0821b668-0

COLLABORATORS

	TITLE : 0821b668-0		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

0821b668-0

1.1 PackDev

```
                                        PackDev
                   A Device Packer With XPK Support
                             V 1.6
                        by Christian Wasner
                           03-Sep-95
```

```
        ~Legal~Stuff~~~

        ~Introduction~~

        ~Usage        ~

        ~Packers~~~~~~~

         DOS and I/O

        ~Compatibility~

        ~Bugs~~~~~~~~~~

        ~Future~~~~~~~~

        ~History~~~~~~~

         Author
                        ---- ANOTHER FINE PHANTASM TOOL ----
        _____ _____ _____  ____ _____
        ____  /  /  /__ ____ _/ ._ __   /  __/___ _/
        _/ __/      /     /  / /// /    /__  / / / /
        ___//__/__/_/__/__/__/__/__/__/___ /_/_/_/LNF
```

1.2   Legal Stuff

Disclaimer

The author cannot be held liable for the suitability or accuracy of this
manual and/or the program(s) it describes. Any damage directly or
indirectly caused by the use or misuse of this manual and/or the program it
describes is the sole responsibility of the user her/him self.

Copyright/Distribution

These programs are FREEWARE, so no financial donations are required (but
welcome). They may be freely distributed as long as all files remain
unchanged and are included with the distribution. Distribution on disks or
CDs is permitted only on the disks or CDs from Fred Fish or the Aminet CDs.
Electronical distribution (e.g. by Aminet, mailboxes, modems) is allowed.
Inclusion into freeware software packages is allowed, inclusion into other
packages must be expressly allowed by me in written form.

The following files should come along:

PackDev   (23928 bytes)
PackDev.guide (30607 bytes)

1.3   Introduction

With this program you can read or write data directly from/to a ←
disk. The
blocks of the disk are read and stored into a file. When reading from a
DOS disk, only the blocks that are used are archived. When writing an
archive to a disk, only the blocks are written that are stored in the
archive. Optionally the data read from a disk can be packed with an
xpk
packer. The program currently only has a shell interface, but ←
it won't
crash if started from workbench.

All this sounds like "Oh no, yet another DMS clone", but this program can
handle any
device
with a
filesystem
(e.g. DHx:, DFx:, a RAD: of any size
etc.), DMS can only handle floppy disks or
devices
with the size of floppy
disks. Even Non-DOS disks can be handled (but then all blocks are read).
Another advantage of PackDev is that is doesn't use stolen code like the
authors of DMS do (see below). PackDev supports the
xpk
packer system, so

it's much more flexible than DMS. You can use any
                 xpk
                  packer you like,
i.e. you can use a packer suited to the type of data on the disk. When
comparing size, you see that PackDev is only 22 KB long. If you pack it
with PowerPacker, it would be less than 12 KB !

A disk is read as follows:

    1. Read
                  device
                   data from DOS and the disk bootblock
    2. Check if the
                  filesystem
                    of the disk is supported (currently OFS,
      FFS, OFS International, FFS International, OFS Directory Dache and
      FFS Directory Cache are supported)
    3. Inhibit
                  device
                     If the
                  filesystem
                   is supported and the ALL keyword is not set:

    4. Read root block (contains location of block allocation map = BAM)
    5. Read BAM and store it (after packing, if specified)
    5. Read used blocks and store them (after packing, if specified)
    6. Quit

    If the
                  filesystem
                   is not supported or the ALL keyword is set:

    4. Read all blocks and store them (after packing, if specified)
    5. Quit

Writing functions similarly.

When writing to a
                  filesystem
                  , DevPack will only allow
                  filesystems
                   that
are exactly of the same partition size, block size and number of reserved
blocks. If the disk should be formatted, the track size of the disk (e.g.
11 blocks for a floppy disk) must currently be equal, too.

1.4 Documentation

                 Program: PackDev

Template: FROM/O,TO/O,P=PACK/K/O,XB=XPKBUFSIZE/K/N/O,M=MEMTYPE/K/O,
        CU=CLRUNUSED/S/O,ETDF=ETDFORMAT/S/O,TDF=TDFORMAT/S/O,
        TDFL=TDFLABEL/S/O,NVF=NOVERIFY/S/O,ALL/S/O,NV=NOVERBOSE/S/O,
        Q=QUIET/S/O,NC=NOCONFIRM/S/O,VIEWFILE/K/O,VIEWFILESYS/K/O,

          TESTFILE/K/O,BLOCKLIST/K/O,PASSWORD/K/O,TN=TRUENAME/S/O

Purpose: Reading/writing data directly from/to
          filesystems
           with
          xpk
                  support.

