DosD

COLLABORATORS			
	TITLE : DosD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

DosD

1.1 DosD

Konstanten

abortBusy	abortDiskError	accessRead
accessWrite	actionEnd	actionNotKnown
addNotify	badHunk	badNumber
badStreamName	badTemplate	beginning
break	bufFull	bufLine
bufNone	bufferOverflow	changeFH
changeLock	changeMode	changeSignal
commentTooBig	copyDir	copyDirFh
createDir	ctrlC	ctrlD
ctrlE	ctrlF	current
currentVolume	deleteObject	deleteProtected
deviceNotMounted	die	dirNotFound
directoryNotEmpty	diskChange	diskFull
diskInfo	diskNotValidated	diskType
diskWriteProtected	doWriteProtect	dosCLI
dosDisk	dosExAllControl	dosFALSE
dosFib	dosFileHandle	dosName
dosRdArgs	dosStdPkt	dosTRUE
edComment	edDate	edName
edProtection	edSize	edType
end	error	event
examineAll	examineFh	examineNext
examineObject	exclusiveLock	fail
ffsDisk	fhFromLock	fileNotObject
findInput	findOutput	findUpdate
flush	format	formatMAX
freeLock	freeRecord	getBlock
info	inhibit	interDOSDisk
interFFSDisk	invalidComponentName	invalidLock
invalidResidentLibrary	isFileSystem	isSoftLink
itemEqual	itemError	itemNothing
itemQuoted	itemUnquoted	keyNeedsArg
kickstartDisk	ldAll	lenDatString
lineTooLong	linkHard	linkSoft
locateObject	lockCollision	lockDifferent
lockRecord	lockSame	lockSameHandler

lockSameVolume	lockTimeout	makeLink
maxMultiArgs	maxTemplateItems	moreCache
msdosDisk	newFile	nil
noDefaultDir	noDisk	noDiskPresent
noFreeStore	noMoreEntries	notADosDisk
notExecutable	notImplemented	notReallyDos
notifyClass	notifyCode	objectExists
objectInUse	objectLinked	objectNotFound
objectTooLarge	objectWrongType	ok
oldFile	parent	parentFh
read	readLink	readOnly
readProtected	readReturn	readWrite
recExclusive	recExclusiveImmed	recShared
recSharedImmed	recordNotLocked	removeNotify
renameAcrossDevices	renameDisk	renameObject
reportInsert	reportLock	reportStream
reportTask	reportVolume	requiredArgMissing
runExecute	runSystem	runSystemAsynch
sameLock	screenMode	seek
seekError	serializeDisk	setComment
setDate	setFileSize	setMap
setProtect	sharedLock	stFile
stLinkDir	stLinkFile	stPipeFile
stRoot	stSoftLink	stUserDir
startup	taskTableFull	ticksPerSecond
timer	tooManyArgs	tooManyLevels
unLockError	unmatchedQuotes	unreadableDisk
validated	validating	waitChar
warn	write	writeProtect
writeProtected	writeReturn	

Typ-Deklarationen

AChain	AChainFlagSet	AChainFlags
AChainPtr	AdoTags	AllocProc
AnchorPath	AnchorPathFlagSet	AnchorPathFlags
AnchorPathPtr	AssignList	AssignListPtr
BSTR	CSource	CSourcePtr
CliInitFlagSet	CliInitFlags	CliProcList
CliProcListPtr	CommandLineInterface	↩
↩ CommandLineInterfaceAPtr		
CommandLineInterfacePtr	Date	DateFormat
DatePtr	DateTime	DateTimeFlagSet
DateTimeFlags	DateTimePtr	DevProc
DevProcFlagSet	DevProcFlags	DevProcPtr
