUnGetC()

```
PROTOTYP
      ULONG res = pOS_FileUnGetC
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          ULONG character
        );

FUNKTION
      Zurückstellen eines Zeichens in eine Lesedatei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei
      character (_R_D0)
        Das Zeichen das zurückgestellt werden soll und beim nächsten
        Aufruf von pOS_FileGetC() geliefert wird.

ERGEBNIS
      res (_R_D0)
        das zurückgestellte Zeichen oder pOS_DOSERR falls
        bereits ein Zeichen zurückgestellt und nicht gelesen
```

```
        wurde.

SIEHE AUCH
      pOS_FileGetC(), pOS_FileGets(), pOS_FileRead(), pOS_FileFRead()

AMIGA FUNKTION
      LONG UnGetC(BPTR fh, LONG character);
```

1.34 pdos.library/pOS_ChangeDosFileAccess()

```
PROTOTYP
      BOOL res = pOS_ChangeDosFileAccess
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          ULONG mode
        );

FUNKTION
      Ändern der Zugriffsart auf eine Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei, deren Zugriffsart geändert
        werden soll
      mode (_R_D0) (enum pOS_FileHandleMode)
        neue Zugriffsart, möglich sind
        FILEHDMOD_Read : Datei kann ausgelesen werden
        FILEHDMOD_Write: Datei kann beschrieben werden

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Zugriffsmodus geändert wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.
        (z.B. wenn bereits ein anderer Process die Datei benutzt)

SIEHE AUCH
      pOS_FileOpen(),

AMIGA FUNKTION
      BOOL ChangeMode(ULONG type, BPTR fh, ULONG mode);
```

1.35 pdos.library/pOS_SetFileDate()

```
PROTOTYP
      BOOL res = pOS_SetFileDate
        (
          pOS_DosBase *dosbase,
```

```
          pOS_FileHandle *filehandle,
          const pOS_DateStamp *datestamp
        );

FUNKTION
      Setzen des Änderungsdatums eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die Datei, deren Datum/Zeit geändert
        werden soll.
      datestamp (_R_A1)
        Neue Uhrzeit und Datum für die Datei.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neuen Werte gesetzt wurden,
        sonst FALSE. Dann kann mit pOS_GetIoErr() die Fehlerursache
        ermittelt werden.

SIEHE AUCH
      pOS_OpenFile(), pOS_SetObjectDate()

AMIGA FUNKTION
      BOOL SetFileDate(STRPTR filename, struct DateStamp *datestamp);
```

1.36 pdos.library/pOS_SetFileSize()

```
PROTOTYP
      BOOL res = pOS_SetFileSize
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          dossize_t size
        );

FUNKTION
      Verändern der Größe einer geöffneten Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die geöffnete Datei
      size (_R_D0)
        neue Größe der Datei

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Datei auf die neue Länge verlängert/verkürzt
        wurde. FALSE im Fehlerfall, dann liefert pOS_GetIoErr()
        die Fehlerursache.
```

```
HINWEIS
      Beim Verkürzen wird einfach der Rest der Datei abgeschnitten.
      Beim Verlängern wird der neue Bereich mit '\0' gefüllt.

SIEHE AUCH
      pOS_OpenFile(), pOS_SetObjectSize()

AMIGA FUNKTION
      LONG SetFileSize(BPTR fh, LONG offset, LONG mode);
```

1.37 pdos.library/pOS_OpenFileFromLock()

```
PROTOTYP
      __ARID__ pOS_FileHandle *res = pOS_OpenFileFromLock
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock
        );

FUNKTION
      Öffnen einer bereits verriegelten Datei. Danach kann
      auf die Datei zugegriffen werden.

      Ist der Lock FILELKACC_Shared so wird die Datei im Mode
      FILEHDMOD_Read geöffnet.
      Ist der Lock vom Type FILELKACC_Exclusive wird die Datei
      im Mode FILEHDMOD_Write geöffnet.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        FileLock auf die Datei, welcher in einen FileHandle umgewandelt
        werden soll.

ERGEBNIS
      res (_R_D0)
        FileHandle auf die Datei, die mit pOS_CloseFile() wieder
        geschlossen werden muß. NULL im Fehlerfall.
        Konnte die FH erfolgreich geöffnet werden, ist der Lock nicht
        mehr verfügbar. Konnte jedoch das File nicht geöffnet werden, so
        existiert der Lock weiterhin.

SIEHE AUCH
      pOS_FileOpen(); pOS_LockObject()

AMIGA FUNKTION
      BPTR OpenFromLock(BPTR lock);
```

1.38 pdos.library/pOS_NameFromFH()

```
PROTOTYP
      BOOL res = pOS_NameFromFH
        (
          pOS_DosBase *dosbase,
          const pOS_FileHandle *filehandle,
          dosname_t *buffer,
          size_t size
        );

FUNKTION
      Ermitteln des vollständigen Pfadnamens einer geöffneten Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf die geöffnete Datei
      buffer (_R_A1)
        Speicherbereich der den vollständigen Pfad aufnehmen soll
      size (_R_D0)
        Größe von buffer

ERGEBNIS
      res (_R_D0)
        TRUE wenn der vollständige Pfad in buffer steht.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die
        Fehlerursache (DOSERR_NameToLong wenn buffer zu klein ist).

SIEHE AUCH
      pOS_NameFromObjectLock()

AMIGA FUNKTION
      BOOL NameFromFH(BPTR fh, STRPTR buffer, LONG size)
```

1.39 pdos.library/pOS_SetDosScreenMode()

```
PROTOTYP
      BOOL res = pOS_SetDosScreenMode
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          ULONG mode
        );

FUNKTION
      Ändern eines Handlerprozesses zwischen CON und RAW-Modus.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle auf das CON: oder RAW:-Fenster
```

```
      mode (_R_D0)
        0 : RAW-Modus aktivieren
        1 : CON-Modus aktivieren

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Modus gesetzt wurde.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH

AMIGA FUNKTION
      BOOL SetMode(BPTR fh, LONG mode);
```

1.40 pdos.library/pOS_ConstructDosFH()

```
PROTOTYP
      VOID pOS_ConstructDosFH
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          pOS_DosDevice *dosdevice,
          ULONG mode
        );

FUNKTION
      FileHandle-Struktur installieren.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        Zeiger auf den FileHandle, welcher installiert werden soll
      dosdevice (_R_A1)
        eigener Handler
      mode (_R_D0) (enum pOS_FileHandleMode)
        Dateimodus
          FILEHDMOD_Read, FILEHDMOD_Write, FILEHDMOD_ReadWrite,
          FILEHDMOD_Append, FILEHDMOD_GBuffer, FILEHDMOD_DobBuf

HINWEIS
      Nur für Handler sinnvoll.

SIEHE AUCH
      pOS_DestructDosFH(), pOS_OpenFile()

AMIGA FUNKTION
```

1.41 pdos.library/pOS_DestructDosFH()

```
PROTOTYP
      VOID pOS_DestructDosFH
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle
        );

FUNKTION
      FileHandle-Struktur deinstallieren.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        Zeiger auf den FileHandle, welcher deinstalliert werden soll

SIEHE AUCH
      pOS_ConstructDosFH()

HINWEIS
      Nur für Handler sinnvoll.

AMIGA FUNKTION
```

1.42 pdos.library/pOS_LockObject()

```
PROTOTYP
      __ARID__ pOS_FileLock *res =  pOS_LockObject
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          const dosname_t *filename,
          ULONG mode
        );

FUNKTION
      Vorbereiten eines Verzeichnisses oder einer Datei für Zugriffe

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        wird ein filelock angegeben, kann der filename relativ
        dazu angegeben werden.
        bei NULL wird pr_CurrentDir verwendet
      filename (_R_A1)
        Name des Verzeichnisses oder der Datei
      mode (_R_D0) (enum pOS_FileLockAccess)
        FILELKACC_Shared    : andere Processe dürfen Lesend
                              auf die Datei/Verzeichnis zugreifen
        FILELKACC_Exclusive : alleiniges Zugriffsrecht auf die Datei,
                              z.B. zum Beschreiben
```

```
        FILELKACC_Unique    : MultiAssigns werden NICHT berücksichtigt
        FILELKACC_NoReq     : Es wird im Fehlerfall kein Report-Requester ↩
           geöffnet

ERGEBNIS
      res (_R_D0)
        FileLock auf das verriegelte Verzeichnis oder Datei.
        Muß mit pOS_UnlockObject() wieder aufgehoben werden.
        NULL im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_UnlockObject(), pOS_FileOpen(), pOS_OpenFileFromLock()

AMIGA FUNKTION
      BPTR Lock(STRPTR filename, LONG mode);
```

1.43 pdos.library/pOS_UnlockObject()

```
PROTOTYP
      VOID pOS_UnlockObject
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileLock *filelock
        );

FUNKTION
      Zugriff auf ein Verzeichnis oder eine Datei beenden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren Verriegelung aufgehoben werden soll.
        ACHTUNG: auf filelock darf nach dieser Funktion nicht
                 mehr zugegriffen werden darf

SIEHE AUCH
      pOS_LockObject(), pOS_DupObjectLock(), pOS_ParentObjectDir(),
      pOS_CreateDirectory()

AMIGA FUNKTION
      VOID UnLock(BPTR fh);
```

1.44 pdos.library/pOS_DupObjectLock()

```
PROTOTYP
      pOS_FileLock *res = pOS_DupObjectLock
        (
          pOS_DosBase *dosbase,
```

```
          const pOS_FileLock *filelock
        );

FUNKTION
      Duplizieren eines FileLocks.
      Funktioniert nur auf Shared-Locks!

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren FileLock dupliziert werden soll

ERGEBNIS
      res (_R_D0)
        Zeiger auf den duplizierten FileLock. Dieser muß mittel
        pOS_UnlockObject() wieder freigegeben werden.
        NULL im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_DupFileHandle(), pOS_UnlockObject(), pOS_LockObject()

AMIGA FUNKTION
      BPTR DupLock(BPTR lock);
```

1.45 pdos.library/pOS_ParentObjectDir()

```
PROTOTYP
      pOS_FileLock *res = pOS_ParentObjectDir
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock
        );

FUNKTION
      Verriegeln des übergeordneten Verzeichnisses eines bereits
      verriegelten Verzeichnisses oder Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren übergeordnetes Verzeichnis verriegelt werden soll.

ERGEBNIS
        res (_R_D0)
          FileLock auf das übergeordnete verriegelte Verzeichnis.
          Dieser muß mit pOS_UnlockObject() wieder aufgehoben werden.
          NULL im Fehlerfall, dann liefert pOS_IoErr() die Fehlerursache.

      HINWEIS:
        Liefert NULL wenn der filelock auf ein Laufwerk/Volume zeigt
```

```
        (z.B. SYS:), da hier kein übergeordnetes Verzeichnis existiert.

SIEHE AUCH
      pOS_UnlockObject(), pOS_LockObject()

AMIGA FUNKTION
      BPTR ParentDir(BPTR lock);
```

1.46 pdos.library/pOS_NameFromObjectLock()

```
PROTOTYP
      BOOL res = pOS_NameFromObjectLock
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          dosname_t *buffer,
          size_t size
        );

FUNKTION
      Ermitteln des vollständigen Pfadnamens einer bereits
      verriegelten Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei
      buffer (_R_A1)
        Speicherbereich der den vollständigen Pfadnamen aufnehmen soll
      size (_R_D0)
        Größe von buffer

ERGEBNIS
      res (_R_D0)
        TRUE wenn der vollständige Pfad in buffer steht.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die
        Fehlerursache (DOSERR_NameToLong wenn buffer zu klein ist).

SIEHE AUCH
      pOS_NameFromFH(), pOS_LockObject()

AMIGA FUNKTION
      BOOL NameFromLock(BPTR fh, STRPTR buffer, LONG size);
```

1.47 pdos.library/pOS_ExamineObject()

```
PROTOTYP
      BOOL res = pOS_ExamineObject
        (
```

```
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          pOS_FileInfoBlock *fileinfoblock
        );

FUNKTION
      Untersuchen eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        über welche die Informationen ermittelt werden.
        Sollen mit pOS_ExNextObject() weitere Datei-Informationen
        erfragt werden, muß filelock auf ein Verzeichnis zeigen.
      fileinfoblock (_R_A1)
        Zeiger auf den FileInfoBlock der die Daten aufnehmen soll.
        Der Speicher muß mittels pOS_AllocDosObject() reserviert und
        mit pOS_FreeDosObject() freigegeben werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten ermittelt werden konnten.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_LockObject(), pOS_ExamineFH(), pOS_ExNextObject(),
      ExamineObjectName()

AMIGA FUNKTION
      BOOL Examine(BPTR fh, struct FileInfoBlock *fib);
```

1.48 pdos.library/pOS_ExNextObject()

```
PROTOTYP
      BOOL res = pOS_ExNextObject
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          pOS_FileInfoBlock *fileinfoblock
        );

FUNKTION
      Untersuchen der nächsten Datei innerhalb eines Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis
      fileinfoblock (_R_A1)
        Zeiger auf den FileInfoBlock der die Daten aufnehmen soll.
        Der Speicher muß mittels pOS_AllocDosObject() reserviert und
```

```
        mit pOS_FreeDosObject() freigegeben werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn weitere Dateien ermittelt werden konnten.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.
        (DOSERR_NoMoreEntries wenn keine weiteren Dateien vorhanden sind)

SIEHE AUCH
      pOS_ExamineObject()

AMIGA FUNKTION
      BOOL ExNext(BPTR fh, struct FileInfoBlock *fib);
```

1.49 pdos.library/pOS_IsFileSystem()

```
PROTOTYP
      BOOL res = pOS_IsFileSystem
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock
        );

FUNKTION
      Ermitteln ob es sich bei einem Gerät um ein Dateisystem handelt.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei
        oder das Volume.

ERGEBNIS
      res (_R_D0)
        TRUE wenn es sich um ein FileSystem handelt, sonst FALSE.

SIEHE AUCH
      pOS_LockObject(), pOS_IsFileSystemName()

AMIGA FUNKTION
      BOOL IsFileSystem(STRPTR filename);
```

1.50 pdos.library/pOS_CreateDirectory()

```
PROTOTYP
      __ARID__ pOS_FileLock *res = pOS_CreateDirectory
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
```

```
          const dosname_t *dirname
        );

FUNKTION
      Erzeugen eines neuen Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        wird ein filelock angegeben, kann der dirname relativ
        dazu angegeben werden.
      dirname (_R_A1)
        Name des Verzeichnisses, das erzeugt werden soll. Darf auch
        einen Pfad enthalten, welcher aber bereits existieren muß.

ERGEBNIS
      res (_R_D0)
        FileLock auf das erzeugte und verriegelte Verzeichnis
        (vom Typ Exclusive-Lock).
        Muß mit pOS_UnlockObject() wieder aufgehoben werden.
        NULL im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_UnlockObject(), pOS_CreatePath()

AMIGA FUNKTION
      BPTR CreateDir(STRPTR dirname);
```

1.51 pdos.library/pOS_RenameObject()

```
PROTOTYP
      BOOL res = pOS_RenameObject
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          pOS_FileLock *filelock,
          const dosname_t *filename
        );

FUNKTION
      Umbenenen einer Datei oder eines Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filelock (_R_A1)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        welche umbenannt werden soll.
      filename (_R_A2)
        Neuer Name des Verzeichnisses oder der Datei.
```

```
ERGEBNIS
      res (_R_D0)
        TRUE wenn das Verzeichnis/die Datei umbenannt werden konnte.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_LockObject(), pOS_RenameObjectName()

AMIGA FUNKTION
      BOOL Rename(STRPTR oldfilename, STRPTR newfilename);
```

1.52 pdos.library/pOS_DeleteObjectLk()

```
PROTOTYP
      BOOL res = pOS_DeleteObjectLk
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileLock *filelock
        );

FUNKTION
      Löschen eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        welche gelöscht werden soll.

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Verzeichnis/die Datei gelöscht wurde. Dann
        darf auf filelock nicht mehr zugegriffen werden.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Ein Verzeichnis kann nur gelöscht werden, wenn sich keine
      Dateien mehr darin befinden.

SIEHE AUCH
      pOS_LockObject(), pOS_DeleteObjectName()

AMIGA FUNKTION
      BOOL DeleteFile(STRPTR name);
```

1.53 pdos.library/pOS_IsFileSystemName()

```
PROTOTYP
```

```
      BOOL res = pOS_IsFileSystemName
        (
          pOS_DosBase *dosbase,
          _R_A0 const pOS_FileLock *filelock,
          _R_A1 const dosname_t *filename
        );

FUNKTION
      Ermitteln ob es sich bei einem Gerät um ein Dateisystem handelt.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        wird ein filelock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name des Volumes/Verzeichnisses/Datei

ERGEBNIS
      res (_R_D0)
        TRUE wenn es sich um ein FileSystem handelt,
        sonst FALSE.

SIEHE AUCH
      pOS_LockObject(), pOS_IsFileSystem()

AMIGA FUNKTION
      BOOL IsFileSystem(STRPTR filename);
```

1.54 pdos.library/pOS_DeleteObjectName()

```
PROTOTYP
      BOOL res = pOS_DeleteObjectName
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *filename
        );

FUNKTION
      Löschen eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein filelock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name des Verzeichnisses oder der Datei, die gelöscht werden
        soll.

ERGEBNIS
```

```
      res (_R_D0)
        TRUE wenn das Verzeichnis/die Datei gelöscht wurde.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Ein Verzeichnis kann nur gelöscht werden, wenn sich keine
      Dateien mehr darin befinden.

SIEHE AUCH
      pOS_LockObject(), pOS_DeleteObjectLk()

AMIGA FUNKTION
      BOOL DeleteFile(STRPTR name);
```

1.55 pdos.library/pOS_SetObjectDate()

```
PROTOTYP
      BOOL res = pOS_SetObjectDate
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          const pOS_DateStamp *datestamp
        );

FUNKTION
      Setzen des Änderungsdatums eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei
      datestamp (_R_A1)
        Neue Uhrzeit und Datum für die Datei.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neuen Werte gesetzt wurden,
        sonst FALSE. Dann kann mit pOS_GetIoErr() die Fehlerursache
        ermittelt werden.

SIEHE AUCH
      pOS_LockObject(), pOS_SetFileDate(), pOS_ExamineObject(),
      pOS_SetObjectDateName()

AMIGA FUNKTION
      BOOL SetFileDate(STRPTR dateiname, struct DateStamp *datestamp);
```