          The following parameters are supported:

          FROM, TO

          PACK (or P)

          MEMTYPE (or M)

          CLRUNUSED (or CU)

          ETDFORMAT (or ETDF)

          TDFORMAT (or TDF)
               (CHANGED)

          TDFLABEL (or TDFL)
               (NEW)

          NOVERIFY (or NVF)
               (NEW)

          ALL

          NOVERBOSE (or NV)

          QUIET (or Q)

          NOCONFIRM (or NC)

          VIEWFILE

          VIEWFILESYS

          TESTFILE

          BLOCKLIST

          PASSWORD

          TRUENAME (or TN)
               (NEW)

          Examples for usage

1.5 Packers

XPK packers and their efficiency with PackDev

All packers were used with an efficiency of 100. The source disk was the
Workbench 3.0 disk from which devs#?, install#? prefs#?, system#? and
libs#? were deleted in order to get a slightly fragmented disk that is
app. 50% full. the test computers were an A500 with a 030 board (25 MHz)
along with the disk in DF0: (a typical usage with an average-speed CPU)
and an A4000/060 along with the disk in RAD: (one of the fastest Amigas
along with one of the fastest device). The data on disk was 422400 bytes.
Only packing was tested.

		------ A500/030/DF0: -------				------ A4000/060/RAD: -------		
Vers.	Packer	Time	Size	Bytes/s	Gain/s	Time	Size	Bytes/s	Gain/s	
1.98	MASH	44.4	201516	9500	4977	5.0	201520	84800	44353	+
1.2	CBR1	23.2	365012	18200	2473	0.4	365012	1056000	143470	
1.0	ACCA	23.8	261604	17750	6761	0.6	261604	728250	277234	
36.1	LHLB	80.3	215996	5250	2569	8.1	215996	52100	25481	
1.10	SQSH	41.6	229820	10150	4631	3.7	229820	114750	52331	
1.0	SMPL	25.5	372960	16550	1937	0.8	372960	555750	65052	
2.0	SHRI	84.9	191300	4950	2723	7.6	191304	55400	30327	+
1.2	RLEN	23.9	358232	17600	2680	0.4	358232	960000	145836	
3.3	RDCN	23.6	264364	17900	6707	0.5	264360	782200	292666	+
1.0	NUKE	31.1	220136	13550	6499	2.1	220136	201100	96316	
1.0	NONE	22.6	436396	18650	−618	0.4	436396	1056000	−34990	#
1.0	IMPL	136.7	214756	3050	1518	17.8	214756	23650	11639	
1.0	IDEA	27.4	436520	15350	−514	0.9	436520	449350	−15021	*
0.63	HUFF	27.4	317788	15350	3812	1.2	317788	357950	88654	
1.36	HFMN	24.1	315188	17500	4452	0.5	315192	782200	198533	+
1.3	FEAL	35.7	436512	11800	−395	1.7	436512	245550	−8204	*
1.6	FAST	33.5	250536	12600	5136	2.0	250536	215500	87685	
1.2	ENCO	23.3	436288	18100	−595	0.4	436288	1056000	−34720	*
0.1	DLTA	23.2	436180	18150	−593	0.4	436180	1056000	−34450	%
0.58	DHUF	23.1	436228	18250	−598	0.4	436228	1056000	−34570	$
1.0	CBR0	23.5	365012	17900	2437	0.4	365012	1005700	136638	
3.0	BLZW	25.1	262112	16800	6375	0.9	262116	459100	174221	+
1.6	RAKE	26.5	212108	15900	7935	1.1	212112	398450	198384	+
1.0	PWPK	65.0	223624	6500	3059	7.0	223628	60150	28315	+

*: These libraries are crypters, they don't pack
#: This is a do-nothing library, it obviously doesn't pack
$: This library seems to be nonfunctional
+: The archive sizes are different on different systems. I assume that
   these packers need a minimum input buffer size, and when the disk is
   completely read and the buffer is not filled with enough data, the
   packer packs also the remaining space of the buffer which contains
   random trash. Perhaps someone knows better.
%: This is a packer specialized in sounds

Note that I cannot get the packer libraries xpkCRMS.library and
xpkCRM2.library opened (yes, I have the CrM.library), so their specs are
missing here.