DeviceList	DeviceListPtr	DeviceListType
DeviceNode	DeviceNodePtr	DosEnvec
DosEnvecPtr	DosInfo	DosInfoPtr
DosLibrary	DosLibraryPtr	DosList
DosListPtr	DosPacket	DosPacketPtr
ErrorString	ErrorStringPtr	ExAllControl
ExAllControlPtr	ExAllData	ExAllDataPtr
FileHandle	FileHandlePtr	FileInfoBlock
FileInfoBlockPtr	FileLock	FileLockPtr
FileSysStartupMsg	FileSysStartupMsgPtr	FreeProc
ILArray	ILArrayPtr	InfoData
InfoDataPtr	LocalVar	LocalVarPtr

```
    LockDosFlagSet              LockDosFlags                NotifyFlagSet
    NotifyFlags                 NotifyMessage               NotifyMsgPtr
    NotifyRequest               NotifyRequestPtr            NpTags
    PathInfo                    PathInfoPtr                 Process
    ProcessFlagSet              ProcessFlags                ProcessId
    ProcessPtr                  ProtectionFlagSet           ProtectionFlags
    RDAFlagSet                  RDAFlags                    RDArgs
    RDArgsPtr                   ReadProc                    RecordLock
    RecordLockArrayPtr          RecordLockPtr               ResidentSegment
    ResidentSegmentAPtr         ResidentSegmentPtr          RootNode
    RootNodeFlagSet             RootNodeFlags               RootNodePtr
    SegmentPtr                  StandardPacket              StandardPacketPtr
    Str255                      StrPtr                      SysTags
    TaskArray                   TaskArrayAPtr               TaskArrayPtr
    Var16FlagSet                Var8FlagSet                 VarFlagSet
    VarFlags
```

1.2 TMP:Modula-2/DosD.def

```
DEFINITION MODULE DosD; (*$ Implementation:=FALSE *)
(* 01-Mar-1992/cn *)

FROM SYSTEM IMPORT ADDRESS,BPTR,BYTE,CAST,LONGSET,WORD;

FROM ExecD IMPORT
  Library , LibraryPtr , MemReqSet , Message , MessagePtr , MinList , MinNode
 , MsgPort , MsgPortPtr , Node , SignalSemaphore , Task , TaskPtr ;

FROM Timer IMPORT
  TimeRequestPtr ;

FROM UtilityD IMPORT
  HookPtr ,tagUser, TagItemPtr ;

IMPORT R;

TYPE  StrPtr =ADDRESS;

CONST
 dosName="dos.library";

 dosTRUE=LONGINT(-1);
 dosFALSE=LONGINT(0);

(*
 Mögliche Werte des Parameters accessMode der Prozedur Open
*)
 readWrite=1004;
 readOnly=1005;
 oldFile=readOnly;
 newFile=1006;

(*
 Mögliche Werte des Parameters mode der Prozedur Seek
*)
```

```
 beginning=-1;
 current=0;
 end=1;

(*
 Mögliche Werte des Parameters accessMode der Prozedur Lock
*)
 sharedLock=-2;
 exclusiveLock=-1;
 accessRead=sharedLock;
 accessWrite=exclusiveLock;

(*
 Rückgbewerte von SameLock()
*)
 lockDifferent=-1;
 lockSame=0;
 lockSameVolume=1;
 lockSameHandler=lockSameVolume;

(*
 Mögliche Werte des Parameters type der Prozedur ChangeMode
*)
 changeLock=0;
 changeFH=1;

(*
 Werte fuer MakeLink()
*)
 linkHard=0;
 linkSoft=1;

(*