1.56 pdos.library/pOS_SetObjectSize()

```
PROTOTYP
      BOOL res = pOS_SetObjectSize
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          dossize_t size
        );

FUNKTION
      Verändern der Größe einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf die verriegelte Datei
      size (_R_D0)
        Neue Größe der Datei

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Datei auf die neue Länge verlängert/verkürzt
        wurde. FALSE im Fehlerfall, dann liefert pOS_GetIoErr()
        die Fehlerursache.

HINWEIS
      Wird die Funktion auf ein Verzeichnis angewendet,
      wird immer FALSE zurückgeliefert.

SIEHE AUCH
      pOS_LockObject(), pOS_SetFileSize(), pOS_Examine(),
      pOS_SetObjectSizeName()

AMIGA FUNKTION
      LONG SetFileSize(BPTR fh, LONG offset, LONG mode);
```

1.57 pdos.library/pOS_GetDosInfoData()

```
PROTOTYP
      BOOL res = pOS_GetDosInfoData
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          pOS_DosInfoData *dosinfodata
        );

FUNKTION
      Ermitteln von Informationen über einen Datenträger.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
```

```
      filelock (_R_A0)
        Filelock auf den Datenträger, ein Verzeichnis oder eine Datei
      dosinfodata (_R_A1)
        Zeiger auf DosInfoData der das Ergebnis aufnehemn soll
        Der Speicher muß mittels pOS_AllocDosObject() reserviert und
        mit pOS_FreeDosObject() freigegeben werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten ermittelt werden konnten.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_LockObject(), pOS_GetDosInfoDataName()

AMIGA FUNKTION
      BOOL Info(BPTR fh, struct InfoData *dosinfodata);
```

1.58 pdos.library/pOS_SetObjectComment()

```
PROTOTYP
      BOOL res = pOS_SetObjectComment
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          const CHAR *comment
        );

FUNKTION
      Setzen eines Dateikommentars

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren Kommentar geändert werden soll
      comment (_R_A1)
        Kommentar der gesetzt werden soll

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Kommentar gesetzt wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_LockObject(), pOS_SetObjectCommentName()

AMIGA FUNKTION
      BOOL SetComment(STRPTR filename, STRPTR comment);
```

1.59 pdos.library/pOS_SetObjectProtection()

```
PROTOTYP
      BOOL res = pOS_SetObjectProtection
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          ULONG flags
        );

FUNKTION
      Setzen der Schutzbits für ein Verzeichnis oder eine Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren Schutzbits verändert werden sollen.
      flags (_R_D0) (enum pOS_FileProtectionBits)
        Bit ist gesetzt wenn
        FIBF_Delete     : Datei gelöscht werden kann
        FIBF_Execute    : Datei ausgeführt werden kann
        FIBF_Write      : Datei beschrieben werden kann
        FIBF_Read       : Datei gelesen werden kann

        FIBF_Archive    : Datei seit der letzten Archivierung nicht
                          geändert wurde.
                          Dieses Bit wird bei jedem Schreibzugriff vom
                          Betriebssystem gelöscht.
        FIBF_Pure       : Datei wiedereintrittsfähig ist
        FIBF_Script     : Datei ein Script ist (kann ohne Execute
                          gestartet werden)

        Gruppenzugehörigkeit, Bit ist gesetzt wenn
        FIBF_GRP_Delete : Gruppe löschen darf
        FIBF_GRP_Execute: Gruppe ausführen darf
        FIBF_GRP_Write  : Gruppe schreiben darf
        FIBF_GRP_Read   : Gruppe lesen darf


        Andere, Bit ist gesetzt wenn
        FIBF_OTR_Delete : Anderer löschen darf
        FIBF_OTR_Execute: Anderer ausführen darf
        FIBF_OTR_Write  : Anderer schreiben darf
        FIBF_OTR_Read   : Anderer lesen darf

        Beim Erzeugen einer Datei werden die Bits so gesetzt,
        so daß die Datei lesbar/schreibbar/löschbar und evtl. ausführbar ist.
        Gruppenzugehörige und Andere haben keine Zugriffsrechte.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neuen Bits gesetzt werden konnten,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.
```

```
HINWEIS
      Die Schutzbits können nur vom Eigentümer verändert werden.
      Versuchen andere diese zu ändern, wird als Fehlerursache
      DOSERR_NoAccess zurückgegeben.

SIEHE AUCH
      pOS_LockObject(), pOS_ExamineObject(), pOS_SetObjectProtectionName()

AMIGA FUNKTION
      BOOL SetProtection(STRPTR filename, LONG flags);
```

1.60 pdos.library/pOS_SameDosDevice()

```
PROTOTYP
      BOOL res = pOS_SameDosDevice
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock1,
          const pOS_FileLock *filelock2
        );

FUNKTION
      Ermitteln, ob sich zwei Objekte auf dem selben Gerät befinden.
      Die Objekte können dabei auf unterschiedlichen Partitionen des
      selben physikalischen Gerätes sein.

      res= (pOS_SameDosObject() & FILELKSF_Device) ? TRUE:FALSE

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock1 (_R_A0)
        Zeiger auf das erste Objekt
      filelock2 (_R_A1)
        Zeiger auf das zweite Objekt

ERGEBNIS
      res (_R_D0)
        TRUE wenn sich beide Objekte auf dem selben Gerät befinden,
        sonst FALSE.

SIEHE AUCH
      pOS_LockObject(), pOS_SameDosObject()

AMIGA FUNKTION
      LONG SameDevice(BPTR filelock1, BPTR filelock2);
```

1.61 pdos.library/pOS_SameDosObject()

```
PROTOTYP
      ULONG res = pOS_SameDosObject
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock1,
          const pOS_FileLock *filelock2
        );

FUNKTION
      Ermitteln, ob sich zwei Filelocks auf dasselbe Objekt beziehen.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock1 (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei
      filelock2 (_R_A1)
        Zeiger auf das verriegelte Verzeichnis oder die Datei

ERGEBNIS
      res (_R_D0) (enum pOS_SameFileLockType)
        FILELKSF_Different : wenn sich die Objekte auf unterschiedlichen
                            Datenträger befinden
        FILELKSF_Device    : wenn sich die Objekte auf dem selben Gerät
                            befinden (z.B. scsi.device, Unit 0)
        FILELKSF_DDev      : wenn sich die Objekte auf dem selben DosDevice
                            befinden (z.B. a:, df0:, dh1:)
        FILELKSF_Volume    : wenn sich die Objekte auf dem selben Volume
                            befinden (z.B. Ram Disk:)
        FILELKSF_Object      und
        FILELKSF_Same      : die beiden Locks bezeihen sich auf das selbe Objekt

SIEHE AUCH
      pOS_LockObject(), pOS_DupObjectLock(), pOS_SameDosDevice()

AMIGA FUNKTION
      LONG SameLock(BPTR filelock1, BPTR filelock2);
```

1.62 pdos.library/pOS_ChangeDosObjectAccess()

```
PROTOTYP
      BOOL res = pOS_ChangeDosObjectAccess
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          ULONG mode
        );

FUNKTION
      Ändern des Zugriffsmodus

PARAMETER
```

```
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf das verriegelte Verzeichnis oder die Datei,
        deren Zugriffsmodus geändert werden soll
      mode (_R_D0) (enum pOS_FileLockAccess)
        FILELKACC_Shared    : andere Processe können lesend auf
                              die Datei zugreifen
        FILELKACC_Exclusive : alleiniges Zugriffsrecht (z.B. zum Beschreiben)

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Zugriffsmodus geändert wurde,
        sonst FALSE. Dann gibt pOS_GetIoErr() die Fehlerursache zurück.

SIEHE AUCH
      pOS_LockObject()

AMIGA FUNKTION
      BOOL ChangeMode(ULONG type, BPTR filelock, ULONG mode);
```

1.63 pdos.library/pOS_ConstructDosLk()

```
PROTOTYP
      VOID pOS_ConstructDosLk
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock,
          pOS_DosDevice *dosdevice,
          ULONG mode
        );

FUNKTION
      FileLock-Struktur installieren.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf die zu installierende FileLock-Struktur
      dosdevice (_R_A1)
        eigener Handler
      mode (_R_D0) (enum pOS_FileLockAccess)
        FILELKACC_Shared, FILELKACC_Exclusive

HINWEIS
      Nur für Handler sinnvoll.

SIEHE AUCH
      pOS_DestructDosLk(), pOS_LockObject()

AMIGA FUNKTION
```

1.64 pdos.library/pOS_DestructDosLk()

```
PROTOTYP
      VOID pOS_DestructDosLk
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *filelock
        );

FUNKTION
      FileLock-Struktur deinstallieren.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf die zu deinstallierende FileLock-Struktur

HINWEIS
      Nur für Handler sinnvoll.

SIEHE AUCH
      pOS_ConstructDosLk()

AMIGA FUNKTION
```

1.65 pdos.library/pOS_RenameObjectName()

```
PROTOTYP
      BOOL res = pOS_RenameObjectName
        (
          pOS_DosBase *dosbase,
          const struct pOS_FileLock *currentlock,
          const dosname_t *oldname,
          const dosname_t *newname
        );

FUNKTION
      Umbenenen einer Datei oder eines Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      oldname (_R_A1)
        Alter Name des Verzeichnisses oder der Datei.
      newname (_R_A2)
        Neuer Name des Verzeichnisses oder der Datei.

ERGEBNIS
```

```
      res (_R_D0)
        TRUE wenn das Verzeichnis/die Datei umbenannt werden konnte.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_RenameObject()

AMIGA FUNKTION
      BOOL Rename(STRPTR oldfilename, STRPTR newfilename);
```

1.66 pdos.library/pOS_SetObjectDateName()

```
PROTOTYP
      BOOL res = pOS_SetObjectDateName
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *currentlock,
          const dosname_t *filename,
          const pOS_DateStamp *datestamp
        );

FUNKTION
      Setzen des Änderungsdatums eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name des Verzeichnisses oder der Datei, deren Datum und
        Uhrzeit neu gesetzt werden soll.
      datestamp (_R_A2)
        Neue Uhrzeit und Datum für die Datei.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neuen Werte gesetzt wurden,
        sonst FALSE. Dann kann mit pOS_GetIoErr() die Fehlerursache
        ermittelt werden.

SIEHE AUCH
      pOS_SetFileDate(), pOS_SetObjectDate()

AMIGA FUNKTION
      BOOL SetFileDate(STRPTR dateiname, struct DateStamp *datestamp);
```

1.67 pdos.library/pOS_SetObjectSizeName()

```
PROTOTYP
      BOOL res = pOS_SetObjectSizeName
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *currentlock,
          const dosname_t *filename,
          dossize_t size
        );

FUNKTION
      Verändern der Größe einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name des Verzeichnisses oder der Datei, dessen Größe neu
        gesetzt werden sollen.
      size (_R_D0)
        Neue Größe der Datei

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Datei auf die neue Größe verlängert/verkürzt
        wurde. FALSE im Fehlerfall, dann liefert pOS_GetIoErr()
        die Fehlerursache.

SIEHE AUCH
      pOS_SetFileSize(), pOS_SetObjectSize()

AMIGA FUNKTION
      LONG SetFileSize(BPTR fh, LONG offset, LONG mode);
```

1.68 pdos.library/pOS_GetDosInfoDataName()

```
PROTOTYP
      BOOL res = pOS_GetDosInfoDataName
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *filename,
          pOS_DosInfoData *dosinfodata
        );

FUNKTION

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
```

```
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name des Datenträger, des Verzeichnises oder einer Datei
      dosinfodata (_R_A2)
        Zeiger auf DosInfoData der das Ergebnis aufnehemn soll
        Der Speicher muß mittels pOS_AllocDosObject() reserviert und
        mit pOS_FreeDosObject() freigegeben werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten ermittelt werden konnten.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_GetDosInfoData()

AMIGA FUNKTION
      BOOL Info(BPTR fh, struct InfoData *dosinfodata);
```

1.69 pdos.library/pOS_SetObjectCommentName()

```
PROTOTYP
      BOOL res = pOS_SetObjectCommentName
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *currentlock,
          const dosname_t *filename,
          const CHAR *comment
        );

FUNKTION
      Setzen eines Dateikommentars

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name eines Verzeichnises oder einer Datei
      comment (_R_A2)
        Kommentar der gesetzt werden soll

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Kommentar gesetzt wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_SetObjectComment()
```

```
AMIGA FUNKTION
      BOOL SetComment(STRPTR filename, STRPTR comment);
```

1.70 pdos.library/pOS_SetObjectProtectionName()

```
PROTOTYP
      BOOL res = pOS_SetObjectProtectionName
        (
          pOS_DosBase *dosbase,
          pOS_FileLock *currentlock,
          const dosname_t *filename,
          ULONG flags
        );

FUNKTION
      Setzen der Schutzbits für ein Verzeichnis oder eine Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name eines Verzeichnises oder einer Datei
      flags (_R_D0) (enum pOS_FileProtectionBits)
        Bit ist gesetzt wenn
        FIBF_Delete     : Datei gelöscht werden kann
        FIBF_Execute    : Datei ausgeführt werden kann
        FIBF_Write      : Datei beschrieben werden kann
        FIBF_Read       : Datei gelesen werden kann

        FIBF_Archive    : Datei seit der letzten Archivierung nicht
                          geändert wurde.
                          dieses Bit wird bei jedem Schreibzugriff vom
                          Betriebssystem gelöscht.
        FIBF_Pure       : Datei wiedereintrittsfähig ist
        FIBF_Script     : Datei ein Script ist (kann ohne Execute
                          gestartet werden)

        Gruppenzugehörigkeit, Bit ist gesetzt wenn
        FIBF_GRP_Delete : Gruppe löschen darf
        FIBF_GRP_Execute: Gruppe ausführen darf
        FIBF_GRP_Write  : Gruppe schreiben darf
        FIBF_GRP_Read   : Gruppe lesen darf


        Andere, Bit ist gesetzt wenn
        FIBF_OTR_Delete : Anderer löschen darf
        FIBF_OTR_Execute: Anderer ausführen darf
        FIBF_OTR_Write  : Anderer schreiben darf
        FIBF_OTR_Read   : Anderer lesen darf

        Beim Erzeugen einer Datei werden die Bits so gesetzt,
```

```
        sodaß die Datei lesbar/schreibbar/löschbar und evtl. ausführbar ist.
        Gruppenzugehörige und Andere haben keine Zugriffsrechte.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die neuen Bits gesetzt werden konnten,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Die Schutzbits können nur vom Eigentümer verändert werden.
      Versuchen andere diese zu ändern, wird als Fehlerursache
      DOSERR_NoAccess zurückgegeben.

SIEHE AUCH
      pOS_SetObjectProtection()

AMIGA FUNKTION
      BOOL SetProtection(STRPTR filename, LONG flags);
```

1.71 pdos.library/pOS_ExamineObjectName()

```
PROTOTYP
      BOOL res = pOS_ExamineObjectName
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *filename,
          pOS_FileInfoBlock *fileinfoblock
        );

FUNKTION
      Untersuchen eines Verzeichnisses oder einer Datei.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      filename (_R_A1)
        Name eines Verzeichnises oder einer Datei
      fileinfoblock (_R_A2)
        Zeiger auf den FileInfoBlock der die Daten aufnehmen soll.
        Der Speicher muß mittels pOS_AllocDosObject() reserviert und
        mit pOS_FreeDosObject() freigegeben werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten ermittelt werden konnten.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      ExamineObject()
```

```
AMIGA FUNKTION
      BOOL Examine(BPTR fh, struct FileInfoBlock *fib);
```

1.72 pdos.library/pOS_CopyObjectName()

```
PROTOTYP
      BOOL res = pOS_CopyObjectName
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *oldname,
          const dosname_t *newname,
          ULONG mode
        );

FUNKTION
      Kopieren einer Datei oder eines kompletten Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      oldname (_R_A1)
        Alter Name eines Verzeichnises oder einer Datei
           (currentlock wird beachtet)
      newname (_R_A2)
        Neuer Name eines Verzeichnises oder einer Datei
           (Process-CurrentDir wird beachtet, currentlock wird NICHT beachtet)
      mode (_R_D0)
        Für zukünftige Erweiterungen; z.Z. auf 0 setzen.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Datei oder das Verzeichnis kopiert wurde.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_MoveObjectName()

AMIGA FUNKTION
```

1.73 pdos.library/pOS_MoveObjectName()

```
PROTOTYP
      BOOL res = pOS_MoveObjectName
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
```

```
          const dosname_t *oldname,
          const dosname_t *newname,
          ULONG mode
        );

FUNKTION
      Verschieben einer Datei oder eines kompletten Verzeichnisses
      in ein anderes Verzeichnis oder auf einen anderen Datenträger.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der filename relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      oldname (_R_A1)
        Alter Name eines Verzeichnises oder einer Datei
           (currentlock wird beachtet)
      newname (_R_A2)
        Neuer Name eines Verzeichnises oder einer Datei,
        der sich auf auf einem anderen Datenträger weisen darf.
           (currentlock wird NICHT beachtet)
      mode (_R_D0)
        Für zukünftige Erweiterungen; z.Z. auf 0 setzen.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Datei oder das Verzeichnis verschoben wurden.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Befindet sich Quell- und Zieldatei auf dem selben Datenträger,
      wird die Datei umbenannt. Sonst wird die Datei zuerst auf den
      Zieldatenträger kopiert und dann vom Quelldatenträger gelöscht.

SIEHE AUCH
      pOS_CopyObjectName(), pOS_RenameObject(), pOS_RenameObjectName(),
      pOS_DeleteObjectLk(), pOS_DeleteObjectName()

AMIGA FUNKTION
```

1.74 pdos.library/pOS_CreatePath()

```
PROTOTYP
      __ARID__ pOS_FileLock *res = pOS_CreatePath
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock,
          const dosname_t *drawerpath
        );

FUNKTION
      Erzeugen eines neuen Verzeichnisses, evtl. mit weiteren
      Unterverzeichnissen.
```

```
PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock (_R_A0)
        wird ein currentlock angegeben, kann der drawerpath relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      drawerpath (_R_A1)
        Pfadangabe der Schublade(n) die erzeugt werden soll(en).

ERGEBNIS
      res (_R_D0)
        FileLock auf das zuletzt (tiefste) erzeugte und verriegelte
        Verzeichnis (vom Typ Exclusive-Lock).
        Muß mit pOS_UnlockObject() wieder aufgehoben werden.
        NULL im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Diese Funktion kann bei zusammengesetzten Pfaden, die komplette
      Pfadhierarchie herstellen und sollte deshalb der Funktion  ↩
         pOS_CreateDirectory()
      vorgezogen werden. Es dürfen bereits Teile des neuen Pfades
      existieren.

SIEHE AUCH
      pOS_UnlockObject(), pOS_CreateDirectory()

AMIGA FUNKTION
```