1.6 Compatibility

This program needs OS V2.0+ to run. If you want to use
xpk
packers (very
likely), you need the xpkmaster.library and some packer sublibraries (I
suggest SHRI, MASH, NUKE, RAKE, RDCN or ACCA). The
xpk
package and some
more sublibraries should be present in any good pd mailbox and in the
Aminet. They are not included here because it's much larger than the
PackDev package.

1.7 Bugs

If the disk should be formatted, the track size of the disk (e.g. 11
blocks for a floppy disk) must currently be equal to the track size of the
disk from which the file is read. This is not necessary, but coding the
thing this way is easier and faster.

Should you detect a bug, please tell me (email or phone). Be as specific as
you can.

1.8 Future

What may be done in the future:

Support of ProfessionalFileSystem and MS-DOS
filesystem
GUI (yes, really, I will do it...tomorrow :-))

Device
that treats an archive like a disk
Built-in packer

What will not be done in the future:

DMS compatibility (see below)

Localization (I hate those zillions of useless files, there is no support
for people who cannot speak English. This sounds arrogant, but I think,
this tool should not be used by inexperienced users anyway)

Versions for each type of processor (I made the experience that doing this
causes a negligible speedup that is not worth even writing this sentence,
but perhaps there will be a C compiler that can do better...)

1.9   History

```
Aug-14 1994 V1.0 - Initial release, never released I think...

Aug-18 1994 V1.1 - Minor bugs fixed

Apr-16 1995 V1.2 - Problems  with  OFS disks fixed (PackDev didn't know the
                     number of free/used blocks)
                   - ALL, NOVERBOSE, QUIET and NOCONFIRM keyword added
                   - Doc file corrected and improved
                   - Filesystem  type  is  now  read  from block 0 instead of
                     reading  it from the DOS node, because the latter always
                     contains DOS\0 for Amiga floppies
                   - Minor bugfixes

Apr-30 1995 V1.3 - If the partition with LIBS: on it is to be handled,
                     PackDev could not open XPK (sub-)libraries, fixed
                   - Minimum XPK buffer size corrected
                   - TESTFILE parameter added
                   - Checksums installed, in case an xpk packer doesn't keep
                     them..
                   - Argument handling changed (you got me, Christian...)
                   - BLOCKLIST parameter added
                   - PASSWORD parameter added

Jul-02 1995 V1.4 - OS 2.0 workaround: Filesystems cannot be inhibited if
                     the DosList is locked. Now it is unlocked before
                     inhibiting (Thanks, Golly).
                   - Bugfix: DosList was locked with LDF_READ|LDF_DEVICES,
                     but unlocked with LDF_READ|LDF_VOLUMES.
                   - Minor docfile editing

Aug-17 1995 V1.5 - Read/write error output/user interaction was only done
                     if QUIET or NOCONFIRM were set. This must be vice versa,
                     of course (Thanks Dirk)
                   - Documentation is now in Amigaguide format (Thanks to
                     Edd Dumbill, the author of Heddley, a great Amigaguide
                     editor)

Sep-03 1995 V1.6 - Added new parameters: TDFLABEL, NOVERIFY, TRUENAME
                   - TDFORMAT does no longer write label buffers, use TDFLABEL
                     for this in future
                   - XPK timing docs revised
                   - Guide file improved
                   - The guide became longer than the executable :-)
```

1.10   Author

```
Christian Wasner

Phone  ++49-40-7236349

Email: wasner@ifmsun1.ifm.uni-hamburg.de (or wasner@ifm.uni-hamburg.de)
       u241045@niesel.dkrz.d400.de
```

        CRISI@BLACKBOX.SHNET.ORG

If possible, use email. If you phone me, please do it from 8pm to 10pm and
don't forget that I have Central European Time here, so 8pm for you may not
mean 8pm for me !

Everybody who reports a bug via email, receives the next (bugfixed) version
directly via email.

1.11   FROM, TO

              These parameter specifies the source data and the destination ←
                 data. The
source is specified first and the destination second. Either both or none
of these two keywords must be set. If the one parameter is a
              filesystem
              the other one must be a file and vice versa. If both ←
                 parameters are
specified along with their keywords, they can be placed anywhere and in any
order of the command string. If VIEWFILE, VIEWFILESYS or TESTFILE is set,
FROM and TO must not be used.