 Mögliche Werte für den Parameter type der Prozedur SetVBuf
*)
 bufLine=0; (* flush on \n, etc. *)
 bufFull=1; (* never flush except when needed *)
 bufNone=2; (* no buffering *)

(*
 Mögliche Rückgabewerte von IoErr()
*)
 noFreeStore=103;
 taskTableFull=105;
 badTemplate=114;
 badNumber=115;
 requiredArgMissing=116;
 keyNeedsArg=117;
 tooManyArgs=118;
 unmatchedQuotes=119;
 lineTooLong=120;
 fileNotObject=121;
 invalidResidentLibrary=122;
 noDefaultDir=201;
 objectInUse=202;
 objectExists=203;
 dirNotFound=204;
```

```
 objectNotFound=205;
 badStreamName=206;
 objectTooLarge=207;
 actionNotKnown=209;
 invalidComponentName=210;
 invalidLock=211;
 objectWrongType=212;
 diskNotValidated=213;
 diskWriteProtected=214;
 renameAcrossDevices=215;
 directoryNotEmpty=216;
 tooManyLevels=217;
 deviceNotMounted=218;
 seekError=219;
 commentTooBig=220;
 diskFull=221;
 deleteProtected=222;
 writeProtected=223;
 readProtected=224;
 notADosDisk=225;
 noDisk=226;
 noMoreEntries=232;
 (*36*) isSoftLink=233;
 (*36*) objectLinked=234;
 (*36*) badHunk=235;
 (*36*) notImplemented=236;
 (*36*) recordNotLocked=240;
 (*36*) lockCollision=241;
 (*36*) lockTimeout=242;
 (*36*) unLockError=243;
 (*36*) bufferOverflow=303;
 (*36*) break=304;
 (*36*) notExecutable=305;

(*
 Mögliche Werte des Parameters code der Prozedur ErrorReport
*)
 abortBusy=288;
 abortDiskError=296;

(*
 Mögliche Werte des Parameters type der Prozedur ErrorReport
*)
 reportStream=0;
 reportTask=1;
 reportLock=2;
 reportVolume=3;
 reportInsert=4;

(*
 Vordefinierte Werte für den Parameter returnCode der Prozedur Exit
*)
 ok=0;
 warn=5;
 error=10;
 fail=20;
```

```
(*
 Rückgbewerte von ReadItem()
*)
 itemEqual=-2;
 itemError=-1;
 itemNothing=0;
 itemUnquoted=1;
 itemQuoted=2;

(*
 Mögliche Werte des Parameters type der Prozeduren AllocDosObject
 und FreeDosObject
*)
 dosFileHandle=0;
 dosExAllControl=1;
 dosFib=2;
 dosStdPkt=3;
 dosCLI=4;
 dosRdArgs=5;

 runExecute=-1;
 runSystem=-2;
 runSystemAsynch=-3;

(*
 break flags Exec.SetSignal, etc
*)
 ctrlC=12;
 ctrlD=13;
 ctrlE=14;
 ctrlF=15;

 ticksPerSecond=50;

TYPE
  BSTR =BPOINTER TO  Str255 ;
  DeviceListPtr =BPOINTER TO  DeviceList ;
  ProcessId = MsgPortPtr ;
  SegmentPtr =BPTR;
  Str255 =ARRAY[0..255] OF CHAR;

  FileHandle =RECORD
  link: MessagePtr ;
  port: MsgPortPtr ;
  type: ProcessId ;
  buf:LONGINT;
  pos:LONGINT;
  end:LONGINT;
  func1:LONGINT;
  func2:LONGINT;
  func3:LONGINT;
  arg1:LONGINT;
  arg2:LONGINT;
 END;
  FileHandlePtr =BPOINTER TO  FileHandle ;

  FileLockPtr =BPOINTER TO  FileLock ;
```

```
  FileLock =RECORD
  link: FileLockPtr ;
  key:LONGINT;
  access:LONGINT;
  task: ProcessId ;
  volume: DeviceListPtr ;
 END;

  Date =RECORD
  days:LONGINT;
  minute:LONGINT;
  tick:LONGINT;
 END;