1.75 pdos.library/pOS_IsInfinite()

```
PROTOTYP
      SLONG res = pOS_IsInfinite
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *currentlock1,
          const dosname_t *dirname1,
          const pOS_FileLock *currentlock2,
          const dosname_t *dirname2
        );

FUNKTION
      Überprüfen, ob ein Verzeichnis ein Unterverzeichnis eines
      Anderen ist.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      currentlock1 (_R_A0)
        wird ein currentlock angegeben, kann der dirname1 relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      dirname1 (_R_A1)
        Name des ersten Verzeichnisses
      currentlock2 (_R_A2)
```

```
        wird ein currentlock angegeben, kann der dirname2 relativ
        dazu angegeben werden. Bei NULL wird pr_CurrentDir verwendet.
      dirname2 (_R_A3)
        Name des Verzeichnisses, bei dem geprüft wird, ob es Unter-
        verzeichnis des Ersten ist.

ERGEBNIS
      res (_R_D0)
        TRUE wenn das zweite Verzeichnis ein Unterverzeichnis des
        Ersten ist, sonst FALSE. Die Funktion arbeitet auch bei
        Assigns korrekt.

BEISPIELE
      pOS_IsInfinite(worklook,"verz2",NULL,"work:verz1/verz2/verz3")
        wobei worklock auf "Work:" zeigt
        -> liefert FALSE

      pOS_IsInfinite(work1look,"verz2",NULL,"work:verz1/verz2/verz3")
        wobei work1lock auf "Work:verz1" zeigt
        -> liefert TRUE

      pOS_IsInfinite(NULL,"A:verz2",NULL,"work:verz1/verz2/verz3")
        wobei der Assign A: auf "Work:verz1" zeigt
        -> liefert TRUE

HINWEIS
      pOS_IsInfinite(Source,Dest) sollte vor dem Kopieren eines
      Verzeichnis aufgerufen werden, um nichtendene rekursive
      Kopiervorgänge zu vermeiden.

SIEHE AUCH

AMIGA FUNKTION
```

1.76 pdos.library/pOS_AddPart()

```
PROTOTYP
      BOOL res = pOS_AddPart
        (
          pOS_DosBase *dosbase,
          dosname_t *buffer,
          const dosname_t *filepart,
          size_t size
        );

FUNKTION
      Anhängen eines Datei- oder Verzeichnisnamens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      buffer (_R_A0)
        Zeiger auf den Puffer der das Ergebnis aufnehmen soll
        und bereits einen Pfad-Namen enthalten darf.
```

```
        ACHTUNG: Die letzte Komponente darf kein Dateiname sein,
                 sonst wird ein falscher Pfad gebildet.
      filepart (_R_A1)
        Datei- oder Verzeichnisname, der an puffer angehängt
        werden soll; relative oder absolute Pfade sind erlaubt.
      size (_R_D0)
        Größe von buffer

ERGEBNIS
      res (_R_D0)
        TRUE wenn alles ok ist,
        FALSE wenn der Speicherbereich zu klein ist. buffer
        wird in diesem Fall nicht verändert. pOS_GetIoErr()
        liefert den Fehlercode DOSERR_NoMem.

BEISPIEL
      "a"    + "b"  => "a/b"
      "a/"   + "b"  => "a/b"
      "a:/"  + "b"  => "a:/b"
      "a//"  + "b"  => "a//b"
      "a:b"  + "c:" => "c:"
      "a/c"  + "/b" => "a/b"
      "a:c/" + "/b" => "a:b"
      "a:c/" + ""   => "a:c/"
      "a:c"  + "b/" => "a:c/b/"

      pOS_AddPart(...,"c:","datei",...)
      => "c:datei"
      pOS_AddPart(...,"c:verz1","verz2/datei",...)
      => "c:verz1/verz2/datei"
      pOS_AddPart(...,"c:verz1","s:verz2/datei",...)
      => "s:verz2/datei"
      pOS_AddPart(...,"c:verz1","/datei",...)
      => "c:datei"
      pOS_AddPart(...,"c:verz1/","/datei",...)
      => "c:datei"
      pOS_AddPart(...,"c:verz1/","datei",...)
      => "c:verz1/datei"
      pOS_AddPart(...,"c:datei","datei",...)
      => nicht erlaubt, liefert "c:datei/datei" <=

SIEHE AUCH
      pOS_PathPart(), pOS_FilePart(), pOS_SplitName()

AMIGA FUNKTION
      BOOL AddPart(STRPTR buffer, STRPTR filepart, ULONG size);
```

1.77 pdos.library/pOS_PathPart()

```
PROTOTYP
      dosname_t *res = pOS_PathPart
        (
          pOS_DosBase *dosbase,
          const dosname_t *filepart
```

```
        );

FUNKTION
      Ermittelt das Ende der vorletzen Komponente eines
      Pfadnamens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filepart (_R_A0)
        Zeiger auf die Pfadangabe

ERGEBNIS
      res (_R_D0)
        Zeiger auf das erste Zeichen hinter der vorletzen Komponente.
        In der Regel auf den begin des Dateinamens oder des vorherigen '/'.
        Die Adresse von filepart bis res enthält den Verzeichnisanteil.

BEISPIEL
      "xxx:yyy/zzz/qqq"  "xxx:yyy"  "aa:yyy/"
                 ^           ^           ^

      pOS_PathPart(...,"c:verz1/datei") => Zeiger auf '/'
      pOS_PathPart(...,"c:verz1/") => Zeiger auf '/'
      pOS_PathPart(...,"c:datei") => Zeiger auf 'd'
      pOS_PathPart(...,"datei") => Zeiger auf 'd'

SIEHE AUCH
      pOS_AddPart(), pOS_FilePart(), pOS_SplitName()

AMIGA FUNKTION
      STRPTR PathPart(STRPTR filepart);
```

1.78 pdos.library/pOS_FilePart()

```
PROTOTYP
      dosname_t *res = pOS_FilePart
        (
          pOS_DosBase *dosbase,
          const dosname_t *filepart
        );

FUNKTION
      Ermitteln der letzten Komponente einer Pfadangabe.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filepart (_R_A0)
        Zeiger auf die Pfadangabe

ERGEBNIS
      res (_R_D0)
        zeiger auf den ersten Buchstaben der letzen Komponente
```

```
        (in der Regel der Dateiname).

BEISPIEL
      "xxx:yyy/zzz/qqq"  "xxx:yyy"  "aa:yyy/"
                  ^          ^             ^

      pOS_FilePart(...,"c:verz1/datei") => Zeiger auf 'd'
      pOS_FilePart(...,"c:datei") => Zeiger auf 'd'
      pOS_FilePart(...,"datei") => Zeiger auf 'd'
      pOS_FilePart(...,"") => Zeiger auf '\0'
      pOS_FilePart(...,"c:verz1/") => Zeiger auf '\0'

SIEHE AUCH
      pOS_AddPart(), pOS_PathPart(), pOS_SplitName()

AMIGA FUNKTION
      STRPTR FilePart(STRPTR filepart);
```

1.79 pdos.library/pOS_SplitName()

```
PROTOTYP
      size_t res = pOS_SplitName
        (
          pOS_DosBase *dosbase,
          const dosname_t *filepart,
          ULONG separator,
          dosname_t *buffer,
          size_t oldpos,
          size_t size
        );

FUNKTION
      Zerlegen eines Pfadnamens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filepart (_R_A0)
        Zeiger auf die Pfadangabe, die zerlegt werden soll.
      separator (_R_D0)
        Trennzeichen zwischen den Komonenten
        (gewöhnlich ':' oder '/')
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der das Erbegnis
        aufnimmt
      oldpos (_R_D1)
        Aktuelle Position innerhalb der Pfadangabe
      size (_R_D2)
        Größe des buffers für das Ergebnis.

ERGEBNIS
      res (_R_D0)
        neue Position, hinter dem zerlegten Teil.
        -1 wenn die Pfanangabe komplet durchlaufen wurde.
```

```
BEISPIEL
      pOS_SplitName(...,"c:verz1/verz2/datei",':',...,0,...)
      => 2 "c"
      pOS_SplitName(...,"c:verz1/verz2/datei",':',...,2,...)
      => -1 "verz1/verz2/datei"

      pOS_SplitName(...,"c:verz1/verz2/datei",'/',...,2,...)
      => 8 "verz1"
      pOS_SplitName(...,"c:verz1/verz2/datei",'/',...,8,...)
      => 14 "verz2"
      pOS_SplitName(...,"c:verz1/verz2/datei",'/',...,14,...)
      => -1 "datei"

      res=pOS_SplitName(...,"verz1/datei",':',...,0...);
      => -1 "verz1/datei"
      res=pOS_SplitName(...,"c:datei",'/',...,0,...);
      => -1 "c:datei"
      res=pOS_SplitName(...,"",':',...,0,...);
      => -1 ""
      res=pOS_SplitName(...,"",'/',...,0,...);
      => -1 ""

SIEHE AUCH
      pOS_AddPart(), pOS_PathPart(), pOS_FilePart()

AMIGA FUNKTION
          WORD SplitName(STRPTR filepart, UBYTE separator, STRPTR buffer, WORD  ↩
             oldpos, LONG size);

      /*****************************************************************************/ ←
```

1.80 pdos.library/pOS_SetIoErr()

```
PROTOTYP
      SLONG res = pOS_SetIoErr
        (
          pOS_DosBase *dosbase,
          SLONG error
        );

FUNKTION
      Setzen eines pDOS-Fehlercodes. Dieser Wert wird
      von pOS_GetIoErr() zurückgeliefert.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      error (_R_D0) (enum pOS_DosErrors)
        neuer Fehlercode

ERGEBNIS
      res (_R_D0) (enum pOS_DosErrors)
```

```
        alter Fehlercode

SIEHE AUCH
      pOS_GetIoErr()

AMIGA FUNKTION
      LONG SetIoErr(LONG error);
```

1.81 pdos.library/pOS_GetIoErr()

```
PROTOTYP
      SLONG res = pOS_GetIoErr
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Ermitteln des pDOS-Fehlercodes, des letzen Dos-Funktions-
      Aufrufes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0) (enum pOS_DosErrors)
        pDOS-Fehlercode der zuletzt ausgeführten Dos-Funktion.

SIEHE AUCH
      pOS_SetIoErr(), pOS_GetDosErrText(), pOS_PrintDosErr()

AMIGA FUNKTION
      LONG IoErr();
```

1.82 pdos.library/pOS_ExitProcess()

```
PROTOTYP
      VOID pOS_ExitProcess
        (
          pOS_DosBase *dosbase,
          ULONG returnwert
        );

FUNKTION
      Beenden eines Processes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      returnwert (_R_D0)
```

```
        Rückgabewert des Processes (z.B. an die aufrufende Shell)
        Definierte Werte sind: DOSFAIL_OK, DOSFAIL_WARN, DOSFAIL_ABORT,
                               DOSFAIL_ERROR, DOSFAIL_FAIL

HINWEIS
      Diese LowLevel-Funktion sollte in C-Programmen nicht verwendet
      werden, sondern die ANSI-Funktion exit().
      Sonst werden nicht alle reservierten Daten freigeben!
      Z.Z. setzt diese Funktion nur den Fehlercode, das Programm
      wird aber nicht beendet!

SIEHE AUCH
      pOS_SetShellFail()

AMIGA FUNKTION
      VOID Exit(LONG returnwert);
```

1.83 pdos.library/pOS_GetStdInput()

```
PROTOTYP
      pOS_FileHandle *res = pOS_GetStdInput
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Ermitteln des Standard-Eingabestromes eines Processes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0)
        FileHandle des Eingabestromes der nicht verändert oder
        freigegeben werden darf. NULL wenn kein Eingabestrom
        definiert ist.

SIEHE AUCH
      pOS_GetStdOutput(), pOS_GetStdErrOutput()

AMIGA FUNKTION
      BPTR Input();
```

1.84 pdos.library/pOS_GetStdOutput()

```
PROTOTYP
      pOS_FileHandle *res = pOS_GetStdOutput
        (
          pOS_DosBase *dosbase
```

```
        );

FUNKTION
      Ermitteln des Standard-Ausgabestromes eines Processes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0)
        FileHandle des Ausgabestromes der nicht verändert oder
        freigegeben werden darf. NULL wenn kein Ausgabestrom
        definiert ist.

SIEHE AUCH
      pOS_GetStdInput(), pOS_GetStdErrOutput()

AMIGA FUNKTION
      BPTR Output();
```

1.85 pdos.library/pOS_GetStdErrOutput()

```
PROTOTYP
      pOS_FileHandle *res = pOS_GetStdErrOutput
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Ermitteln des Standard-Fehlerausgabestromes eines Processes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0)
        FileHandle des Fehlerausgabestromes der nicht verändert oder
        freigegeben werden darf. NULL wenn kein Fehlerausgabestrom
        definiert ist.

SIEHE AUCH
      pOS_GetStdInput(), pOS_GetStdOutput()

AMIGA FUNKTION
```

1.86 pdos.library/pOS_SetConsoleFH()

```
PROTOTYP
```

```
      __ARID__ pOS_FileHandle *res = pOS_SetConsoleFH
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileHandle *filehandle
        );

FUNKTION
      Setzen eines neuen FileHandles für den aktuellen Process.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        Neuer FileHandle, der z.B. mit pOS_OpenFile() besorgt
        werden muß

ERGEBNIS
      res (_R_D0)
        Alter FileHandle, der vor Beendigung des Processes wieder
        zurückgesetzt werden muß.

AMIGA FUNKTION
      struct MsgPort *SetConsoleTask(struct MsgPort *msgport);
```

1.87 pdos.library/pOS_SetProgDir()

```
PROTOTYP
      __ARID__ pOS_FileLock *res = pOS_SetProgDir
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileLock *filelock
        );

FUNKTION
      Setzen des Verzeichnisses, in dem sich die Programmdatei befindet.
        (PROGDIR:)

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)

ERGEBNIS
      res (_R_D0)
        FileLock des bisherigen Verzeichnisses.

HINWEIS
      Vor Programmende muß wieder das beim Programmstart gültige
      Verzeichnis gesetzt werden.

SIEHE AUCH
      pOS_GetProgDir()

AMIGA FUNKTION
```

```
      BPTR SetProgramDir(BPTR filelock);
```

1.88 pdos.library/pOS_GetProgDir()

```
PROTOTYP
      pOS_FileLock *res = pOS_GetProgDir
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Ermitteln des Verzeichnisses, in dem sich die Programmdatei befindet.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0)
        FileLock auf das Programm-Verzeichnis. Dieser Lock ist ein
        einfaches Dublikat und darf nicht freigegeben werden.

SIEHE AUCH
      pOS_SetProgDir()

AMIGA FUNKTION
      BPTR GetProgramDir();
```

1.89 pdos.library/pOS_ConstructProcess()

```
PROTOTYP
      VOID pOS_ConstructProcess
        (
          pOS_DosBase *dosbase,
          pOS_Process *process
        );

FUNKTION
      Process-Struktur installieren.  (internal)

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      process (_R_A0)
        Zeiger auf die Process-Struktur, die installiert werden soll.

SIEHE AUCH
      pOS_DestructProcess()

AMIGA FUNKTION
```

1.90 pdos.library/pOS_DestructProcess()

```
PROTOTYP
      VOID pOS_DestructProcess
        (
          pOS_DosBase *dosbase,
          pOS_Process *process
        );

FUNKTION
      Process-Struktur deinstallieren. (internal)

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      process (_R_A0)
        Zeiger auf die Process-Struktur, die deinstalliert werden soll.

SIEHE AUCH
      pOS_ConstructProcess()

AMIGA FUNKTION
```

1.91 pdos.library/pOS_DosDelay()

```
PROTOTYP
      VOID pOS_DosDelay
        (
          pOS_DosBase *dosbase,
          ULONG ticks
        );

FUNKTION
      Unterbrechen eines Processes. Während der Wartezeit steht
      der Process in der Wait-Liste und braucht keine Rechenzeit.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      ticks (_R_D0)
        Unterbrechungszeit in Ticks (50tel-Sekunden)

AMIGA FUNKTION
      VOID Delay(ULONG ticks);
```

1.92 pdos.library/pOS_CreateProcessA()

```
PROTOTYP
      pOS_Process *res = pOS_CreateProcessA
```

```
        (
          pOS_DosBase *dosbase,
          const pOS_TagItem *tags
        );

FUNKTION
      Erzeugen eines neuen Processes.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      tags (_R_A0) (enum pOS_DosTags)
        DOSTAG_Name          : Name des neuen Processes
                               (Default Background Process)
        DOSTAG_StackSize     : Größe des Stacks (Default 8192, mind. 2400)
        DOSTAG_Priority      : Priorität
                               (Default 0; von -128 bis 127 möglich)
        DOSTAG_SegList       : Zeiger auf Segment-Liste (Default NULL)
        DOSTAG_Entry         : Zeiger auf die Startfunktion des neuen
                               Processes (Default NULL)
        DOSTAG_Data1         : Benutzerdaten, die in der Start-Message
                               des Processes stehen (Default 0)
        DOSTAG_Arguments     : Argumentzeile (Default NULL)
        DOSTAG_ProcNum       : Nummer des neuen Processes oder -2 für
                               die nächste freie Nummer (Default -2)
        DOSTAG_ProcFlags     : (enum pOS_ProcessFlags; Default 0)

          PROCF_FreeSegList   - pr_SegList wird beim Ende freigegeben
          PROCF_FreeCurrDir   - pr_CurrentDir wird an Ende freigegeben
          PROCF_FreeCIS       - pr_CIS wird freigegeben
          PROCF_FreeCOS       - pr_COS wird freigegeben
          PROCF_FreeCES       - pr_CES wird freigegeben
          PROCF_FreeConsoleFH - pr_ConsoleFH
          PROCF_FreeProgDir   - pr_ProgDirLock
          PROCF_FreeProcNum   - pr_TaskNum

        DOSTAG_Synchronous   : Ablaufmodus synchron (Default FALSE)
        DOSTAG_InputFH       : FileHandle auf den Eingabestrom (Default NULL)
        DOSTAG_OutputFH      : FileHandle auf den Ausgabestrom (Default NULL)
        DOSTAG_ErrorFH       : FileHandle auf den Fehlerausgabestrom
                               (Default NULL)
        DOSTAG_ConsoleFH     : FileLock auf Console (Default NULL)
        DOSTAG_CurrDirLock   : FileLock auf das aktuelle Verzeichnis
                               (Default NULL)
        DOSTAG_DupFHs        : alle FileHandle des aufrufenden Processes
                               werden an den neuen Process übertragen
                               (Default FALSE)
        DOSTAG_DefaultProc   : für CIS, COS, CES, CurrentDir werden
                               Defaults verwendet (Default FALSE)
        DOSTAG_DupPath       : (BOOL) Default: FALSE, alle PATH-Daten
                               übernehmen
        DOSTAG_DupAlias      : (BOOL) Default: FALSE, alle ALIAS-Daten
                               übernehmen
        DOSTAG_DupVars       : (BOOL) Default: FALSE, alle VARIABLE-Daten
                               übernehmen
        DOSTAG_CrtHomeShell  : (BOOL) Default: FALSE, create pr_HomeShell
        DOSTAG_ProgDirLock   : (pOS_FileLock*) ProgDir-Lock setzen
```

```
        ACHTUNG: Entweder DOSTAG_SegList oder DOSTAG_Entry muß gesetzt sein.

ERGEBNIS
      res (_R_D0)
        Zeiger auf die Process-Struktur des neuen Processes.
        NULL im Fehlerfall, dann gibt pOS_GetIoErr() die Fehlerursache
        zurück.

SIEHE AUCH
      pOS_LoadSegmentA(), pOS_CreateProc()

AMIGA FUNKTION
      struct Process *CreateNewProc(struct TagItem *tags);
```