File names will be handled the following way: When writing to the file, a
suffix ".pkd" will be added to the file name if it's not present. When
reading from it and the file name doesn't contain the ".pkd" suffix, it's
checked first if there is a file <name>.pkd. If this file doesn't exist
then the original file name is used. Note that this can be switched off
with the TRUENAME parameter.

Note that copy-protected disks (i.e. with a custom track format) cannot be
handled.

Examples:

DH0: foobar   Read data from DH0: and write it to foobar.pkd foobar
DH0:   Read data from foobar.pkd (foobar if not present) and
                     write it to DH0:

TO foobar FROM DH0: Read data from DH0: and write it to foobar.pkd, other
                     parameters can be placed anywhere between, before or
                     after them.

1.12   PACK (or P)

              This parameter is optional, but strongly suggested.   It ←
                 may only be
specified along with a READ action, because when writing to a disk, the
packer type of the archive is automatically recognized. Along with PACK an

        xpk

                    packer name and the efficiency can be specified (separated by a ←
                       ".").
Note that some
                 xpk
                    packers ignore the efficience value. A discussion of
packer speed is found in v).

When comparing with DMS, a general rule is : The less full a disk is, the
slower is DMS because DMS always reads all blocks, even if they are not put
into the archive (maybe ParCon is afraid of "lamers" who think, DMS forgets
to read these blocks :-).

Examples:

PACK NUKE   (Use NUKE packer with default efficiency)
P SHRI.75   (Use SHRI packer with efficience 75)

1.13   MEMTYPE (or M)

                This parameter is optional. It specifies the memory type that is ←
                   used for

                device
                 buffers.   Some older
                devices
                  may require chip memory for their
buffers. For example, under 1.3 the
                trackdisk.device
                  needed chipmem
because it used the blitter to decode the data (not because the disk DMA
functions with chipmem only). Possible values are CHIP, FAST, ANY. Under
2.0+ trackdisk doesn't need chipmem any longer. Default is ANY.

Examples:

M CHIP (chipmem will be used or program fails)
M FAST (fastmem will be used or program fails)
M ANY (default: highest-priority memory will be used)

1.14   CLRUNUSED (or CU)

This optional parameter is allowed only when writing to a disk. If set,
all unused blocks are overwritten with zeroes. This has the disadvantage
that the write operation becomes slower, but has the advantage that tools
like DiskSalv ((C) by Dave Haynie) don't find old file fragments when
trying to undelete an accidentally deleted file (deleted after usage of
PackDev, of course). This parameter is unset by default.

Example:

CU   (PackDev overwrites unused tracks with zeroes)

1.15   ETDFORMAT (or ETDF)

                This   parameter   is   optional. It should be set if you want to ←
                  write to an
unformatted floppy disk which is to be accessed by
                trackdisk.device
                . It
may not function if you write to disks that are accessed by other
                devices
                ,
because this parameter causes PackDev to use a system format routine that
functions with floppy disks (DFx:), but may not function with other
                devices
                (like scsi
                devices
                  and especially replacement
                devices
                 for flppies). The
advantage of this format routine is that
                label buffers
                 can be written with
it in one row (Filenotes are NOT stored here, I told this in an earlier
version of the doc file... but nobody noticed it/everybody believed it).
TDFORMAT will ignore
                label buffers
                 and TDFLABEL will start an extra run for
writing them the
                label buffers
                . For experts: This causes PackDev to
format with ETD_FORMAT. ETDF is unset by default.

Example

ETDF (enhanced format routine for floppies)

1.16   TDFORMAT (or TDF)

                This parameter is optional. It should not be set if you want to ←
                  write to
an unformatted floppy disk with
                trackdisk.device
                 (i.e. DFx) because these
support ETDFORMAT (see above).   It will function if you write to a hard
disk, RAD:, FFx:, diskspare.device disks etc., because this parameter
causes PackDev to use the general format routine that should function with
any disk
                device
                 that is supported by DOS. The drawback of this general
system format routine (TD_FORMAT, to be specific) is that it does not write

the
                label buffers
                  (don't ask my why, I don't know). If you write an
archive to disk that has data in its
                label buffers
                 and needs them (some
non-dos disks with trackloaders) then use TDFLABEL (see below). Standard
disks with a
                filesystem
                 (i.e. "dir" etc. works with them) normally don't
store data within the
                label buffers
                 , except there are stupid executables on
them, but this is not very probable. I don't suggest to use this parameter
for hard disks, RAD: etc. because their tracks don't need to be
Amiga-formatted.   It is implemented because there may exist some
                devices
                that need formatting but don't support the enhanced routine. TDF ←
                  is unset
by default.