  DatePtr =POINTER TO  Date ;

  ProtectionFlags =(
  delete,execute,writeProt,readProt,archive,pure,script,hidden,
  pf8,pf9,pf10,pf11,pf12,pf13,pf14,pf15,pf16,pf17,pf18,pf19,pf20,
  pf21,pf22,pf23,pf24,pf25,pf26,pf27,pf28,pf29,pf30,pf31
 );
  ProtectionFlagSet =SET OF  ProtectionFlags ;

  FileInfoBlock =RECORD
  diskKey:LONGINT;
  dirEntryType:LONGINT;
  fileName:ARRAY [0..107] OF CHAR;
  protection: ProtectionFlagSet ;
  entryType:LONGINT;
  size:LONGINT;
  numBlocks:LONGINT;
  date: Date ;
  comment:ARRAY [0..79] OF CHAR;
  reserved:ARRAY [0..35] OF CHAR;
 END;
  FileInfoBlockPtr =POINTER TO  FileInfoBlock ;

CONST
 stRoot=1;
 stUserDir=2;
 stSoftLink=3;
 stLinkDir=4;
 stFile=-3;
 stLinkFile=-4;
 stPipeFile=-5;

TYPE
  InfoData =RECORD
  numSoftErrors:LONGINT;
  unitNumber:LONGINT;
  diskState:LONGINT;
  numBlocks:LONGINT;
  numBlocksUsed:LONGINT;
  bytesPerBlock:LONGINT;
  diskType:LONGINT;
  volumeNode: DeviceListPtr ;
  inUse:LONGINT;
 END;
```

```
  InfoDataPtr =POINTER TO  InfoData ;

CONST
(*
 Mögliche Werte von InfoData.diskState
*)
 writeProtect=80;
 validating=81;
 validated=82;

(*
 Mögliche Werte von InfoData.diskType
*)
 noDiskPresent=-1;
 unreadableDisk=CAST(LONGINT,"BAD\o");
 dosDisk=CAST(LONGINT,"DOS\x00");
 (*36*) ffsDisk=CAST(LONGINT,"DOS\x01");
 (*??*) interDOSDisk=CAST(LONGINT,"DOS\x02");
 (*??*) interFFSDisk=CAST(LONGINT,"DOS\x03");
 notReallyDos=CAST(LONGINT,"NDOS");
 kickstartDisk=CAST(LONGINT,"KICK");
 (*??*) msdosDisk=CAST(LONGINT,"MSD\o");

TYPE
  ProcessFlags =(
  freeSegList,freeCurrDir,freeCLI,closeInput,closeOutput,freeArgs,
  pr6,pr7,pr8,pr9,pr10,pr11,pr12,pr13,pr14,pr15,pr16
 );
  ProcessFlagSet =SET OF  ProcessFlags ;

  PathInfoPtr =BPOINTER TO  PathInfo ;
  PathInfo =RECORD
  nextPath: PathInfoPtr ;
  lock: FileLockPtr ;
 END;

  CommandLineInterface =RECORD
  result2:LONGINT;
  setName: BSTR ;
  commandDir: PathInfoPtr ;
  returnCode:LONGINT;
  commandName: BSTR ;
  failLevel:LONGINT;
  prompt: BSTR ;
  standardInput: FileHandlePtr ;
  currentInput: FileHandlePtr ;
  commandFile: BSTR ;
  interactive:LONGINT; (* LONGBOOLEAN *)
  background:LONGINT; (* LONGBOOLEAN *)
  currentOutput: FileHandlePtr ;
  defaultStack:LONGINT;
  standardOutput: FileHandlePtr ;
  module:BPTR;
 END;
  CommandLineInterfacePtr =BPOINTER TO  CommandLineInterface ;
  CommandLineInterfaceAPtr =POINTER TO  CommandLineInterface ;
```

```
  Process =RECORD
  task: Task ;
  msgPort: MsgPort ;
  pad:WORD;
  segList:BPTR;
  stackSize:LONGINT;
  globVec:ADDRESS;
  taskNum:LONGINT;
  stackBase:BPTR;
  result2:LONGINT;
  currentDir: FileLockPtr ;
  cis: FileHandlePtr ;
  cos: FileHandlePtr ;
  consoleTask: ProcessId ;
  fileSystemTask: ProcessId ;
  cli: CommandLineInterfacePtr ;
  returnAddr:ADDRESS;