1.93 pdos.library/pOS_RunCommand()

```
PROTOTYP
      SLONG res = pOS_RunCommand
        (
          pOS_DosBase *dosbase,
          pOS_SegmentLst *seglist,
          size_t stacksize,
          pOS_FileHandle *cin,
          pOS_FileHandle *cout,
          pOS_FileHandle *cerr,
          const CHAR *arguments
        );

FUNKTION
      Synchrones ausführen eines Kommandos.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      seglist (_R_A0)
        Zeiger auf die Segmentliste des Kommandos
      stacksize (_R_D0)
        Größe des Stack für das Kommando
      cin (_R_A1)
        Eingabestrom für das Kommando oder NULL
      cout (_R_A2)
        Ausgabestrom für das Kommando oder NULL
      cerr (_R_A3)
        Fehlerausgabestrom für das Kommando oder NULL
      arguments (_R_A4)
        Argumentzeile die an das Programm übergeben wird oder NULL

ERGEBNIS
      res (_R_D0)
        Rückgabewert des ausgeführten Programms oder pOS_DOSERR im
        Fehlerfall. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
```

```
      pOS_SystemA()

AMIGA FUNKTION
      LONG RunCommand(BPTR seglist, ULONG stacksize, STRPTR command,
        ULONG commandsize);
```

1.94 pdos.library/pOS_SetShellFail()

```
PROTOTYP
      ULONG res = pOS_SetShellFail
        (
          pOS_DosBase *dosbase,
          ULONG error
        );

FUNKTION
      Setzen eines Programmfehlercodes für die aufrufende
      Shell-Umgebung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      error (_R_D0)
        Fehlercode für die Shell-Umgebung. Definiert sind
          DOSFAIL_OK    ( 0) : kein Fehler
          DOSFAIL_WARN  ( 5) : Warnung
          DOSFAIL_ABORT ( 8) : Abbruch durch CTRL-C
          DOSFAIL_ERROR (10) : einfacher Fehler
          DOSFAIL_FAIL  (20) : schwerer Fehler
        Die Programme können auch eigene, größere Fehlercodes
        für spezifische Fehlerursachen zurückgeben.

ERGEBNIS
      res (_R_D0)
        Bisheriger Programmfehlercode.

HINWEIS
      Da es mit der exit()-Funktion der meisten Comiler Probleme
      gibt und auch der Rückgabewert der main()-Routine nicht gesetzt
      wird, kann mittels dieser Funktion ein Fehlercodes für die
      Shell als Rückgabewert gesetzt werden.

SIEHE AUCH
      pOS_SetIoErr(), pOS_ExitProcess()

AMIGA FUNKTION
```

1.95 pdos.library/pOS_DeleteProcess()

```
PROTOTYP
```

```
      VOID pOS_DeleteProcess
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_Process *process,
          pOS_Task *signaltask,
          ULONG signal
        );

FUNKTION
      Entfernen eines Processes aus der Task-Liste des Systems.
      Nach dem Entfernen kann einem anderen Task/Process ein Signal
      als Bestätigung geschickt werden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      process (_R_A0)
        Zeiger auf den zu entfernenden Task
      signaltask (_R_A1)
        Zeiger auf den Task/Process, der nach dem Entfernen das definierte
        Signal bekommt oder NULL
      signal (_R_D0)
        Signal, das an signaltask, nach dem Entfernen, geschickt werden
        soll.

HINWEIS
      Nach dem Funktionsaufruf darf nicht mehr auf den Zeiger
      process zurückgegriffen werden.

SIEHE AUCH
      pexec.library/pOS_DeleteTask()

AMIGA FUNKTION
```

1.96 pdos.library/pOS_SystemA()

```
PROTOTYP
      SLONG res = pOS_SystemA
        (
          pOS_DosBase *dosbase,
          const CHAR *command,
          const pOS_TagItem *tags
        );

FUNKTION
      Ausführen eines Kommandos (wahlweise asynchron).

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      command (_R_A0)
        Kommando-String der ausgeführt werden soll.
      tags (R_A1) (enum pOS_DosTags)
        DOSTAG_Synchronous  : (Default FALSE)
```

```
          DOSTAG_StackSize    : (Default 8192)
          DOSTAG_Priority,    : (Default 0)
          DOSTAG_Data1        : (Default 0)
          DOSTAG_ProcNum      : (Default -1)
          DOSTAG_ProcFlags    : (Default 0)
          DOSTAG_CurrDirLock  : (Default NULL)
          DOSTAG_InputFH      : (Default NULL)
          DOSTAG_OutputFH     : (Default NULL)
          DOSTAG_ErrorFH      : (Default NULL)
          DOSTAG_ConsoleFH    : (Default NULL)
          DOSTAG_Script       : (Default FALSE)
          DOSTAG_ScriptFile   : (Default NULL)
          DOSTAG_ScriptSnoop  : (Default FALSE)
          DOSTAG_Arguments    : (Default NULL)
          DOSTAG_DupFHs       : (Default FALSE)
          DOSTAG_Name         : (Default NULL)
          DOSTAG_DupPath      : (BOOL) Default: FALSE, alle PATH-Daten
                                übernehmen
          DOSTAG_DupAlias     : (BOOL) Default: FALSE, alle ALIAS-Daten
                                übernehmen
          DOSTAG_DupVars      : (BOOL) Default: FALSE, alle VARIABLE-Daten
                                übernehmen
          DOSTAG_CrtHomeShell : (BOOL) Default: FALSE, create pr_HomeShell
          DOSTAG_ProgDirLock  : (pOS_FileLock*)

  ERGEBNIS
        res (_R_D0)
          pOS_DOSERR im Fehlerfall, dann liefert pOS_GetIoErr() die
          Fehlerursache. Sonst der Returnwert des synchronen Kommandos
          oder 0 bei einer asynchronen Kommandoausführung.

  SIEHE AUCH
        pOS_CreateProcessA(), pOS_RunCommand()

  AMIGA FUNKTION
            LONG SystemTagList(STRPTR command, struct TagItem *tags);

        /*****************************************************************************/ ←
```

1.97 pdos.library/pOS_AddSegment()

```
  PROTOTYP
        BOOL res = pOS_AddSegment
          (
            pOS_DosBase *dosbase,
            pOS_SegmentLst *segmentlist
          );

  FUNKTION
        Einfügen einer Segmentliste in die Dos-Segment-Liste.

  PARAMETER
        dosbase (_R_LB)
```

```
        Zeiger auf pOS_DosLibrary
      segmentlist (_R_A0)
        Zeiger auf die einzufügende Segmentliste

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Segmente eingefügt werden konnten, sonst FALSE.

HINWEIS
      segmentlist->sel_Node.ln_Name enthält den Namen des Segments
      der bei pOS_LoadSegmentA() angegeben wurde

SIEHE AUCH
      pOS_RemSegment(), pOS_OpenSegment()

AMIGA FUNKTION
      BOOL AddSegment(STRPTR name, BPTR seglist, LONG type);
```

1.98 pdos.library/pOS_RemSegment()

```
PROTOTYP
      BOOL res = pOS_RemSegment
        (
          pOS_DosBase *dosbase,
          pOS_SegmentLst *segmentlist
        );

FUNKTION
      Entfernen einer Segmentliste aus der Dos-Segment-Liste.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      segmentlist (_R_A0)
        Zeiger auf die Segmentliste, die entfernt werden soll

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Segmente aus der Liste entfernt wurden,
        sonst FALSE (wenn ein Segment noch in Benutzung ist)

SIEHE AUCH
      pOS_AddSegment()

AMIGA FUNKTION
      BOOL RemSegment(struct Segment *segment)
```

1.99 pdos.library/pOS_OpenSegment()

```
PROTOTYP
```

```
      __ARID__ pOS_SegmentLst *res = pOS_OpenSegment
        (
          pOS_DosBase *dosbase,
          _R_A0 const CHAR *segmentname,
          _R_D0 ULONG reserved
        );

FUNKTION
      Öffnen einer Segmentliste zur Benutzung anhand des Namens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      segmentname (_R_A0)
        Name des zu benutzenden Segements.
      reserved (_R_D0)
        für zukünftige Erweiterungen reserviert, immer auf 0 setzen

ERGEBNIS
      res (_R_D0)
        Zeiger auf die Segmentliste oder NULL wenn es nicht vorhanden ist.

HINWEIS
      Die Segmentliste muß bereits in der Dos-Segment-Liste vorhanden sein.

SIEHE AUCH
      pOS_CloseSegment(), pOS_AddSegment()

AMIGA FUNKTION
      ähnlich struct Segment *FindSegment(STRPTR segmentname, struct Segment *,  ↩
         LONG);
```

1.100 pdos.library/pOS_CloseSegment()

```
PROTOTYP
      VOID pOS_CloseSegment
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_SegmentLst *segmentlist
        );

FUNKTION
      Schliesen einer Segmentliste nach der Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      segmentlist (_R_A0)
        Zeiger auf die Segmentlite die geschlossen werden soll
        ACHTUNG: Auf den Zeiger darf nach dem Aufruf der Funktion
                 nicht mehr zugegriffen werden.

SIEHE AUCH
      pOS_OpenSegment(), pOS_RemSegment()
```

```
AMIGA FUNKTION
```

1.101 pdos.library/pOS_LoadSegmentA()

```
PROTOTYP
      __ARID__ pOS_SegmentLst *res = pOS_LoadSegmentA
        (
          _R_LB pOS_DosBase *dosbase,
          _R_A0 const dosname_t *filename,
          _R_A1 const pOS_TagItem *tags
        );

FUNKTION
      Laden eines ausführbaren Programmes in den Speicher.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filename (_R_A0)
        Dateiname des ausführbaren Programmes
      tags (_R_A1)
        für zukünftige Erweiterungen, immer NULL angeben

ERGEBNIS
      res (_R_D0)
        Zeiger auf die geladene Sementliste im Speicher, die mit
        pOS_UnloadSegment() wieder entfernt werden muß oder
        NULL im Fehlerfall. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_UnloadSegment()

AMIGA FUNKTION
      BPTR LoadSeg(STRPTR filename);
```

1.102 pdos.library/pOS_UnloadSegment()

```
PROTOTYP
      VOID pOS_UnloadSegment
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_SegmentLst *segmentlist
        );

FUNKTION
      Entfernen einer Segmentliste aus dem Speicher.

PARAMETER
      dosbase (_R_LB)
```

```
        Zeiger auf pOS_DosLibrary
      segmentlist (_R_A0)
        Zeiger auf die Segmentliste die aus dem Speicher entfernt werden soll
        ACHTUNG: Auf den Zeiger darf nach dem Aufruf der Funktion
                 nicht mehr zugegriffen werden.

SIEHE AUCH
      pOS_LoadSegmentA()

AMIGA FUNKTION
      LONG UnLoadSeg(BPTR segmentlist);
```

1.103 pdos.library/pOS_InternalLoadSegment()

```
PROTOTYP
      BOOL res = pOS_InternalLoadSegment
        (
          pOS_DosBase *dosbase,
          pOS_ISegmentData *segmentdata
        );

FUNKTION
      Laden eines ausführbaren Programmes in den Speicher,
      normal nur zur internen Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      segmentdata (_R_A0)
        Zeiger auf die Segment-Daten

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Segmente geladen wurden. Diese müssen
        mit pOS_InitialUnloadSegment() wieder entfernt werden.
        Sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_InternalUnloadSegment(), pOS_LoadSegmentA()

AMIGA FUNKTION
      BPTR InternalLoadSeg(BPTR fh, BPTR table, LONG *functions, LONG *stack);
```

1.104 pdos.library/pOS_InternalUnloadSegment()

```
PROTOTYP
      BOOL res = pOS_InternalUnloadSegment
        (
          pOS_DosBase *dosbase,
          pOS_ISegmentData *segmentdata
```

```
        );

FUNKTION
      Entfernen einer mittels pOS_InternalLoadSegment()
      geladenen Segmentliste; normal nur zur internen Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      segmentdata (_R_A0)
        Zeiger auf die Segment-Daten

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Segmente aus dem Speicher entfernt wurden.

SIEHE AUCH
      pOS_InternalLoadSegment();

AMIGA FUNKTION
          LONG InternalUnLoadSeg(BPTR segmentlist, APTR function);

      /*****************************************************************************/ ←
```

1.105 pdos.library/pOS_InitDosIOReq()

```
PROTOTYP
      pOS_Process *res = pOS_InitDosIOReq
        (
          pOS_DosBase *dosbase,
          const pOS_DosDevice *dosdevice,
          pOS_DosIOReq *dosioreq
        );

FUNKTION
      Installieren einer DosIOReq-Struktur.
      Muß vor der IO-Benutzung aufgerufen werden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger aus das DosDevice
      ioreq (_R_A1)
        Zeiger auf die zu installierende Struktur

ERGEBNIS
      res (_R_D0)
        Zeiger auf den eigenen Process oder NULL im Fehlerfall.

HINWEIS
      Alle Daten aus dosdevice->ddv_IOMaster werden nach dosioreq
      übertragen.
```

```
SIEHE AUCH

AMIGA FUNKTION
```

1.106 pdos.library/pOS_InitDosDevice()

```
PROTOTYP
      VOID pOS_InitDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice
        );

FUNKTION
      Installieren einer DosDevice-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger aus die DosDevice-Struktur, das installiert werden soll

SIEHE AUCH

AMIGA FUNKTION
```

1.107 pdos.library/pOS_AddDosDevice()

```
PROTOTYP
      VOID pOS_AddDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice
        );

FUNKTION
      Einfügen eines DosDevices in die pOS-DosDevice-Liste

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, das eingefügt werden soll

SIEHE AUCH
      pOS_RemDosDevice(), pOS_OpenDosDevice()

AMIGA FUNKTION
```

1.108 pdos.library/pOS_RemDosDevice()

```
PROTOTYP
      VOID pOS_RemDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice
        );

FUNKTION
      Entfernen eines DosDevices aus der pOS-DosDevice-Liste

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, das entfernt werden soll

SIEHE AUCH
      pOS_AddDosDevice(), pOS_CloseDosDevice()

AMIGA FUNKTION
```

1.109 pdos.library/pOS_OpenDosDevice()

```
PROTOTYP
      BOOL res = pOS_OpenDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevPathInfo *dosdevpathinfo
        );

FUNKTION
      Öffnen eines DosDevices zur Benutzung. Evtl. wird dabei das
      Gerät aus einer Mountliste nachgeladen.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevpathinfo (_R_A0)
        Daten auf das zu öffnende DosDevice
        dosdevpathinfo->dopi_CurrDir
          NULL für pr_CurrentDir, sonst kann dopi_PathName relativ zu
          diesem Verzeichnis erfolgen
        dosdevpathinfo->dopi_PathName:
          Name des Device/Verzeichnis/Datei
        Die Struktur darf bis zum pOS_CloseDosDevice() nicht verändert werden.

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Gerät benutzt werden kann, dann muß mittels
        pOS_CloseDosDevice() das Gerät wieder freigegeben werden.
```

```
        Sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Zur direkten Kommunikation mit einem Handler.

SIEHE AUCH
      pOS_CloseDosDevice(), pOS_AddDosDevice()

AMIGA FUNKTION
```

1.110 pdos.library/pOS_CloseDosDevice()

```
PROTOTYP
      VOID pOS_CloseDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevPathInfo *dosdevpathinfo
        );

FUNKTION
      Freigeben eines DosDevices nach der Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevpathinfo (_R_A0)
        Zeiger der an pOS_OpenDosDevice() übergeben und
        dessen Daten nicht verändert wurden.

SIEHE AUCH
      pOS_OpenDosDevice(), pOS_RemDosDevice()

AMIGA FUNKTION
```

1.111 pdos.library/pOS_AddDosDefDevice()

```
PROTOTYP
      VOID pOS_AddDosDefDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDefDevice *dosdefdevice
        );

FUNKTION
      Installieren eines DosDefaultDevices.
      Wemm ein Paket keinem der derzeit gültigen DosDevices
      zugeordnet werden kann, wird es an die DosDefaultDevices
      weitergegeben.

PARAMETER
```

```
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdefdevice (_R_A0)
        Zeiger auf pOS_DosDefDevice-Struktur

SIEHE AUCH
      pOS_RemDosDefDevice()

AMIGA FUNKTION
```

1.112 pdos.library/pOS_RemDosDefDevice()

```
PROTOTYP
      VOID pOS_RemDosDefDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDefDevice *dosdefdevice
        );

FUNKTION
      Entfernt ein DosDefaultDevice.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdefdevice (_R_A0)
        Zeiger auf pOS_DosDefDevice-Struktur.

SIEHE AUCH
      pOS_AddDosDefDevice()

AMIGA FUNKTION
```