Example

TDF (standard format routine)

1.17 TDFLABEL (or TDFL)

                With this parameter set the same is done as with TDFORMAT (see ←
                  above), but
the
                label buffers
                 are also written. Because this must be done in an extra
run, TDFLABEL is twice as slow as TDFORMAT. Use this parameter only if

                label buffers
                 need to be written (some trackloader disks may use them) and
something other than a disk written to by
                trackdisk.device
                 (DFx:) is the
target.

Example

TDFL (standard format routine with extra write run for
                label buffers
)

1.18 NOVERIFY (or NVF)

This parameter switches verifying off when writing to a disk. Use it with
RAD:, but all other disks (especially floppies) tend to be unreliable, so
disks that are not for one-shot usage should always be verified when
written to.

Example

NV (No verify)

1.19   ALL

This parameter can be used when reading from a
device
. If set, PackDev
reads all blocks from the
device
, no matter if the
filesystem
 is known or
not. This is useful for all these poor demo disks that have a
filesystem
on it, but also raw data on blocks that are not used by the
filesystem
. I
hesitated before implementing this parameter because I HATE disks of this
kind, but SiliconSurfer insisted on it. Come on boys, stop these lame
combination of DOS and trackloaders...

Example

ALL (All blocks are read)

1.20   TRUENAME (or TN)

If this parameter is set, the archive file name is used exactly as
specified, i.e.   no .pkd stuff is done with the filename (see FROM, TO).
This parameter can only be set if an action is specified that involves an
archive, of course.

Example

TN (Use file name exactly as specified)

1.21   NOVERBOSE (or NV)

NOVERBOSE (or NV) =================

This parameter suppresses the output of the blocks currently worked on. This is useful when the output is redirected to a file. It's disabled by default.

Example

NV (no output of blocks currently worked on)

1.22 QUIET (or Q)

If this parameter is set, nothing is written to the standard output and nothing is read from the standard input. This means that Packdev will always abort if there are any problems (read/write error, ^C pressed, write to existing file etc.) and reports no error text. It's disabled by default.

Example

Q (No input and output)

1.23 NOCONFIRM (or NC)

If this parameter is set, PackDev will never ask for user input, i.e. it will immediately start with the specified action without waiting for confirmation. When errors occur, PackDev will always abort (see also QUIET). The difference between QUIET and NOCONFIRM is that NOCONFIRN doesn't disable output, but QUIET does. This is useful when starting PackDev with a file as standard input or when starting it from another program. It's disabled by default.

Example

NC (No input)

1.24 VIEWFILE

This keyword will make PackDev output a file's filesystem information. It must be specified along with a file name and nothing else.

Example

VIEWFILE foo.bar

1.25 VIEWFILESYS

This keyword will make PackDev output some information about a
filesystem
. It
must be specified along with a
filesystem
 name and nothing else.

Example

VIEWFILESYS DH0:

1.26 TESTFILE

if PackDev is started with this parameter, along with a file name, the
complete archive is read for testing purposes. It must be specified along
with a file name and nothing else.

Example

TESTFILE foo.bar

1.27 BLOCKLIST

If this parameter is set along with a file name, an ASCII block list is
generated, one line for each block with the block number in decimal nad
hexadecimal notation. This parameter cannot be set along with VIEWFILE and
VIEWFILESYS.

Example

BLOCKLIST foobar

1.28 PASSWORD

Setting this parameter allows usage of the
XPK
 packers that are able to
crypt the data. It can be used in combination with creating an archive if
an
xpk
 packer is used that supports crypting. When extracting or ←
testing
a crypted archive you must specify the password or decrypting will fail.
Also note that there are countries where crypting data is not allowed or
restricted. So if you live e.g. in the Iran, be careful that your head
isn't chopped off...