  pktWait:ADDRESS;
  windowPtr:ADDRESS;
  homeDir:  FileLockPtr ;
  flags: ProcessFlagSet ;
  exitCode:PROC;
  exitData:ADDRESS;
  arguments: StrPtr ;
  localVars: MinList ;
  shellPrivate:LONGCARD;
  ces: FileHandlePtr ;
 END;
  ProcessPtr =POINTER TO  Process ;

  DosPacket =RECORD
  link: MessagePtr ;
  port: MsgPortPtr ;
  type:LONGINT;
  res1:LONGINT;
  res2:LONGINT;
  arg1:LONGINT;
  arg2:LONGINT;
  arg3:LONGINT;
  arg4:LONGINT;
  arg5:LONGINT;
  arg6:LONGINT;
  arg7:LONGINT;
 END;
  DosPacketPtr =POINTER TO  DosPacket ;

  StandardPacket =RECORD
  msg: Message ;
  pkt: DosPacket ;
 END;
  StandardPacketPtr =POINTER TO  StandardPacket ;

CONST
 (*
  Mögliche Werte für DosPacket.type
 *)
 nil=0;
```

```
startup=nil;
getBlock=2;
setMap=4;
die=5;
event=6;
currentVolume=7;
locateObject=8;
renameDisk=9;
write=ORD('W');
read=ORD('R');
freeLock=15;
deleteObject=16;
renameObject=17;
moreCache=18;
copyDir=19;
waitChar=20;
setProtect=21;
createDir=22;
examineObject=23;
examineNext=24;
diskInfo=25;
info=26;
flush=27;
setComment=28;
parent=29;
timer=30;
inhibit=31;
diskType=32;
diskChange=33;
setDate=34;
(*36*) sameLock=40;
screenMode=994;
(*36*) changeSignal=995;
readReturn=1001;
writeReturn=1002;
findUpdate=1004;
findInput=1005;
findOutput=1006;
actionEnd=1007;
seek=1008;
(*36*) format=1020;
(*36*) makeLink=1021;
setFileSize=1022;
doWriteProtect=1023;
(*36*) readLink=1024;
(*36*) fhFromLock=1026;
(*36*) isFileSystem=1027;
(*36*) changeMode=1028;
(*36*) copyDirFh=1030;
(*36*) parentFh=1031;
(*36*) examineAll=1033;
(*36*) examineFh=1034;
(*36*) lockRecord=2008;
(*36*) freeRecord=2009;
(*36*) addNotify=4097;
(*36*) removeNotify=4098;
(*37*) serializeDisk=4200;
```

```
TYPE
  TaskArray =RECORD
  maxCli:LONGINT;
  cli:ARRAY [1..99] OF  ProcessId ;
 END;
  TaskArrayPtr =BPOINTER TO  TaskArray ;
  TaskArrayAPtr =POINTER TO  TaskArray ;

  CliProcList =RECORD
  node: MinNode ;
  first:LONGINT;
  array: TaskArrayAPtr ;
 END;
  CliProcListPtr =POINTER TO  CliProcList ;

  ResidentSegmentAPtr =POINTER TO  ResidentSegment ;
  ResidentSegmentPtr =BPOINTER TO  ResidentSegment ;
  ResidentSegment =RECORD
  next: ResidentSegmentPtr ;
  usecount:LONGINT;
  segment: SegmentPtr ;
  name: Str255 ; (* BString!! *)
 END;

  DosInfo =RECORD
  resList: ResidentSegmentPtr ;
  devInfo: DeviceListPtr ;
  devices:BPTR;
  handlers:BPTR;
  netHand: ResidentSegmentPtr ;
  devLock: SignalSemaphore ;
  entryLock: SignalSemaphore ;
  deleteLock: SignalSemaphore ;
 END;
  DosInfoPtr =BPOINTER TO  DosInfo ;

  RootNodeFlags =(
  rnf0,private1,rnf2,rnf3,rnf4,rnf5,rnf6,rnf7
  ,rnf8,rnf9,rnf10,rnf11,rnf12,rnf13,rnf14,rnf15
  ,rnf16,rnf17,rnf18,rnf19,rnf20,rnf21,rnf22,rnf23
  ,wildStar,rnf25,rnf26,rnf27,rnf28,rnf29,rnf30,rnf31
 );