1.113 pdos.library/pOS_CreateDosAssign()

```
PROTOTYP
      pOS_DosDevice *res = pOS_CreateDosAssign
        (
          pOS_DosBase *dosbase,
          const CHAR *assignName,
          __ARID__ pOS_FileLock *filelock,
          const dosname_t *filename,
          ULONG type
        );

FUNKTION
      Erzeugen eines Assigns.

PARAMETER
      dosbase (_R_LB)
```

```
        Zeiger auf pOS_DosLibrary
      assignname (_R_A0)
        Name des Assigns ohne Doppelpunkt
        existiert der Assign bereits, liefert pOS_GetIoErr()
        den Fehlercode DOSERR_ObjectExists.
      filelock (_R_A1)
        FileLock auf das Verzeichnis, dann darf der filelock
        nicht mehr benutzt werden oder NULL
      filename (_R_A2)
        Name des Verzeichnisses oder NULL
      type (_R_D0) (enum pOS_DosDeviceTypes)
        DDTYP_Assign: normalen, sofortigen Assign erzeugen
          filelock oder filename kann benutzt werden
        DDTYP_LateAssign: Assign steht erst nach der ersten Benutzung
                          und wird dabei in einen normalen Assign gewandelt
          filename muß verwendet werden
        DDTYP_NonBindingAssign: Pfadalias der nicht über einen Lock aufgelöst ↩
           wird
          filename muß verwendet werden

ERGEBNIS
      res (_R_D0)
        Zeiger auf das DosDevice des Assigns oder NULL.
        Dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Wurde ein filelock angegeben und die Funktion schlug fehlt,
      so ist der filelock selber freizugeben.
      Ansonsten darf auf diesen filelock nicht mehr zugegriffen
      werden und auch keine Freigabe erfolgen.

SIEHE AUCH
      pOS_DeleteDosAssign(), pOS_LockObject(), pOS_UnlockObject()

AMIGA FUNKTION
      LONG AssignLock(STRPTR assignname, BPTR filelock);
```

1.114 pdos.library/pOS_CreateDOSTemplate()

```
PROTOTYP
      pOS_DosDevice *res = pOS_CreateDOSTemplate
        (
          pOS_DosBase *dosbase,
          const CHAR *template
        );

FUNKTION
      interne Funktion

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      template (_R_A0)
```

```
ERGEBNIS
      res (_R_D0)

SIEHE AUCH

AMIGA FUNKTION
```

1.115 pdos.library/pOS_CreateDOSVolume()

```
PROTOTYP
      pOS_DosDevice *res = pOS_CreateDOSVolume
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice,
          const CHAR *volumename,
          const pOS_DateStamp *datestamp
        );

FUNKTION
      Erzeugen eines neuen Datenträgers.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, der den Datenträger enthält
      volumename (_R_A1)
        Name des Datenträgers, der erzeugt werden soll
      datestamp (_R_A2)
        Erzeugungsdatum und Uhrzeit
        NULL für das aktuelle Datum/Zeit

ERGEBNIS
      res (_R_D0)
        Zeiger auf das erzeugte DosDevice oder NULL.
        Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_OpenDOSVolume()

AMIGA FUNKTION
```

1.116 pdos.library/pOS_OpenDOSVolume()

```
PROTOTYP
      pOS_DosDevice *res = pOS_OpenDOSVolume
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice,
          const CHAR *volumename,
```

```
          const pOS_DateStamp *datestamp
        );

FUNKTION
      Öffnen eines Datenträgers zur Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, der den Datenträger enthält
      volumename (_R_A1)
        Name des zu öffnenden Datenträgers
      datestamp (_R_A2)
        Erstellungsdatum/zeit des zu öffnenden Datenträgers

ERGEBNIS
      res (_R_D0)
        Zeiger auf das geöffnete DosDevice das mittels
        pOS_CloseDOSVolume() wieder geschlossen werden muß
        oder NULL, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_CloseDOSVolume(), pOS_CreateDOSVolume()

AMIGA FUNKTION
```

1.117 pdos.library/pOS_CloseDOSVolume()

```
PROTOTYP
      VOID pOS_CloseDOSVolume
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice
        );

FUNKTION
      Schliesen eines Datenträgers nach der Benutzung.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, der den Datenträger verwaltet

SIEHE AUCH
      pOS_OpenDOSVolume()

AMIGA FUNKTION
```

1.118 pdos.library/pOS_GetDosDevice()

```
PROTOTYP
      pOS_DosDevice *res = pOS_GetDosDevice
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          const dosname_t *pathname
        );

FUNKTION
      Ermitteln des verwaltenden DosDevices.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        FileLock auf das aktuelle Verzeichnis, dann kann pathname
        relativ dazu erfolgen oder NULL für pr_CurrentDir
      pathname (_R_A1)
        Dateiname dessen DosDevice ermittelt werden soll

ERGEBNIS
      res (_R_D0)
        Zeiger auf das ermittelte DosDevice
        oder NULL, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_OpenDosDevice(), pOS_GetDosHandler()

AMIGA FUNKTION
```

1.119 pdos.library/pOS_GetDosHandler()

```
PROTOTYP
      pOS_DosDevice *res = pOS_GetDosHandler
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          const dosname_t *pathname
        );

FUNKTION
      Ermitteln des verwaltenden Handlers.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        FileLock auf das aktuelle Verzeichnis, dann kann pathname
        relativ dazu erfolgen oder NULL für pr_CurrentDir
      pathname (_R_A1)
        Dateiname dessen Handler ermittelt werden soll
```

```
ERGEBNIS
      res (_R_D0)
        Zeiger auf den ermittelten Handler (DosDevice)
        oder NULL, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Ist das DosDevice kein DDTYP_Volume, dann liefert diese
      Funktion das selbe Ergebnis wie pOS_GetDosDevice().

SIEHE AUCH
      pOS_GetDosDevice(), pOS_OpenDosDevice()

AMIGA FUNKTION
```

1.120 pdos.library/pOS_AddDosDeviceBuffers()

```
PROTOTYP
      dossize_t res = pOS_AddDosDeviceBuffers
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice,
          SLONG size
        );

FUNKTION
      Vergrößern/verkleinern eines Device-Puffers

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice
      size (_R_D0)
        neue Puffergröße

ERGEBNIS
      res (_R_D0)
        alte Puffergröße oder pOS_DOSERR im Fehlerfall.
        Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH

AMIGA FUNKTION
```

1.121 pdos.library/pOS_InhibitDosDevice()

```
PROTOTYP
      BOOL res = pOS_InhibitDosDevice
        (
          pOS_DosBase *dosbase,
```

```
          pOS_DosDevice *dosdevice,
          ULONG status
        );

FUNKTION
      Sperren eines Laufwerkes gegen Zugriffe
      (z.B. beim Formatieren)

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice das gesperrt/entsperrt werden soll
      status (_R_D0)
        TRUE zum Sperren gegen Zugriffe
        FALSE zum Aufheben der Zugriffssperre

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Gerät gesperrt/entsperrt wurde.
        Sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH

AMIGA FUNKTION
      LONG Inhibit(STRPTR name, LONG status);
```

1.122 pdos.library/pOS_RelabelDosDevice()

```
PROTOTYP
      BOOL res = pOS_RelabelDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevice *dosdevice,
          const CHAR *volumename
        );

FUNKTION
      Umbenennen eines Datenträgers.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevice (_R_A0)
        Zeiger auf DosDevice, dessen Datenträgername geändert werden soll
      volumename (_R_A1)
        Neuer Name für den Datenträger

ERGEBNIS
      res (_R_D0)
        TRUE wenn der neue Name gesetzt wurde,
        sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
```

```
AMIGA FUNKTION
      LONG Relabel(STRPTR devicename, STRPTR volumename);
```

1.123 pdos.library/pOS_GetDosDeviceName()

```
PROTOTYP
      pOS_DosDevice *res = pOS_GetDosDeviceName
        (
          pOS_DosBase *dosbase,
          const CHAR *volumename,
          ULONG type
        );

FUNKTION
      Ermitteln eines DosDevices anhand eines Volume/Verzeichnis/Dateinamens.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      volumename (_R_A0)
        Name des Datenträgers der gesucht wird
      type (_R_D0) (enum pOS_DosDeviceTypes)
        Type des gesuchten Gerätes
        (DDTYP_Handler, DDTYP_Assign, DDTYP_Volume, DDTYP_LateAssign,
         DDTYP_NonBindingAssign, DDTYP_Foreign)

ERGEBNIS
      res (_R_D0)
        Zeiger auf das gefundene DosDevice oder NULL
        wenn das DosDevice nicht vorhanden ist.

SIEHE AUCH
      pOS_GetDosMountName(), pOS_GetDosHandler()

AMIGA FUNKTION
```

1.124 pdos.library/pOS_GetDosMountName()

```
PROTOTYP
      pOS_DosMountDevice *res = pOS_GetDosMountName
        (
          pOS_DosBase *dosbase,
          const CHAR *pathname
        );

FUNKTION
      Ermitteln eines DosMountDevices anhand eines Volume/Verzeichnis/ ↩
         Dateinamens.
```

```
PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      pathname (_R_A0)
        Volume/Verzeichnis/Dateiname, dessen DosMountDevice
        ermittelt werden soll.

ERGEBNIS
      res (_R_D0)
        Zeiger auf das gefundene DosMountDevice oder NULL
        wenn das DosMountDevice nicht vorhanden ist.

SIEHE AUCH
      pOS_GetDosDeviceName(), pOS_GetDosHandler()

AMIGA FUNKTION
```

1.125 pdos.library/pOS_CreateDosDevFromMount()

```
PROTOTYP
      pOS_DosDevice *res = pOS_CreateDosDevFromMount
        (
          pOS_DosBase *dosbase,
          const pOS_DosMountDevice *dosmountdevice
        );

FUNKTION
      Mounten eines Gerätes mit evtl. nachladen von Disk.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosmountdevice (_R_A0)
        Zeiger auf die Mount-Daten (z.B. von pOS_CreateDosMount())

ERGEBNIS
      res (_R_D0)
        Zeiger auf das gemountetes DosDevice
        oder NULL, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_CreateDosMount()

AMIGA FUNKTION
```

1.126 pdos.library/pOS_DeleteDosAssign()

```
PROTOTYP
      BOOL res = pOS_DeleteDosAssign
        (
```

```
          pOS_DosBase *dosbase,
          const CHAR *assignname,
          const pOS_FileLock *filelock,
          ULONG reserved
        );

FUNKTION
      Entfernen eines Assigns, bzw. eines Teil-Assigns bei
      einem Multi-Assign.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      assignname (_R_A0)
        Name des Assigns ohne Doppelpunkt
      filelock (_R_A1)
        FileLock auf das zu entfernende Verzeichnis
        (bei MultiAssigns einzeln aufhebbar)
        sonst NULL, dann wird der Assign vollständig aufgehoben
      reserved (_R_D0)
        für zukünftige Erweiterungen reserviert, immer 0 angeben

ERGEBNIS
      res (_R_D0)
        TRUE wenn der Assign entfernt werden konnte sonst FALSE
        (wenn der Assign intern gelockt ist oder nicht existiert).

SIEHE AUCH
      pOS_CreateDosAssign(), pOS_LockObject(), pOS_UnlockObject()

AMIGA FUNKTION
      LONG AssignLock(STRPTR assignname, BPTR filelock);
      LONG RemAssignList(STRPTR assignname, BPTR filelock);
```

1.127 pdos.library/pOS_AddDosAssign()

```
PROTOTYP
      BOOL res = pOS_AddDosAssign
        (
          pOS_DosBase *dosbase,
          const CHAR *assignname,
          __ARID__ pOS_FileLock *filelock,
          ULONG reserved
        );

FUNKTION
      Hinzufügen eines Verzeichnis zu einem (Multi-)Assign

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      assignname (_R_A0)
        Name des Assigns ohne Doppelpunkt
      filelock (_R_A1)
```

```
        FileLock auf das hinzuzufügende Verzeichnis
      reserved (_R_D0)
        für zukünftige Erweiterungen reserviert, immer 0 angeben

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Verzeichnis dem Assign hinzugefügt wurde.
        Sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Exisitert noch kein Assign mit diesem Namen, so wird er wie
      bei pOS_CreateDosAssign() erzeugt.
      Schlug die Funktion fehl, so ist der filelock wieder freizugeben.
      Ansonsten darf darauf nicht mehr zugegriffen werden.

SIEHE AUCH
      pOS_CreateDosAssign(), pOS_DeleteDosAssign(),
      pOS_LockObject(), pOS_UnlockObject()

AMIGA FUNKTION
      LONG AssignAdd(STRPTR assignname, BPTR lock);
```

1.128 pdos.library/pOS_GetNextDosDevice()

```
PROTOTYP
      BOOL res = pOS_GetNextDosDevice
        (
          pOS_DosBase *dosbase,
          pOS_DosDevPathInfo *dosdevpathinfo
        );

FUNKTION
      Ermitteln des nächsten Eintrages.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosdevpathinfo (_R_A0)
        Zeiger auf das geöffnende DosDevice

ERGEBNIS
      res (_R_D0)
        TRUE wenn ein weiterer Eintrag gefunden wurde,
        FALSE wenn kein weiterer Eintrag vorhanden ist.

HINWEIS
      Zur Zeit wird der nächste Assign eines Multi-Assigns
      ermittelt.

SIEHE AUCH
      pOS_OpenDosDevice()

AMIGA FUNKTION
```

1.129 pdos.library/pOS_CreateDosArgs()

```
PROTOTYP
      __ARID__ pOS_DosArgs *res = pOS_CreateDosArgs
        (
          pOS_DosBase *dosbase
        );

FUNKTION
      Reservieren und installieren einer DosArgs-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary

ERGEBNIS
      res (_R_D0)
        Zeiger auf die reservierte und installierte DosArgs-Struktur,
        die mittels pOS_DeleteDosArgs() wieder freigegeben werden muß.
        NULL im Fehlerfall bei Speichermangel.

SIEHE AUCH
      pOS_DeleteDosArgs()

AMIGA FUNKTION
```

1.130 pdos.library/pOS_DeleteDosArgs()

```
PROTOTYP
      VOID pOS_DeleteDosArgs
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_DosArgs *dosargs
        );

FUNKTION
      Freigeben des Speichers einer DosArgs-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf die von pOS_CreateDosArgs() gelieferte DosArgs-
        Struktur, die freigegeben werden soll.
        ACHTUNG: nach der Funktion darf auf dosargs nicht mehr
                 zugegriffen werden.

SIEHE AUCH
      pOS_CreateDosArgs(), pOS_ReadDosArgsA()

AMIGA FUNKTION
      VOID FreeArgs(struct RDArgs *dosargs);
```

1.131 pdos.library/pOS_CreateDosTokenString()

```
PROTOTYP
      const CHAR *res = pOS_CreateDosTokenString
        (
          pOS_DosBase *dosbase,
          pOS_DosArgs *dosargs,
          const CHAR *string
        );

FUNKTION
      Umwandeln eines Strings in einen Token-String.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf die DosArgs
      string (_R_A1)
        der zu konvertierende String

ERGEBNIS
      res (_R_D0)
        Zeiger auf den konvertierten String
        oder NULL im Fehlerfall bei Speichermangel.

HINWEIS
      Es werden folgende Kombinationen konvertiert:
      - *A oder *a  => Alarmsignal     (BEL)
      - *B oder *b  => Backspace       (BS)
      - *E oder *e  => Escape          (ESC)
      - *F oder *f  => Formfeed        (FF)
      - *N oder *n  => Linefeed        (LF)
      - *T oder *t  => Tabulator       (HT)
      - *R oder *r  => Carriage Return (CR)

      Spezielle-Steuerzeichen:
      ------------------------
      - *%      => es wird kein Zeichen ausgegeben, dient der
                   Trennung von Sequenzen
      - *x00    => Hex-Wert wird in Ascii gewandelt
      - *000    => Oct-Wert wird in Ascii gewandelt


SIEHE AUCH
      pOS_CreateDosArgs()

AMIGA FUNKTION
```

1.132 pdos.library/pOS_CreateDosTokenList()

```
PROTOTYP
```

```
      pOS_DosErrors res = pOS_CreateDosTokenList
        (
          pOS_DosBase *dosbase,
          pOS_DosArgs *dosargs
        );

FUNKTION
      Zerlegen der Befehlsschablone in die interne Tokenform.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf die DosArgs

ERGEBNIS
      res (_R_D0)
        DOSERR_None wenn alles ok war,
        sonst der spezifische Fehlercode.

SIEHE AUCH
      pOS_CreateDosArgs()

AMIGA FUNKTION
```

1.133 pdos.library/pOS_InterpretDosTokenList()

```
PROTOTYP
      pOS_DosErrors res = pOS_InterpretDosTokenList
        (
          pOS_DosBase *dosbase,
          pOS_DosArgs *dosargs
        );

FUNKTION
      Auswerten der Befehlsschablone und zuweisen der Argumente.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf die DosArgs, die bereits durch pOS_CreateDosTokenList()
        bearbeitet wurden

ERGEBNIS
      res (_R_D0)
        DOSERR_None wenn alles ok war.
        Sonst der spezifische Fehlercode.

SIEHE AUCH
      pOS_CreateDosArgs(), pOS_CreateDosTokenList()

AMIGA FUNKTION
```

1.134 pdos.library/pOS_ReadDosArgsA()

```
PROTOTYP
      __ARID__ pOS_DosArgs *res = pOS_ReadDosArgsA
        (
          pOS_DosBase *dosbase,
          const CHAR *template,
          ULONG *results,
          size_t numresult,
          const pOS_TagItem *tags
        );

FUNKTION
      Auswerten der Kommandozeilenargumente.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      template (_R_A0)
        Zeiger auf die Befehlsschablone, die das Format der
        Eingabe vorgibt; es besteht aus einer Folge von Optionen
        ohne Leerzeichen, die durch Kommata getrennt sind und
        durch Bezeichner ergänzt werden können;
        Mögliche Bezeichner sind:
        /S für eine Schalteroption; das zugehörige Element im
               Feld results (Datentyp: ULONG) ist eine Variable, die
               ungleich FALSE ist, wenn die Option auftritt;
        /K für eine Schlüsselwortoption; das zugehörige Element
               im Feld results wird nur gefüllt, wenn das
               Schlüsselwort auftritt;
        /N für eine numerische Option; das zugehörige Element im
               Feld results (Datentyp: SLONG *) ist ein Zeiger auf die
               numerische Variable bzw. NULL, falls der Benutzer die
               Option nicht eingegeben hat;
        /T für eine Umschaltoption; das zugehörige Element im
               Feld results (Datentyp: ULONG) ist eine Variable, deren
               logischer Wert (FALSE oder ungleich FALSE) negiert
               wird, wenn die Option auftritt;
               weiterhin sind die Angaben ON, OFF, TOGGLE möglich
        /A für ein Argument, das immer angegeben werden muß;
        /F für eine abschließende Option, die als letztes
               angegeben werden muß; alle folgenden Eingaben werden
               dieser Option zugeordnet;
        /M für eine Option, das zugehörige
               Element im Feld results (Datentyp: CHAR *) ist ein
               Zeiger auf ein Feld von Zeigern auf Zeichenketten, das
               mit einem NULL-Zeiger abgeschlossen wird, bzw. NULL,
               falls der Benutzer die Option nicht eingegeben hat.
        Wird kein Bezeichner (oder nur /A oder /K) angegeben, ist
        das zugehörige Element im Feld results (Datentyp: CHAR *)
        ein Zeiger auf eine Zeichenkette bzw. NULL, falls der
        Benutzer die Option nicht eingegeben hat.
      results (_R_A1)
        Zeiger auf ein initialisiertes Langwort-Feld, das die
        Ergebnisse aufnehmen soll.
```

```
      numresult (_R_D0)
        Anzahl an result-Elementen (normal sizeof(results)/sizeof(ULONG))
      tags (_R_A2)
        ARGTAG_HelpText:      Zeiger auf den Hilfstext (Default NULL)
        ARGTAG_Arguments:     Argumentzeile (Default Shell-Befehlsargumente)
        ARGTAG_SearchLable:   Label nach dem gesucht werden soll
                              (Default NULL = erstes gefundene Label benutzen)
        ARGTAG_Seperator:     Trennzeichen (Default ',')
        ARGTAG_VarFlags:      enum pOS_LocalVarType (Default LOCALVAR_Var)
        ARGTAG_Process:       Process vom den die Variablen benutzt werden ↩
           sollen
                              (Default NULL = aktueller Process)
        ARGTAG_VarName:       Name der Variable (für pOS_GetVar()) (Default NULL ↩
           )
        ARGTAG_RawLine:       TRUE um die Argumentzeile mittel ↩
           pOS_CreateDosTokenString()
                              zu konvertieren (Default FALSE)
        ARGTAG_FileName:      Name der Datei mit den Argumenten (Default NULL)
        ARGTAG_PrgHeaderText: (const CHAR*) z.B. "mein programm"
        ARGTAG_PrgVerText:    (const CHAR*) z.B. "$VER: 1.0 (16.03.96)"
        ARGTAG_Snoop:         (BOOL) default=FALSE, Zuweisungsliste ausgeben
        ARGTAG_FH:            (struct pOS_FileHandle*) Daten über FileHandle ↩
           laden

ERGEBNIS
      res (_R_D0)
        Zeiger auf die erzeugte DosArgs-Struktur, die mittels
        pOS_DeleteDosArgs() freigegeben werden muß.
        Null im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_DeleteDosArgs(), pOS_WriteDosArgsA(), Shell-Con.txt

AMIGA FUNKTION
      struct RDArgs *ReadArgs(STRPTR template, LONG *results, struct RDArgs *);
```