Example

PASSWORD YouWillNeverGuessThis

1.29 Examples

Reading the disk in DF0: and storing the data into RAM:disk.pkd ↩
, packing
it with the SHRI algorithm with best efficiency:

```
PackDev DF0: RAM:disk P SHRI.100
```

Reading the disk in DF0: and storing the data into RAM:disk.pkd, packing
it with the SHRI algorithm with default efficiency:

```
PackDev DF0: RAM:disk P SHRI
```

Crypting DF0: to RAM:Secret.pkd, using the IDEA algorithm:

```
PackDev DF0: RAM:Secret P IDEA.100 PASSWORD StupidPassword
```

Decrypting RAM:Secret.pkd to df1:, a password is assumed to be needed:

```
PackDev RAM:Secret DF0: PASSWORD StupidPassword
```

Writing data from DH0:dh1data.pkd to DH1:

```
PackDev DH0:dh1data DH1:
```

Writing data to an unformatted floppy disk (DF1:), not asking for user
input, nonstandard parameter order, creating a block list file.

```
PackDev ETDF NI FROM DF1: TO DH0:disk BLOCKLIST ram:disk.blocks
```

Viewing the header of a .pkd file (ram:abc.pkd; not present: ram:abc):

```
PackDev VIEWFILE ram:abc
```

Viewing
                device
                 information of DH0:

```
PackDev VIEWFILESYS DH0:
```

Testing archive integrity of foobar.pkd, an encrypted archive

```
    PackDev TESTFILE foobar PASSWORD YohMan
```

1.30 Label Buffers

Label buffers are an extra storage space on disks that are ←
    accessed with

trackdisk.device
. Each block on such a disk is associated with a label
buffer that can hold 16 bytes. As far as I know label buffers are not
needed to be implemented into other disk
devices
 which are to work with

filesystems
 (i.e. diskspare, scsi etc.). It seems that it's not necessary
to bother about label buffers when writing to something else than standard
floppies (i.e.   with
trackdisk.device
), but special label buffer handling
is supported by PackDev nevertheless.

1.31 Devices, Handlers, Filesystems

What are devices, handlers and filesystems ?

The name "device" has a twofold meaning. First, all hardware drivers
(virtual or not) named xxx.device are called device. They are not
controlled by DOS, but by exec.library, the hard core of the Amiga OS, and
are closest to the actual hardware (there are also resources which are even
closer, this topic is meaningless here, and so they are treated as
nonexistant in this context). Second, DOS filesystems (e.g. DF0:),
handlers (e.g. PRT:), volumes (i.e. disk names) and assigns (e.g. LIBS:)
are also called devices.   Because PackDev only supports (exec-) devices
that can be accessed by a standard set of
trackdisk.device
 commands
and that have a filesystem on top of it. Only DOS-devices that represent a
filesystem are needed and these DOS-devices are called filesystems in this
guide.

A filesystem is a link between DOS and block-based devices. It is like a
database that generates a kind of tree with directories as branches and
files as leaves.   Most filesystems support links which can be roughly
described as ropes that are tied to a branch and point to a leaf or a
branch elsewhere, even on another tree (i.e. another filesystem). There
currently are 3 basic filesystems for the Amiga: OldFileSystem (FS),
FastFileSystem (FFS) and FastFileSystem with directory caching (DCFS) .
You may have also heard of Internatinal filesystem. This could be

interpreted as a subspecies of the three basic filesystems which support
correct upper-/lowercase handling of non-US letters, e.g. Umlauts or
accented letters for names of files, directories or links.

Below is a simple representation of how DOS communicates with several types
of I/O devices and device-like (in DOS terms) I/O drivers PackDev supports
any filesystem that is marked with an asterisk (*). Note that CD0: and
RAM: are not supported because they are special cases of devices with a
filesystem and they don't communicate with DOS in the way mentioned above.

```
          DOS-device           Handler

      |--- Filesystem DF0: ---
             trackdisk.device
             , unit 0              (*)
      |
      |--- Filesystem DF1: ---
             trackdisk.device
             , unit 1              (*)
      |
      |--- Filesystem DH0: --- scsi.device                           (*)
      |
      |--- Filesystem RAD: --- ramdrive.device                       (*)
      |
      |--- Filesystem FF0: --- fmsdisk.device                        (*)
      |
      |--- Filesystem RAM: --- RAM (ram-handler for Kick up to 1.3)
      |
DOS -|--- Filesystem CD0: --- cdrom-handler
      |
      |         etc.
      |
      |--- Handler CON: ------ console.device
      |
      |--- Handler SER: ------ serial.device
      |
      |--- Handler PAR: ------ parallel.device
      |
      |--- Handler PRT: ------ printer.device
      |    (port-handler)
      |
      |         etc.
```

1.32 trackdisk.device

              This is the standard
              device
               for Amiga floppies. It represents the way a
track-based
              device
                has to behave with a set of basic commands. It is the
only
              device
                that is guaranteed to support

```
        label buffers
        .  That means that

        filesystems
          don't  use
        label-buffers
        ,  they are probably interesting when
considering trackdisk.device.
```