  RootNodeFlagSet =SET OF  RootNodeFlags ;

  RootNode =RECORD
  taskArray: TaskArrayPtr ;
  consoleSegment: SegmentPtr ;
  time: Date ;
  restartSeg:BPTR;
  info: DosInfoPtr ;
  fileHandlerSegment: SegmentPtr ;
  cliList: MinList ;
  bootProc: ProcessId ;
  shellSegment: SegmentPtr ;
  flags: RootNodeFlagSet ;
 END;
```

```
 RootNodePtr =POINTER TO  RootNode ;

 ErrorString =RECORD
 nums: POINTER TO ARRAY [0..1000] OF LONGINT;
 strings:POINTER TO ARRAY [0..1000] OF  StrPtr ;
END;
 ErrorStringPtr =POINTER TO  ErrorString ;

 DosLibrary =RECORD
 lib: Library ;
 root: RootNodePtr ;
 gv:ADDRESS;
 a2:LONGINT;
 a5:LONGINT;
 a6:LONGINT;
 errors: ErrorStringPtr ;
 timeReq: TimeRequestPtr ;
 utilityBase: LibraryPtr ;
 intuitionBase: LibraryPtr ;
END;
 DosLibraryPtr =POINTER TO  DosLibrary ;

 DosEnvec =RECORD
 tableSize:LONGCARD;
 sizeBlock:LONGCARD;
 secOrg:LONGCARD;
 surfaces:LONGCARD;
 sectorsPerBlock:LONGCARD;
 blocksPerTrack:LONGCARD;
 reserved:LONGCARD;
 preAlloc:LONGCARD;
 interleave:LONGCARD;
 lowCyl:LONGCARD;
 highCyl:LONGCARD;
 numBuffers:LONGCARD;
 bufMemType:LONGCARD;
 maxTransfers:LONGCARD;
 mask:LONGSET;
 bootPri:LONGINT;
 dosType:ARRAY [0..3] OF CHAR;
 baud:LONGCARD;
 control:LONGCARD;
 bootBlocks:LONGCARD;
END;
 DosEnvecPtr =BPOINTER TO  DosEnvec ;

 FileSysStartupMsg =RECORD
 unit:LONGCARD;
 device: BSTR ;
 environ: DosEnvecPtr ;
 flags:LONGSET;
END;
 FileSysStartupMsgPtr =BPOINTER TO  FileSysStartupMsg ;

 AssignListPtr =POINTER TO  AssignList ;
 AssignList =RECORD
 next: AssignListPtr ;
```

```
  lock: FileLockPtr ;
 END;

  DeviceListType =(device,directory,volume,late,nonBinding);
  DeviceList =RECORD
  next: DeviceListPtr ;
  pad1,pad2,pad3:BYTE;
  type: DeviceListType ;
  task: ProcessId ;
  lock: FileLockPtr ;
  CASE : DeviceListType  OF
  | device:
   handler: BSTR ;
   stackSize:LONGINT;
   priority:LONGINT;
   startup: FileSysStartupMsgPtr ;
   segList:BPTR;
   globVec:BPTR;
  | volume:
   volumeDate: Date ;
   lockList: FileLockPtr ;
   diskType:LONGINT;
   unused:LONGINT
  | directory,late,nonBinding:
   assignName:ADDRESS;
   list: AssignListPtr ;
  END;
  name: BSTR ;
 END;
  DeviceNode = DeviceList ;
  DeviceNodePtr =POINTER TO  DeviceNode ;
  DosList = DeviceList ;
  DosListPtr =POINTER TO  DosList ;

  LockDosFlags =(
  ldbRead,ldbWrite,devices,volumes,assigns,entry,ldbDelete,ldb7
  ,ldb8,ldb9,ldb10,ldb11,ldb12,ldb13,ldb14,ldb15
  ,ldb16,ldb17,ldb18,ldb19,ldb20,ldb21,ldb22,ldb23
  ,ldb24,ldb25,ldb26,ldb27,ldb28,ldb29,ldb30,ldb31
 );
  LockDosFlagSet =SET OF  LockDosFlags ;

CONST
 ldAll= LockDosFlagSet {devices,volumes,assigns};

TYPE
  DevProcFlags =(
  unlock,assign,dpf2,dpf3,dpf4,dpf5,dpf6,dpf7
  ,dpf8,dpf9,dpf10,dpf11,dpf12,dpf13,dpf14,dpf15