1.135 pdos.library/pOS_PrintDosArgList()

```
PROTOTYP
      VOID pOS_PrintDosArgList
        (
          pOS_DosBase *dosbase,
          pOS_DosArgs *dosargs
        );

FUNKTION
      Ausgeben der Befehlsschablone und der zugewissenen Argumente
      in den Standard-Ausgabestrom.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf DosArgs, die ausgegeben werden sollen
```

```
SIEHE AUCH
      pOS_CreateDosArgs(), pOS_GetStdOutput()

AMIGA FUNKTION
```

1.136 pdos.library/pOS_DosFindToken()

```
PROTOTYP
      const pOS_DosToken *res = pOS_DosFindToken
        (
          pOS_DosBase *dosbase,
          pOS_DosArgs *dosargs,
          const CHAR *tokenname
        );

FUNKTION
      Suchen eines Schlüsselwortes in einer Befehlsschablone.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosargs (_R_A0)
        Zeiger auf DosArgs
      tokenname(_R_A1)
        gesuchtes Schlüsselwort (Templatename oder deren Abkürzung)

ERGEBNIS
      res (_R_D0)
        Zeiger auf den gefundenen DosToken
        oder NULL, wenn er nicht vorhanden ist

SIEHE AUCH
      pOS_CreateDosArgs()

AMIGA FUNKTION
      LONG FindArg(STRPTR template, STRPTR tokenname);
```

1.137 pdos.library/pOS_WriteDosArgsA()

```
PROTOTYP
      BOOL res = pOS_WriteDosArgsA
        (
          pOS_DosBase *dosbase,
          const CHAR *template,
          ULONG *results,
          size_t numresults,
          const pOS_DosArgs *dosargs,
          const pOS_TagItem *tags
        );
```

```
FUNKTION
      Erzeugen einer Datei mit Kommandozeilenargumenten, die mittels
      pOS_ReadDosArgsA() wieder gelesen werden kann.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      template (_R_A0)
        Zeiger auf die Befehlsschablone, wie sie bereits bei
        pOS_ReadDosArgsA() beschrieben wurde; oder NULL.
      results (_R_A1)
        Zeiger auf ein initialisiertes Langwort-Feld, das die
        zu schreibenden Daten enthält; oder NULL.
      numresult (_R_D0)
        Anzahl an result-Elementen (normal sizeof(results)/sizeof(ULONG));
        oder 0.
      dosargs (_R_A2)
        Rückgabewert der Funktion pOS_ReadDosArgsA() mit den geparsten
        Argumenten.
      tags (_R_A3)
        ARGTAG_SearchLable:   Label nach dem gesucht werden soll
                              (Default NULL = erstes gefundene Label benutzen)
        ARGTAG_FileName:      Name der Datei, die bei Bedarf auch erzeugt wird
        ARGTAG_FH:            FileHandle auf die bereits geöffnete Datei

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Daten geschrieben wurden;
        sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

HINWEIS
      Es muß entweder template/results/numresults oder dosargs
      übergeben werden. Somit ist es möglich, eine geparste Datei,
      nach ändern der Daten, sofort wieder zurückzuschreiben.

BEISPIEL
      const CHAR *filename="argsfile";
      const CHAR *template="FILE/A, DIRS/S, FILES/S, SIZE/K/N";
      ULONG args[4]={ 0,0,0,0 };
      enum { _FILE,_DIRS,_FILES,_SIZE,_MAX_ARGS };
      pOS_DosArgs *dosargs;

      if(dorargs = pOS_ReadDosArgs(gb_DosBase,template,args,
                sizeof(args)/sizeof(ULONG),
                /* ... , */
                ARGTAG_FileName, filename,
                TAG_END))
      {
        /* ... veränderungen z.B.
           SLONG zahl=10; args[_SIZE]=&zahl;
           args[_DIRS]=TRUE;
        */

        if(pOS_WriteDosArgs(gb_DosBase,NULL,NULL,0,dosargs,
                            ARGTAG_FileName, filename,
                            TAG_END) == FALSE)
```

```
        {
          pOS_PrintDosErr(gb_DosBase,NULL,filename,0);
          /* durch 0 wird der aktuelle Fehlercode automatisch ermittelt */
        }
        pOS_DeleteDosArgs(dosargs);
      }
      /* es darf kein ELSE-Zweig vorhanden sein !!
         die Ausgabe wird bereits von pOS_ReadDosArgs() behandelt */

SIEHE AUCH
      pOS_ReadDosArgsA()

AMIGA FUNKTION
```

1.138 pdos.library/pOS_CreateParse()

```
PROTOTYP
      __ARID__ pOS_Parse *res = pOS_CreateParse
        (
          pOS_DosBase *dosbase,
          pOS_ParseObject **lowerstack,
          pOS_ParseObject **upperstack
        );

FUNKTION
      Reservieren und installieren einer Parse-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      lowerstack (_R_A0)
      upperstack (_R_A1)

ERGEBNIS
      res (_R_D0)
        Erzeugte Parse-Struktur, die mittels pOS_DeleteParse()
        wieder freigegeben werden muß.
        NULL im Fehlerfall bei Speichermangel.

SIEHE AUCH
      pOS_DeleteParse(), pOS_ConstructParse(), pOS_Parsing()

AMIGA FUNKTION
```

1.139 pdos.library/pOS_DeleteParse()

```
PROTOTYP
      VOID pOS_DeleteParse
        (
          pOS_DosBase *dosbase,
```

```
          __ARID__ pOS_Parse *parse
        );

FUNKTION
      Freigeben einer mittels pOS_CreateParse() erzeugten Parse-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      parse (_R_A0)
        Zeiger auf die Parse-Struktur
        ACHTUNG: nach dieser Funktion darf auf die Struktur nicht
                 mehr zugegriffen werden

SIEHE AUCH
      pOS_DestructParse()

AMIGA FUNKTION
```

1.140 pdos.library/pOS_ConstructParse()

```
PROTOTYP
      VOID pOS_ConstructParse
        (
          pOS_DosBase *dosbase,
          pOS_Parse *parse
        );

FUNKTION
      Installieren einer Parse-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      parse (_R_A0)
        Zeiger auf die zu installierende Parse-Struktur

SIEHE AUCH
      pOS_DestructParse(), pOS_Parsing()

AMIGA FUNKTION
```

1.141 pdos.library/pOS_DestructParse()

```
PROTOTYP
      VOID pOS_DestructParse
        (
          pOS_DosBase *dosbase,
          pOS_Parse *parse
        );
```

```
FUNKTION
      Deinstallieren einer Parse-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      parse (_R_A0)
        Zeiger auf die zu deinstallierende Parse-Struktur

SIEHE AUCH
      pOS_ConstructParse(), pOS_DeleteParse()

AMIGA FUNKTION
```

1.142 pdos.library/pOS_Parsing()

```
PROTOTYP
      pOS_ParseObject *res = pOS_Parsing
        (
          pOS_DosBase *dosbase,
          pOS_Parse *parse,
          ULONG prilevel
        );

FUNKTION

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      parse (_R_A0)
        Zeiger auf die von pOS_CreateParse() erzeugte und von
        pOS_ConstructParse() installierte Parse-Struktur
      prilevel (_R_D0)

ERGEBNIS
      res (_R_D0)
        Zeiger auf das gefundene ParseObject oder NULL
        wenn kein passendes Objekt gefunden wurde.

SIEHE AUCH
      pOS_CreateParse(), pOS_ConstructParse()

AMIGA FUNKTION
```

1.143 pdos.library/pOS_ConstructPattern()

```
PROTOTYP
      VOID pOS_ConstructPattern
        (
```

```
          pOS_DosBase *dosbase,
          pOS_PatternMatching *patternmatching
        );

FUNKTION
      Installieren einer PatternMatching-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      patternmatching (_R_A0)
       Zeiger auf die PatternMatching-Struktur, die installiert werden soll

SIEHE AUCH
      pOS_ParsePatternA(), pOS_MatchPattern(), pOS_DestructPattern()

AMIGA FUNKTION
```

1.144 pdos.library/pOS_DestructPattern()

```
PROTOTYP
      VOID pOS_DestructPattern
        (
          pOS_DosBase *dosbase,
          pOS_PatternMatching *patternmatching
        );

FUNKTION
      Deinstallieren einer PatternMatching-Struktur.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      patternmatching (_R_A0)
       Zeiger auf die PatternMatching-Struktur, die deinstalliert werden soll

SIEHE AUCH
      pOS_ConstructPattern()

AMIGA FUNKTION
```

1.145 pdos.library/pOS_ParsePatternA()

```
PROTOTYP
      BOOL res = pOS_ParsePatternA
        (
          pOS_DosBase *dosbase,
          pOS_PatternMatching *patternmatching,
          const CHAR *string,
          const pOS_TagItem* tagList
```

```
        );

FUNKTION
      Vorbereiten einer Muster-Zeichenkette, diese kann mittels
      pOS_MatchPattern() verarbeitet werden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      patternmatching (_R_A0)
        Zeiger auf die PatternMatching-Struktur, die das
        konvertierte Muster aufnehmen soll.
      string (_R_A1)
        Zeiger auf die Muster-Zeichenkette
      tagList (_R_A2)

ERGEBNIS
      res (_R_D0)
        TRUE wenn alles ok ist
        FALSE im Fehlerfall, dann enthält patternmatching->pmt_Parse.ptmt_Error
        den Fehlercode DOSERR_NoMem.

SIEHE AUCH
      pOS_MatchPattern()

AMIGA FUNKTION
      LONG ParsePattern(pattern,buffer,size);
```

1.146 pdos.library/pOS_MatchPattern()

```
PROTOTYP
      BOOL res = pOS_MatchPattern
        (
          pOS_DosBase *dosbase,
          __CONST__ pOS_PatternObj *area[],
          const CHAR *string
        );

FUNKTION
      Ermitteln, ob eine Zeichenkette zu einem Namensmuster paßt.
      Dabei wird nicht zwischen Groß- und Kleinschreibung unterschieden.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      area (_R_A0)
        patternmatching->pmt_First
      string (_R_A1)
        Vergleichszeichenkette

ERGEBNIS
      res (_R_D0)
        TRUE wenn der string auf das Muster paßt
        sonst FALSE.
```

```
SIEHE AUCH
      pOS_ParsePatternA()

AMIGA FUNKTION
          LONG MatchPattern(STRPTR pattern, STRPTR string);

      /*****************************************************************************/ ←
```

1.147 pdos.library/pOS_PathMatchFirst()

```
PROTOTYP
      pOS_DosErrors res = pOS_PathMatchFirst
        (
          pOS_DosBase *dosbase,
          const pOS_FileLock *filelock,
          const dosname_t *filename,
          pOS_AnchorPath *anchorpath,
          const CHAR *pattern,
          ULONG flags
        );

FUNKTION
      Ermitteln einer Datei oder Verzeichnisses anhand eines Namensmusters.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        wird ein filelock angegeben, so kann der filename relativ
        dazu angegeben werden. NULL für pr_CurrentDir
      filename (_R_A1)
        Name des Verzeichnisses/Volumes, in dem die Suche begonnen wird
        darf leer sein ("") aber nicht NULL
      anchorpath (_R_A2)
        Zeiger auf verwendete AnchorPath-Struktur
      pattern (_R_A3)
        Zeiger auf die Muster-Zeichnekette
      flags (_R_D0) (enum pOS_AnchorPathFlags)
        APF_StdPattern : automatisch #? als Pattern kreieren, wenn keines  ↩
            angegeben wurde
        APF_PrtError   : Fehler in StadardFehlerausgabestrom ausgeben
        APF_MultiAssign: alle Verzeichnisse eines MultiAssigns untersuchen

ERGEBNIS
      res (_R_D0)
        DOSERR_None wenn die Daten ermittelt wurden
        sonst ein spezifischer Fehlercode.
        DOSERR_NoMoreEntries wenn keine weiteren Dateien vorliegen
        die auf das Muster passen

HINWEIS
      anchorpath->ap_BreakBits kann z.B. auf DOSSIGF_CTRL_C gesetzt
```

```
      werden, damit mit CTRL+C das scanning abgebrochen werden kann.

      Die AnchorPath-Struktur kann selber verlängert werden,
      dann kann der vollständige Dateiname aus anchorpath->ap_Buf
      gelesen werden. In anchorpath->ap_BufLen muß die Größe des
      Pufferspeichers eingetragen werden.

      struct {
        pOS_AnchorPath AP;
        CHAR Path[pOS_DosPathName_MAX];
      } ExtendedAnchorPath;

SIEHE AUCH
      pOS_PathMatchEnd(), pOS_PathMatchNext(), pOS_ConstructPattern(),
      pOS_ParsePatternA()

AMIGA FUNKTION
      LONG MatchFirst(STRPTR pattern, struct AnchorPath *anchorpath);
```

1.148 pdos.library/pOS_PathMatchNext()

```
PROTOTYP
      pOS_DosErrors res = pOS_PathMatchNext
        (
          pOS_DosBase *dosbase,
          pOS_AnchorPath *anchorpath
        );

FUNKTION
      Ermitteln der nächsten Datei/Verzeichnis anhand eines Namensmusters.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      anchorpath (_R_A0)
        Zeiger auf verwendete AnchorPath-Struktur

ERGEBNIS
      res (_R_D0)
        DOSERR_None wenn die Daten ermittelt wurden
        sonst ein spezifischer Fehlercode.
        DOSERR_NoMoreEntries wenn keine weiteren Dateien vorliegen
        die auf das Muster passen

        ap_Flags:

        APF_IsWild     : Patternmatching paßt auf das aktuelle File oder Dir.
        APF_DoEnterDir : Das aktuelle Objekt ist ein DIR und beim nächsten
                         Aufruf von pOS_PathMatchNext() entscheidet diese Bit,
                         ob in das DIR verzweigt wird.
        APF_ReturnDir  : Der letzte Dir-Eintrag ist erreicht, der nächte Aufruf
                         von pOS_PathMatchNext() ist im Parent-Dir.
```

```
SIEHE AUCH
      pOS_PathMatchFirst(), pOS_PathMatchEnd()

AMIGA FUNKTION
      LONG MatchNext(struct AnchorPath *anchorpath);
```

1.149 pdos.library/pOS_PathMatchEnd()

```
PROTOTYP
      VOID pOS_PathMatchEnd
        (
          pOS_DosBase *dosbase,
          pOS_AnchorPath *anchorpath
        );

FUNKTION
      Freigeben der mittels pOS_PathMatchFirst() oder pOS_PathMatchNext()
      angelegten Speicherbereiche.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      anchorpath (_R_A0)
        Zeiger auf verwendete AnchorPath-Struktur

SIEHE AUCH
      pOS_PathMatchFirst(), pOS_PathMatchNext()

AMIGA FUNKTION
          VOID MatchEnd(struct AnchorPath *anchorpath);

      /*****************************************************************************/ ←
```

1.150 pdos.library/pOS_DosCheckSignal()

```
PROTOTYP
      ULONG res = pOS_DosCheckSignal
        (
          pOS_DosBase *dosbase,
          ULONG signalmask
        );

FUNKTION
      Prüfen auf Unterbrechungssignale.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      signalmask (_R_D0)
```

```
        Unterbrechungssignale die geprüft werden sollen und nach
        dem Aufruf gelöscht sind.
        Definiert sind DOSSIGF_CTRL_C, DOSSIGF_CTRL_D, DOSSIGF_CTRL_E,
        DOSSIGF_CTRL_F.

ERGEBNIS
      res (_R_D0)
        Aufgetretenes Unterbrechungssignal, dessen Maske in signalmask
        gesetzt war oder 0.

AMIGA FUNKTION
          ULONG CheckSignal(ULONG signalmask);

      /*****************************************************************************/ ←
```

1.151 pdos.library/pOS_GetDateStamp()

```
PROTOTYP
      pOS_DateStamp *res = pOS_GetDateStamp
        (
          pOS_DosBase *dosbase,
          pOS_DateStamp *datestamp
        );

FUNKTION
      Ermitteln der aktuellen Systemzeit

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      datestamp (_R_A0)
        Zeiger auf DateStamp der das Ergebnis aufnehmen soll
        und auch zurückgeliefert wird

ERGEBNIS
      res (_R_D0)
        entspricht dem übergabeparameter datestamp

SIEHE AUCH
      pOS_CompareDates(), pOS_DateToStr(), pOS_StrToDate()

AMIGA FUNKTION
      struct DateStamp *DateStamp(struct DateStamp *datestamp);
```