  ,dpf16,dpf17,dpf18,dpf19,dpf20,dpf21,dpf22,dpf23
  ,dpf24,dpf25,dpf26,dpf27,dpf28,dpf29,dpf30,dpf31
 );
  DevProcFlagSet =SET OF  DevProcFlags ;

  DevProc =RECORD
  port: MsgPortPtr ;
  lock: FileLockPtr ;
```

```
  flags: DevProcFlagSet ;
  devNode:ADDRESS;
 END;
  DevProcPtr  = POINTER TO  DevProc ;

  RecordLock =RECORD
  fh: FileHandlePtr ; (* fh=NIL: Ende des Arrays *)
  offset:LONGCARD;
  length:LONGCARD;
  mode:LONGCARD;
 END;
  RecordLockPtr =POINTER TO  RecordLock ;
 (* generischer Array mit offener Obergrenze: *)
  RecordLockArrayPtr =POINTER TO ARRAY [0..4095] OF  RecordLock ;

CONST (* modes für LockRecords(), UnLockRecords() *)
 recExclusive=0;
 recExclusiveImmed=1;
 recShared=2;
 recSharedImmed=3;

TYPE
  ExAllDataPtr =POINTER TO  ExAllData ;
  ExAllData =RECORD
  next: ExAllDataPtr ;
  name: StrPtr ;
  type:LONGINT;
  size:LONGCARD;
  prot: ProtectionFlagSet ;
  date: Date ;
  comment: StrPtr ;
 END;

  ExAllControl =RECORD
  entries:LONGCARD;
  lastKey:LONGCARD;
  matchString: StrPtr ;
  matchFunc: HookPtr ;
 END;
  ExAllControlPtr =POINTER TO  ExAllControl ;

CONST
(*
 Mögliche Werte für den Parameter type von ExAll
*)
 edName=1;
 edType=2;
 edSize=3;
 edProtection=4;
 edDate=5;
 edComment=6;

TYPE
  DateFormat =(formatDOS,formatINT,formatUSA,formatCDN);

CONST
 formatMAX=MAX( DateFormat );
```

```
 lenDatString=16;

TYPE
  DateTimeFlags =(subst,future,dtf2,dtf3,dtf4,dtf5,dtf6,dtf7);
  DateTimeFlagSet =SET OF  DateTimeFlags ;

  DateTime =RECORD
  date: Date ;
  format: DateFormat ;
  flags: DateTimeFlagSet ;
  strDay: StrPtr ;
  strDate: StrPtr ;
  strTime: StrPtr ;
 END;
  DateTimePtr =POINTER TO  DateTime ;

  SysTags =(sysDummy:=tagUser+32,
  sysInput,sysOutput,sysASynch,sysUserShell,sysCustomShell
 );

  NpTags =(npDummy:=tagUser+1000,
  npSeglist,npFreeSeglist,npEntry,npInput,npOutput,npCloseInput
  ,npCloseOutput,npError,npCloseError,npCurrentDir,npStackSize,npName
  ,npPriority,npConsoleTask,npWindowPtr,npHomeDir,npCopyVars,npCli
  ,npPath,npCommandName,npArguments,npNotifyOnDeath,npSynchronous
  ,npExitCode,npExitData
 );

  AdoTags =(adoDummy:=tagUser+2000,
  adoFhMode,adoDirLen,adoCommNameLen,adoCommFileLen,adoPromptLen
 );

CONST
 notifyClass=40000000H;
 notifyCode=1234H;

TYPE
  NotifyFlags =(
  sendMessage,sendSignal,nf2,waitReply,notifyInitial,
  nf5,nf6,nf7,nf8,nf9,nf10,nf11,nf12,nf13,nf14,nf15,
  nf16,nf17,nf18,nf19,nf20,nf21,nf22,nf23,nf24,nf25,
  nf26,nf27,nf28,nf29,nf30,magic
 );
  NotifyFlagSet =SET OF  NotifyFlags ;

  NotifyRequest =RECORD
  name: StrPtr ;
  fullName: StrPtr ;
  userData:ADDRESS;
  flags: NotifyFlagSet ;
  CASE :INTEGER OF
  | 0:
   port: MsgPortPtr ;
  | 1:
   task: TaskPtr ;
   signalNum:SHORTCARD;
   pad:ARRAY[1..3] OF SHORTCARD;
```

```
 END;
 reserved:ARRAY[1..4] OF LONGCARD;
 msgCount:LONGCARD;
 handler: MsgPortPtr ;
END;
 NotifyRequestPtr =POINTER TO  NotifyRequest ;

 NotifyMessage =RECORD
 execMessage: Message ;
 class:LONGCARD;
 code:CARDINAL;
 nReq: NotifyRequestPtr ;
 doNotTouch:LONGCARD;
 doNotTouch2:LONGCARD;
END;
 NotifyMsgPtr =POINTER TO  NotifyMessage ;

 AChainFlags =(
 patternBit,examinedBit,completed,allBit,single,acf5,acf6,acf7
);
 AChainFlagSet =SET OF  AChainFlags ;

 AChainPtr =POINTER TO  AChain ;
 AChain =RECORD
 child,parent: AChainPtr ;
 lock: FileLockPtr ;
 info: FileInfoBlock ;
 flags: AChainFlagSet ;
 (* string:ARRAY [0..??] OF CHAR; *)
END;

 AnchorPathFlags =(
 doWild,itsWild,doDir,didDir,noMemErr,doDot,dirChanged,followHLinks
);
 AnchorPathFlagSet =SET OF  AnchorPathFlags ;

 AnchorPath =RECORD
 base,last: AChainPtr ;
 breakBits:LONGSET;
 foundBreak:LONGSET;
 flags: AnchorPathFlagSet ;
 reserved:SHORTINT;
 strLen:INTEGER;
 info: FileInfoBlock ;
 (* buff:ARRAY [0..strLen-1] OF CHAR; *)
END;
 AnchorPathPtr =POINTER TO  AnchorPath ;

 CSource =RECORD
 buffer: StrPtr ;
 length:LONGINT;
 curChr:LONGINT;
END;
 CSourcePtr =POINTER TO  CSource ;

 RDAFlags =(
 stdIn,noAlloc,noPrompt,rdaf3,rdaf4,rdaf5,rdaf6,rdaf7
```

```
  ,rdaf8,rdaf9,rdaf10,rdaf11,rdaf12,rdaf13,rdaf14,rdaf15
  ,rdaf16
 );
  RDAFlagSet =SET OF  RDAFlags ;
  RDArgs =RECORD
  source: CSource ;
  daList:LONGINT;
  buffer:ADDRESS;
  bufSiz:LONGINT;
  extHelp: StrPtr ;
  flags: RDAFlagSet ;
 END;
  RDArgsPtr =POINTER TO  RDArgs ;

CONST
 maxTemplateItems=100;
 maxMultiArgs=128;

TYPE
  VarFlags =(
  alias,vf1,vf2,vf3,vf4,vf5,vf6,ignore
  ,globalOnly,localOnly,binaryVar,dontNullTerm,vf12,vf13,vf14,vf15
  ,vf16
 );
(*
 VarFlagSet wird als Paramter verwendet (32 bit) in xxxVar().
 Var16FlagSet wird in LocalVar verwendet (16 bit).
 Var8FlagSet wird verwendet um LocalVar.node.type zu setzen.
*)
  VarFlagSet =SET OF  VarFlags ;
  Var16FlagSet =SET OF [alias..vf15];
  Var8FlagSet =SET OF [alias..ignore];

  LocalVar =RECORD
  node: Node ;
  flags: Var16FlagSet ;
  value: StrPtr ;
  len:LONGCARD;
 END;
  LocalVarPtr =POINTER TO  LocalVar ;

  AllocProc =PROCEDURE(LONGINT{R.D0}, MemReqSet {R.D1}):ADDRESS;
  FreeProc =PROCEDURE(ADDRESS{R.A1},LONGINT{R.D0});
  ReadProc =PROCEDURE(
           FileHandlePtr{R.D1},ADDRESS{R.A0},LONGINT{R.D0}):LONGINT;

  ILArray =RECORD
  read: ReadProc ;
  alloc: AllocProc ;
  free: FreeProc ;
 END;
  ILArrayPtr =POINTER TO  ILArray ;

  CliInitFlags =(
  runOutput,userInput,system,systemAsynch,cif4,cif5,cif6,cif7
  ,cif8,cif9,cif10,cif11,cif12,cif13,cif14,cif15
  ,cif16,cif17,cif18,cif19,cif20,cif21,cif22,cif23
```

```
  ,cif24,cif25,cif26,cif27,cif28,cif29,cif30,valid
 );
  CliInitFlagSet =SET OF  CliInitFlags ;

END DosD.noimp
```