1.152 pdos.library/pOS_CompareDates()

```
PROTOTYP
      SWORD res = pOS_CompareDates
        (
```

```
          pOS_DosBase *dosbase,
          const pOS_DateStamp *datestamp1,
          const pOS_DateStamp *datestamp2
        );

FUNKTION
      Vergleichen zweier Zeitwerte

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      datestamp1 (_R_A0)
        erster Zeitwert
      datestamp2 (_R_A1)
        zweiter Zeitwert

ERGEBNIS
      res (_R_D0)
        0 wenn beide Zweitwerte gleich sind
        1 wenn datestamp1 früher wie datestamp2 ist
       -1 wenn datestamp1 später wie datestamp2 ist

SIEHE AUCH
      pOS_GetDateStamp(), pOS_DateToStr(), pOS_StrToDate()

AMIGA FUNKTION
      LONG CompareDates(struct DateStamp *datestamp1, struct DateStamp * ↩
          datestamp2);
```

1.153 pdos.library/pOS_DateToStr()

```
PROTOTYP
      BOOL res = pOS_DateToStr
        (
          pOS_DosBase *dosbase,
          pOS_DateTime *datetime
        );

FUNKTION
      Konvertieren einer DateStamp-Struktur in eine Zeichenkette

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      datetime (_R_A0)
        Zeiger auf die DateTime-Struktur, gefüllt werden muß
        dat_Stamp : mit dem gewünschten Datums/Zeitwert
        dat_Format: (enum pOS_DateTimeFormat)
              DATETFRM_DOS  : dd-mmm-yy
              DATETFRM_INT  : yy-mm-dd
              DATETFRM_USA  : mm-dd-yy
              DATETFRM_CDN  : dd-mm-yy
              DATETFRM_LOCAL: Länderspezifisches Datum
        dat_Flags : (enum pOS_DateTimeFlags)
```

```
              DATETIMEF_Subst     : Datum wird in Heute, Montag, ... konvertiert
              DATETIMEF_Future    : damit Montag als nächster Montag gilt,
                                    sonst als vergangener Montag
              DATETIMEF_UsePreForm: es wird nicht der DateStamp
                                    verwendet, sondern das in einen String
                                    zu wandelne Datum wird aus den Werten
                                    dat_Day, dat_Month, dat_Year
                                    dat_Hour, dat_Minute, dat_Second
                                    ausgelesen

        dat_StrDay : enthält nach dem Aufruf den Wochentagnamen
                     wenn ein Speicherbereich angegeben wurde (Größe  ↩
                        pOS_DATETIMESTR_MAX)
                     ~0 und DATETIMEF_UsePreForm==0
                         Zwischenwerte berechnen (dat_Weekday)

        dat_StrDate: enthält nach dem Aufruf den Datumsstring
                     wenn ein Speicherbereich angegeben wurde (Größe  ↩
                        pOS_DATETIMESTR_MAX)
                     ~0 und DATETIMEF_UsePreForm==0
                         Zwischenwerte berechnen (dat_Year, dat_Month, dat_Day,
                                                  DATETIMEF_366Days)

        dat_StrTime: enthält nach dem Aufruf den Zeitstring
                     wenn ein Speicherbereich angegeben wurde (Größe  ↩
                        pOS_DATETIMESTR_MAX)
                     ~0 und DATETIMEF_UsePreForm==0
                         Zwischenwerte berechnen (dat_Hour, dat_Minute,  ↩
                            dat_Second)

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Komponenten gültige Zeitwerte enthalten haben
        und diese in einen String konvertiert wurden (dat_StrDay,
        dat_StrDate, dat_StrTime; wahlweise noch dat_Day, dat_Month,
        dat_Year, dat_Hour, dat_Minute, dat_Second)
        Im Fehlerfall FALSE.

SIEHE AUCH
      pOS_StrToDate()

AMIGA FUNKTION
      LONG DateToStr(struct DateTime *datetime);
```

1.154 pdos.library/pOS_StrToDate()

```
PROTOTYP
      BOOL res = pOS_StrToDate
        (
          pOS_DosBase *dosbase,
          pOS_DateTime *datetime
        );

FUNKTION
```

```
      Konvertieren einer Zeichenkette in eine DateStamp-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      datetime (_R_A0)
        Zeiger auf die DateTime-Struktur, gefüllt werden muß
        dat_Format: (enum pOS_DateTimeFormat)
                    DATETFRM_DOS  : dd-mmm-yy
                    DATETFRM_INT  : yy-mm-dd
                    DATETFRM_USA  : mm-dd-yy
                    DATETFRM_CDN  : dd-mm-yy
                    DATETFRM_LOCAL: Länderspezifisches Datum
        dat_Flags : (enum pOS_DateTimeFlags)
                    DATETIMEF_Subst     : Datum wird in Heute, Montag, ...
                                          konvertiert
                    DATETIMEF_Future    : damit Montag als nächster Montag
                                          gilt, sonst als vergangener Montag
                    DATETIMEF_UsePreForm: es werden die Zwischenvariablen
                                          ausgelesen sonst die Stringvariablen

        ist DATETIMEF_UsePreForm gesetzt, werden die Datums/Zeitwerte  aus
        dat_Day, dat_Month, dat_Year, dat_Hour, dat_Minute, dat_Second
        gelesen, sonst aus dat_StrDay, dat_StrDate, dat_StrTime

        dat_Stamp : enthält nach dem Aufruf den ermittelten Datums/Zeitwert

ERGEBNIS
      res (_R_D0)
        TRUE wenn der String einen gültigen Zeitwert enthalten hat
        und deren Komponenten ermittelt wurden (dat_Stamp).
        Im Fehlerfall FALSE.

SIEHE AUCH
      pOS_DateToStr()

AMIGA FUNKTION
          LONG StrToDate(struct DateTime *datetime);

      /*****************************************************************************/ ←
```

1.155 pdos.library/pOS_GetVar()

```
PROTOTYP
      ULONG res = pOS_GetVar
        (
          pOS_DosBase *dosbase,
          const CHAR *name,
          UBYTE *buffer,
          size_t size,
          ULONG flags,
          pOS_Process *process
        );
```

```
FUNKTION
      Ermitteln des Wertes einer lokalen oder globalen Umgebungsvariable

PARAMETER
          dosbase (_R_LB)
            Zeiger auf pOS_DosLibrary
          name (_R_A0)
            Name der Umgebungsvariable
          buffer (_R_A1)
            Zeiger auf den Speicherbereich, der den Wert aufnehmen soll
          size (_R_D0)
            Größe des Speicherbereichs
          flags (_R_D1) (enum pOS_LocalVarType | enum pOS_LocalVarFlags)
            LOCALVAR_Var  : eine Umgebungsvariable
            LOCALVAR_Alias: ein Alias

            GVARF_GlobalOnly  : nur globale Variablen
      ~     GVARF_LocalOnly   : nur lokale Variablen
            GVARF_BinaryVar   : Variable ist binär
            GVARF_DontNullTerm: Variable ist binär und deshalb nicht
                                Null-Terminiert

      ~     GVARF_GlobalOnly | GVARF_LocalOnly für globale und lokale Variablen

          process (_R_A2)
            Zeiger auf den Process, dessen Variablen ermittelt werden
            sollen oder NULL für den aktuellen Process

ERGEBNIS
      res (_R_D0)
        Anzahl der in den Speicherbereich übertragenen Zeichen
        (ohne abschließendem Nullbyte)
        pOS_DOSERR im Fehlerfall dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_DeleteVar(), pOS_SetVarLoc()

AMIGA FUNKTION
      LONG GetVar(STRPTR name, STRPTR buffer, LONG size, ULONG flags);
```

1.156 pdos.library/pOS_SetVar()

```
PROTOTYP
      BOOL res = pOS_SetVar
        (
          pOS_DosBase *dosbase,
          const CHAR *name,
          const UBYTE *buffer,
          size_t size,
          ULONG flags,
          pOS_Process *process
        );
```

```
FUNKTION
      Setzen des Wertes einer lokalen oder globalen Umgebungsvariable

PARAMETER
          dosbase (_R_LB)
            Zeiger auf pOS_DosLibrary
          name (_R_A0)
            Name der Umgebungsvariable
          buffer (_R_A1)
            Zeiger auf den Speicherbereich, der den Wert aufnehmen soll
          size (_R_D0)
            Größe des Speicherbereichs
          flags (_R_D1) (enum pOS_LocalVarType | enum pOS_LocalVarFlags)
            LOCALVAR_Var  : eine Umgebungsvariable
            LOCALVAR_Alias: ein Alias

            GVARF_GlobalOnly  : nur globale Variablen
      ~     GVARF_LocalOnly   : nur lokale Variablen
            GVARF_BinaryVar   : Variable ist binär
            GVARF_DontNullTerm: Variable ist binär und deshalb nicht Null- ↩
                Terminiert

      ~     GVARF_GlobalOnly | GVARF_LocalOnly für globale und lokale Variablen

            GVARB_SaveVar: Variable wird ENVARC: gespeichert
                           sonst nach ENV: (nur wenn GVARF_GlobalOnly
                           gesetzt ist)

          process (_R_A2)
            Zeiger auf den Process, dessen Variablen gesetzt werden
            sollen oder NULL für den aktuellen Process

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Variable gesetzt werden konnte,
        sonst FALSE, dann liefert pOS_GetIoErr()die Fehlerursache

SIEHE AUCH
      pOS_DeleteVar(), pOS_SetVarLoc(), pOS_GetVar()

AMIGA FUNKTION
      BOOL SetVar(STRPTR name, STRPTR buffer, LONG size, ULONG flags);
```

1.157 pdos.library/pOS_FindVar()

```
PROTOTYP
      pOS_LocalVar *res = pOS_FindVar
        (
          pOS_DosBase *dosbase,
          const CHAR *name,
          ULONG flags,
          pOS_Process *process
        );
```

```
FUNKTION
      Suchen einer lokalen oder globalen Umgebungsvariable

PARAMETER
          dosbase (_R_LB)
            Zeiger auf pOS_DosLibrary
          name (_R_A0)
            Name der Umgebungsvariable
          flags (_R_D0) (enum pOS_LocalVarType | enum pOS_LocalVarFlags)
            LOCALVAR_Var  : eine Umgebungsvariable
            LOCALVAR_Alias: ein Alias

            GVARF_GlobalOnly  : nur globale Variablen
      ~     GVARF_LocalOnly   : nur lokale Variablen
            GVARF_BinaryVar   : Variable ist binär
            GVARF_DontNullTerm: Variable ist binär und deshalb nicht Null- ↩
                Terminiert

      ~     GVARF_GlobalOnly | GVARF_LocalOnly für globale und lokale Variablen

          process (_R_A1)
            Zeiger auf den Process, dessen Variablen ermittelt werden
            sollen oder NULL für den aktuellen Process

ERGEBNIS
      res (_R_D0)
        Zeiger auf die gefundene LocalVar oder NULL im Fehlerfall.
        Dann liefert pOS_GetIoErr() als Fehler DOSERR_ObjectNotFound.

SIEHE AUCH
      pOS_DeleteVar(), pOS_GetVar(), pOS_SetVarLoc()

AMIGA FUNKTION
          struct LocalVar *FindVar(STRPTR name, ULONG type);


      *
```

1.158 pdos.library/pOS_DeleteVar()

```
PROTOTYP
      BOOL res = pOS_DeleteVar
        (
          pOS_DosBase *dosbase,
          const CHAR *name,
          ULONG flags,
          pOS_Process *process
        );

FUNKTION
      Entfernen einer lokalen oder globalen Umgebungsvariable

PARAMETER
          dosbase (_R_LB)
```

```
            Zeiger auf pOS_DosLibrary
          name (_R_A0)
            Name der Umgebungsvariable
          flags (_R_D0) (enum pOS_LocalVarType | enum pOS_LocalVarFlags)
            LOCALVAR_Var  : eine Umgebungsvariable
            LOCALVAR_Alias: ein Alias

            GVARF_GlobalOnly  : nur globale Variablen
      ~     GVARF_LocalOnly   : nur lokale Variablen
            GVARF_BinaryVar   : Variable ist binär
            GVARF_DontNullTerm: Variable ist binär und deshalb nicht Null- ↩
                Terminiert

      ~     GVARF_GlobalOnly | GVARF_LocalOnly für globale und lokale Variablen

          process (_R_A1)
            Zeiger auf den Process, dessen Variablen gelöscht werden
            sollen oder NULL für den aktuellen Process

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Variable gelöscht werden konnte,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_GetVar(), pOS_SetVar()

AMIGA FUNKTION
      BOOL DeleteVar(STRPTR name, ULONG flags);
```

1.159 pdos.library/pOS_SetVarLoc()

```
PROTOTYP
      BOOL res = pOS_SetVarLoc
        (
          pOS_DosBase *dosbase,
          pOS_LocalVar *localvar,
          const UBYTE *buffer,
          size_t size
        );

FUNKTION
      Setzen des Wertes einer lokalen Umgebungsvariable

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      localvar (_R_A0)
        Name der localen Variablen
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der den Wert aufnehmen soll
      size (_R_D0)
        Größe des Speicherbereichs
```

```
ERGEBNIS
      res (_R_D0)
        TRUE wenn die lokale Umgebungsvariable (erzeugt und) gesetzt
        werden konnte, sonst FALSE. Dann liefert pOS_GetIoErr() DOSERR_NoMem.

SIEHE AUCH
      pOS_DeleteVar(), pOS_SetVar()

AMIGA FUNKTION
```

1.160 pdos.library/pOS_CloneVars()

```
PROTOTYP
      BOOL res = pOS_CloneVars
        (
          pOS_DosBase *dosbase,
          const pOS_ExList *exlist1,
          pOS_ExList *exlist2,
          pOS_MemPool *mempool,
          ULONG flags
        );

FUNKTION
      Kopieren aller Umgebungsvariablen aus einer Liste in eine andere

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      exlist1 (_R_A0)
        Source-Liste, aus der die Variablen gelesen werden
      exlist2 (_R_A1)
        Dest-Liste, in die die Variablen kopiert werden
      mempool (_R_A2)
        MemPool in dem die Variablen erzeugt werden sollen
      flags (_R_D0) (enum pOS_LocalVarType | enum pOS_LocalVarFlags)

ERGEBNIS
      res (_R_D0)
        TRUE wenn alle Variablen kopiert wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH
      pOS_SetVarLoc()

AMIGA FUNKTION
```

1.161 pdos.library/pOS_GetDosErrText()

```
PROTOTYP
      const CHAR *res = pOS_GetDosErrText
```

```
        (
          pOS_DosBase *dosbase,
          pOS_DosErrors doserror
        );

FUNKTION
      Ermitteln eines Fehlertextes zu einem DosErrorCode.
      Der Text wird in der lokalen Sprache zurückgeliefert.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      doserror (_R_D0)
        Fehlercode, dessen Text ermittelt werden soll

ERGEBNIS
      res (_R_D0)
        Zeiger auf den ermittelten Text,
        NULL wenn für doserror DOSERR_None übergeben wurde
        sonst "Unknown DOSERR" für nicht vorhandene Fehlertexte

HINWEIS
      Die Funktion verändert nicht den aktuellen Errorcode
      des Processes.

SIEHE AUCH
      pOS_DosErrorReportA(), pOS_PrintDosErr(), pOS_GetIoErr()

AMIGA FUNKTION
      LONG Fault(LONG doserror, STRPTR header, STRPTR buffer, LONG size);
```

1.162 pdos.library/pOS_PrintDosErr()

```
PROTOTYP
      BOOL res = pOS_PrintDosErr
        (
          pOS_DosBase *dosbase,
          pOS_FileHandle *filehandle,
          const CHAR *string,
          ULONG doserror
        );

FUNKTION
      Ausgeben eines Fehlertextes zu einem DosErrorCode.
      Der Text wird in der lokalen Sprache ausgegeben.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filehandle (_R_A0)
        FileHandle wohin der Fehler ausgegeben werden soll
        oder NULL für den Standard-Fehlerausgabestrom, bzw.
        den Standard-Ausgabestrom wenn kein Fehlerstrom existiert
      string (_R_A1)
```

```
        Optionaler Text, der vor dem Fehlertext ausgegeben werden soll
        NULL für keinen vorangestellten Text
      doserror (_R_D0) (enum pOS_DosErrors)
        Fehlercode, dessen Text ermittelt werden soll
        oder 0 für den aktuellen Fehlercode des eigenen Processes

ERGEBNIS
      res (_R_D0)
        TRUE wenn ein Fehler vorlag und ausgegeben wurde,
        sonst FALSE.

HINWEIS
      Die Funktion verändert nicht den aktuellen Errorcode
      des Processes.

SIEHE AUCH
      pOS_DosErrorReportA(), pOS_GetDosErrText(), pOS_GetIoErr(),
      pOS_GetStdErrOutput()

AMIGA FUNKTION
          LONG PrintFault(LONG doserror, STRPTR header);

      /*****************************************************************************/ ←
```

1.163 pdos.library/pOS_SetCurrentDirLock()

```
PROTOTYP
      __ARID__ pOS_FileLock *res = pOS_SetCurrentDirLock
        (
          pOS_DosBase *dosbase,
          __ARID__ pOS_FileLock *filelock
        );

FUNKTION
      Setzen eines neuen aktuellen Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      filelock (_R_A0)
        Zeiger auf den FileLock, der auf das neue aktuelle Verzeichnis
        zeigt.
        ACHTUNG: nach der Funktion darf auf diesen Zeiger nicht
                 mehr zugegriffen werden

ERGEBNIS
      res (_R_D0)
        FileLock des alten aktuellen Verzeichnisses.
        ACHTUNG: handelt es sich bei diesem Zeiger um das echte
                 Startverzeichnis, darf er nicht verändert werden
                 und muß vor Beendigung des Processes wieder als
                 aktuelles Verzeichnis gesetzt werden.
                 Zeigt er nicht auf das echte Startverzeichnis, so
```

```
                ist er mittels pOS_UnlockObject() freizugeben.

BEISPIEL
   pOS_FileLock *startlock, *newlock;
   if(newlock=pOS_LockObject(,NULL,"RAM:",FILELKACC_Shared))
   {
     startlock=SetCurrentDirLock(newlock);

     /* newlock ist jetzt ungültig !!! */

     /* ... Programmcode ... */

     /* Startverzeichnis von Prozessbeginn setzen */
     pOS_UnlockObject( SetCurrentDirLock(startlock) );
   }


SIEHE AUCH
      pOS_SetCurrentDirName(), pOS_LockObject()

AMIGA FUNKTION
      BPTR CurrentDir(BPTR filelock);
```

1.164 pdos.library/pOS_SetCurrentDirName()

```
PROTOTYP
      BOOL res = pOS_SetCurrentDirName
        (
          pOS_DosBase *dosbase,
          const dosname_t *pathname
        );

FUNKTION
      Setzen eines neuen aktuellen Verzeichnisses.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      pathname (_R_A0)
        Name des neuen Verzeichnisses

ERGEBNIS
      res (_R_D0)
        TRUE wenn das neue Verzeichnis als aktuelles Verzeichnis
        gesetzt werden konnte,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache
        (normal DOSERR_DirNotFound wenn pathname ungültig ist)

HINWEIS
      Vor Beendigung des Prozesses sollte das bei Prozessbeginn
      aktuelle Verzeichnis gesetzt werden.

SIEHE AUCH
      pOS_SetCurrentDirLock()
```

```
AMIGA FUNKTION
```

1.165 pdos.library/pOS_GetCurrentDirName()

```
PROTOTYP
      BOOL res = pOS_GetCurrentDirName
        (
          pOS_DosBase *dosbase,
          dosname_t *buffer,
          size_t size
        );

FUNKTION
      Ermittelt den Namen des aktuellen Verzeichnisses

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      buffer (_R_A0)
        Zeiger auf den Speicherbereich, der den Pfadnamen
        aufnehmen soll
      size (_R_D0)
        Größe des Speicherbereichs

ERGEBNIS
      res (_R_D0)
        TRUE wenn der vollständige Pfadname des aktuellen
        Verzeichnisses in buffer steht,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache
        (DOSERR_NameToLong wenn der Speicherbereich zu klein ist)

SIEHE AUCH
      pOS_SetCurrentDirName(), pOS_SetCurrentDirLock(),
      pOS_NameFromObjectLock(), pOS_NameFromFH()

AMIGA FUNKTION
```

1.166 pdos.library/pOS_ConstructDosMsgNotify()

```
PROTOTYP
      VOID pOS_ConstructDosMsgNotify
        (
          pOS_DosBase *dosbase,
          pOS_DosNotifyReq *dosnotifyreq,
          const dosname_t *filename,
          ULONG flags,
          pOS_MsgPort *msgport
        );
```

```
FUNKTION
      Installieren eines Notify-Requesters
      der mittels Messages kommuniziert

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosnotifyreq (_R_A0)
        Zeiger auf den zu installierenden Notify-Requester
      filename (_R_A1)
        Name des Verzeichnis/Datei die überwacht werden soll
      flags (_R_D0) (enum pOS_DosNotifyReqFlags)
        NOTIREQF_Message  : intern benutzt
        NOTIREQF_Signal   : intern benutzt
        NOTIREQF_WaitReply: es wird erst dann wieder eine Nachricht
                            geschickt, wenn die aktuelle beantwortet
                            wurde
        NOTIREQF_Initial  : wenn nach pOS_DosStartNotify() sofort
                            eine Message geschickt werden soll, falls
                            die Datei bereits exsistiert
      msgport (_R_A2)
        Zeiger auf den MsgPort, der eine Nachricht vom Typ
        pOS_DosNotifyMessage bekommt, wenn die Datei verändert
        wurde. Die Nachricht muß dann mittels pOS_ReplyMsg()
        zurückgeschickt werden.

SIEHE AUCH
      pOS_DosStartNotify()

AMIGA FUNKTION
```

1.167 pdos.library/pOS_ConstructDosSigNotify()

```
PROTOTYP
      VOID pOS_ConstructDosSigNotify
        (
          pOS_DosBase *dosbase,
          pOS_DosNotifyReq *dosnotifyreq,
          const dosname_t *filename,
          ULONG flags,
          pOS_Task *task,
          ULONG sigbit
        );

FUNKTION
      Installieren eines Notify-Requesters
      der mittels Signale kommuniziert

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosnotifyreq (_R_A0)
        Zeiger auf den zu installierenden Notify-Requester
      filename (_R_A1)
```

```
        Name des Verzeichnis/Datei die überwacht werden soll
      flags (_R_D0) (enum pOS_DosNotifyReqFlags)
        NOTIREQF_Message  : intern benutzt
        NOTIREQF_Signal   : intern benutzt
        NOTIREQF_WaitReply: bei Signal unbenutzt
        NOTIREQF_Initial  : wenn nach pOS_DosStartNotify() sofort
                            ein Signal geschickt werden soll, falls
                            die Datei bereits exsistiert
      task (_R_A2)
        Zeiger auf den Task, dem das sigbit geschickt wird,
        wenn die Datei verändert wurde.
      sigbit (_R_D1)
        Signalbit das versendet werden soll

SIEHE AUCH
      pOS_DosStartNotify()

AMIGA FUNKTION
```

1.168 pdos.library/pOS_DestructDosNotify()

```
PROTOTYP
      VOID pOS_DestructDosNotify
        (
          pOS_DosBase *dosbase,
          pOS_DosNotifyReq *dosnotifyreq
        );

FUNKTION
      Deinstallieren eines Notify-Requesters

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosnotifyreq (_R_A0)
        Zeiger auf den zu deinstallierenden Notify-Requester

SIEHE AUCH
      pOS_ConstructDosMsgNotify(), pOS_ConstructDosSigNotify(),

AMIGA FUNKTION
```

1.169 pdos.library/pOS_DosStartNotify()

```
PROTOTYP
      BOOL res = pOS_DosStartNotify
        (
          pOS_DosBase *dosbase,
          pOS_DosNotifyReq *dosnotifyreq
        );
```

```
FUNKTION
      Starten einer Dateiüberwachung durch einen Notify-Requesters

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosnotifyreq (_R_A0)
        Zeiger auf den vollständig installierten DosNotifyReq

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Objekt überwacht wird. Die Überwachung muß
        mittels pOS_DosEndNotify() beendet werden.
        FALSE im Fehlerfall, dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_ConstructDosMsgNotify(), pOS_ConstructDosSigNotify(),
      pOS_DosEndNotify()

AMIGA FUNKTION
      LONG StartNotify(struct NotifyRequest *dosnotifyreq);
```

1.170 pdos.library/pOS_DosEndNotify()

```
PROTOTYP
      VOID pOS_DosEndNotify
        (
          pOS_DosBase *dosbase,
          pOS_DosNotifyReq *dosnotifyreq
        );

FUNKTION
      Beenden einer Dateiüberwachung durch einen Notify-Requesters
      die mit pOS_DosStartNotify() gestartet wurde

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      dosnotifyreq (_R_A0)
        Zeiger auf den vollständig installierten DosNotifyReq

HINWEIS
      Die Überwachung kann mittels pOS_DosStartNotify() wieder
      aktiviert werden oder die Strukturdaten mittels
      pOS_DestructDosNotify() freigegeben werden.

SIEHE AUCH
      pOS_DosStartNotify(), pOS_DestructDosNotify()

AMIGA FUNKTION
          VOID EndNotify(struct NotifyRequest *dosnotifyreq);
```

```
/*****************************************************************************/ ←
```

1.171 pdos.library/pOS_DosErrorReportA()

```
PROTOTYP
      BOOL res = pOS_DosErrorReportA
        (
          pOS_DosBase *dosbase,
          ULONG doserror,
          const pOS_TagItem *tags
        );

FUNKTION
      Erzeugen eines System-Dialogfensters wegen einem Fehler oder
      Rücksprache.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      doserror (_R_D0) (enum pOS_DosErrors)
        Dosfehlernummer, für den das Fenster ausgegeben werden soll
      tags (_R_A0) (enum pOS_DosReportTags)
        DOSRPTAG_FileLock   - (struct pOS_FileLock*)
        DOSRPTAG_FileHandle - (struct pOS_FileHandle*)
        DOSRPTAG_Path       - (const dosname_t*)
        DOSRPTAG_Iteration  - (ULONG*) (0,1,2..) wird vom Req. hochgezählt

ERGEBNIS
      res (_R_D0)
        TRUE wenn das Fenster dargestellt und vom Benutzer das linke
        Gadget ("Retry", "Nochmal versuchen", ...) bestätigt wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_GetDosErrText(), pOS_PrintDosErr()

AMIGA FUNKTION
      BOOL ErrorReport(LONG doserror, LONG type, ULONG arg, struct MsgPort * ↩
          device);
```

1.172 pdos.library/pOS_MountDosDevice()

```
PROTOTYP
      UWORD res = pOS_MountDosDevice
        (
          pOS_DosBase *dosbase,
          const CHAR *devicename,
          const dosname_t *filename,
          const CHAR *args
```

```
        );

FUNKTION
      Anmelden eines neuen Gerätes

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      devicename (_R_A0)
        Name des zu mountenden Devices (z.B. DF0) ohne Doppelpunkt
      filename (_R_A1)
        Name der Datei, aus der die Argumente gelesen werden sollen
      args (_R_A2)
        Argumentstring

      Als Argumente sind folgende Parameter möglich:
        DEVICE/K: Exec-Device-Name (z.B. scsi.device)
        UNIT/K/N: Exec-Device-Unit-Nr
        BPB=BytesPerBlock/K/N: Anzahl Bytes pro Block (z.B. 512)
        BPC=BlocksPerCyl/K/N: Anzahl Blöcke je Zylinder (z.B. 11)
        SC=StartCyl/K/N: Erster Zylinder (z.B. 0)
        EC=EndCyl/K/N:  Letzter Zylinder (z.B. 79)
        SF=Surfaces/K/N: Anzahl Schreib/Leseköpfe (z.B. 2)
        SRV=StartRev/K/N: Anzahl reservierter Blöcke am Anfang
                          (z.B. 2 für BootHeader)
        ERV=EndRev/K/N: Anzahl reservierter Blöcke am Ende (z.B. 0)
        ITL=Interleave/K/N: wieviele Blöcke beim fortlaufenden Lesen
                            überspungen werden müssen (z.B. wenn die
                            Platte schneller läuft, als der Prozessor
                            die Daten lesen kann; heute normal 0)
        BUF=BufferSize/K/N: Größe des Zwischenpuffers
        MTYP=BufMemType/K/N: Type des Zwischenpuffers (MEMF_PUBLIC,
                             MEMF_PRIVATE, ...)
        MXTF=MaxTransfer/K/N: Maximale Blockgröße je Transfer (z.B. 0xffffff)
        MASK/K/N: Maske für DMA-Controller-Speicherzugriff (z.B. 0x7ffffffe)
        BB=BootBlocks/K/N: Anzahl der Bootblöcke (z.B. 2)
        BP=BootPri/K/N: Boot-Priorität des Datenträgers (z.B. 0)
        DFLG/K/N: Flags für pOS_OpenDevice() die an DEVICE weitergereicht
                  werden (normal 0)
        DDNAME/K: DosDevice-Name (z.B. pNETFS.ddv)
        STARTUP/K: Startparameter für Handler
        TYPE_PICO/S: neues Gerät ist ein Minimalgerät
        TYPE_BOD/S: neuen Gerät ist ein blockorientiertes Gerät
        TYPE_NET/S: neues Gerät ist ein Netzgerät
        SRCID/K: eigene Netzadresse (nur TYPE_NET)
        DSTID/K: Netzadresse des Partners (nur TYPE_NET)
        TIMEOUT/N: Acknowledge-Timeout in 1/10 sec (nur TYPE_NET)

HINWEIS
      Es ist entweder filename oder args zu übergeben.
      Werden beide Werte übergeben, so wird nur filename verwendet.
      Es ist entweder TYPE_PICO, TYPE_BOD oder TYPE_NET möglich.

ERGEBNIS
      res (_R_D0)
        DOSERR_Non wenn alles ok ist, sonst einen spezifischen Fehlercode
        (enum pOS_DosErrors)
```

```
SIEHE AUCH
      pOS_CreateDosMount(), pOS_DeleteDosMount(), pOS_GetDosMountName(),
      pOS_CreateDosDevFromMount()

AMIGA FUNKTION
```

1.173 pdos.library/pOS_ConstructShell()

```
PROTOTYP
      VOID pOS_ConstructShell
        (
          pOS_DosBase *dosbase,
          pOS_Shell *shell
        );

FUNKTION
      Installieren einer Shell-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      shell (_R_A0)
        Zeiger auf die Shell-Struktur, die installiert werden soll

SIEHE AUCH
      pOS_ConstructShellScript(), pOS_DestructShell()

AMIGA FUNKTION
```

1.174 pdos.library/pOS_DestructShell()

```
PROTOTYP
      VOID pOS_DestructShell
        (
          pOS_DosBase *dosbase,
          pOS_Shell *shell
        );

FUNKTION
      Deinstallieren einer Shell-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      shell (_R_A0)
        Zeiger auf die Shell-Struktur, die deinstalliert werden soll

SIEHE AUCH
      pOS_ConstructShell()
```

```
AMIGA FUNKTION
```

1.175 pdos.library/pOS_ConstructShellScript()

```
PROTOTYP
      VOID pOS_ConstructShellScript
        (
          pOS_DosBase *dosbase,
          pOS_ShellScript *shellscript
        );

FUNKTION
      Installieren einer ShellScript-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      shellscript (_R_A0)
        Zeiger auf die ShellScript-Struktur, die installiert werden soll

SIEHE AUCH
      pOS_ConstructShell(), pOS_DestructShellScript()

AMIGA FUNKTION
```

1.176 pdos.library/pOS_DestructShellScript()

```
PROTOTYP
      VOID pOS_DestructShellScript
        (
          pOS_DosBase *dosbase,
          pOS_Shell *shell,
          pOS_ShellScript *shellscript
        );

FUNKTION
      Deinstallieren einer ShellScript-Struktur

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      shell (_R_A0)
        Zeiger auf die Shell-Struktur, in der das Shell-Script läuft
      shellscript (_R_A1)
        Zeiger auf die ShellScript-Struktur, die deinstalliert werden soll

SIEHE AUCH
      pOS_ConstructShellScript()
```

```
AMIGA FUNKTION
```

1.177 pdos.library/pOS_DosGetAliasCom()

```
PROTOTYP
      CHAR *res = pOS_DosGetAliasComm
        (
          pOS_DosBase *dosbase,
          const CHAR *zeile,
          CHAR *buffer,
          size_t size
        );

FUNKTION
      Alle Alias der Shell- oder Scriptzeile auflösen

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      zeile (_R_A0)
        Zeiger auf die Shell/Scriptzeile, die Alias-Texte enthalten kann
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der das konvertierte Ergebnis aufnehmen ↩
           soll
      size (_R_D0)
        Größe des Speicherbereichs

ERGEBNIS
      res (_R_D0)
        buffer wenn die Zeile konvertiert wurde,
        sonst NULL, dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_DosGetResString1()

AMIGA FUNKTION
```

1.178 pdos.library/pOS_DosGetResString1()

```
PROTOTYP
      CHAR res* = pOS_DosGetResString1
        (
          pOS_DosBase *dosbase,
          const CHAR *zeile,
          CHAR *buffer,
          size_t size,
          pOS_Shell *shell
        );

FUNKTION
```

```
      Auflösen einer Shell- oder Scriptzeile

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      zeile (_R_A0)
        Zeiger auf die Shell/Scriptzeile, die aufgelöst werden soll
      buffer (_R_A1)
        Zeiger auf den Speicherbereich, der das konvertierte Ergebnis aufnehmen ↩
           soll
      size (_R_D0)
        Größe des Speicherbereichs
      shell (_R_A2)
        Zeiger auf die Shell

ERGEBNIS
      res (_R_D0)
        buffer wenn die Zeile konvertiert wurde,
        sonst NULL, dann liefert pOS_GetIoErr() die Fehlerursache

SIEHE AUCH
      pOS_ConstructShell(), pOS_DosGetAliasCom()

AMIGA FUNKTION
```

1.179 pdos.library/pOS_SetPrompt()

```
PROTOTYP
      BOOL res = pOS_SetPrompt
        (
          pOS_DosBase *dosbase,
          const CHAR *string
        );

FUNKTION
      Setzen der aktuellen Eingabeaufforderung

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      string (_R_A0)
        Text für den neuen Prompt, dabei sind folgende Platzhalter
        möglich:
        %N für die aktuelle Processnummer
        %S für den Namen des aktuellen Verzeichnisses
        %R interner Fehlercode der letzten Dos-Funktion
        %F für das Ergebnis des letzten Kommandos
           (definiert sind DOSFAIL_OK, DOSFAIL_WARN, DOSFAIL_ABORT,
            DOSFAIL_ERROR, DOSFAIL_FAIL)

ERGEBNIS
      res (_R_D0)
        TRUE wenn der neue Prompt gesetzt wurde,
        sonst FLASE. Dann liefert pOS_GetIoErr() die Fehlerursache
```

```
HINWEIS
      Der Default-Prompt ist "%N.%S.%R[%F]> ".

SIEHE AUCH
      pOS_GetPrompt()

AMIGA FUNKTION
      LONG SetPrompt(STRPTR string);
```

1.180 pdos.library/pOS_GetPrompt()

```
PROTOTYP
      BOOL res = pOS_GetPrompt
        (
          pOS_DosBase *dosbase,
          CHAR *buffer,
          size_t size
        );

FUNKTION
      Ermitteln der aktuellen Eingabeaufforderung

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      buffer (_R_A0)
        Zeiger auf den Speicherbereich, der den Prompttext
        aufnehmen soll
      size (_R_D0)
        Größe des Speicherbereichs

ERGEBNIS
      res (_R_D0)
        TRUE wenn der aktuelle Prompt in buffer übertragen wurde,
        sonst FALSE. Dann liefert pOS_GetIoErr() die Fehlerursache.
        (DOSERR_NameToLong wenn der Speicherbereich zu klein ist)

HINWEIS
      Der Default-Prompt ist "%N.%S.%R[%F]> ".

SIEHE AUCH
      pOS_SetPrompt()

AMIGA FUNKTION
      LONG GetPrompt(STRPTR buffer, LONG size);
```

1.181 pdos.library/pOS_CloneProcessPath()

```
PROTOTYP
```

```
      BOOL pOS_CloneProcessPath
        (
          pOS_DosBase *dosbase,
          const pOS_Process *sourceprocess,
          pOS_Process *destinationprocess
        );

FUNKTION
      Kopieren der Suchpfade von einem anderen Process.

PARAMETER
      dosbase (_R_LB)
        Zeiger auf pOS_DosLibrary
      sourceprocess (_R_A0)
        Suchpfade die Übernommen werden sollen
      destinationprocess (_R_A1)
        Prozess, der um die zusätzlichen Suchpfade von sourceprocess
        erweitert wird. Bereits bestehende Suchpfade bleiben erhalten.

ERGEBNIS
      res (_R_D0)
        TRUE wenn die Pfade übernommen wurden;
        sonst FALSE, dann liefert pOS_GetIoErr() die Fehlerursache.

SIEHE AUCH

AMIGA FUNKTION
```