xfs

COLLABORATORS

	TITLE : xfs		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

xfs

1.1 xfs user (friendly) guide

xfs - XFileSystem © 1998 frans (francis swift)

user guide

1.2 disclaimer

```
   This program is provided "as is" without warranty of any kind,
either  expressed  or  implied, including, but no limited to, the
implied warranty of fitness for a particular purpose.  The entire
risk  as  to the results, reliability and performance of this
program is assumed by you.
```

1.3 distribution

```
   This program is freely distributable. Distribution is allowed if
the following conditions are met:

1. Program and documentation must not be changed in any way,
```

except for archiving with an archiver, for which the corresponding unarchiver will work on any CPU type used in Amigas at the time of distribution.

2. Program, documentation and the icons for the program and the documentation must be distributed together.

3. The file names of the program, documentation and icons must not be changed. However, you ARE allowed to change the default tool of the documentation icon. You are also allowed to shapshot the icons if you really need to.

4. Only a nominal and small fee for copying is charged, unless some sort of media is supplied with xfs, in which case it is allowed to add the cost of that media to the charge.

5. xfs or parts of it must not be sold in combination with or as part of commercial software without my written permission.

1.4 introduction

XFileSystem (xfs) is a package consisting of a filesystem (xfsd) and an exec device (fd.device) that enables your amiga to recognise multiple disk types.

Basically I had written two handlers (QL and Spectrum), and had nearly completed another (an 8.3 style PC one) and I just thought it would be more convenient to integrate them. Of course, a low level device which could understand different disk track encoding was also necessary. Luckily I was already writing one (to read 256/512/1024 byte sectors), and that eventually became known as fd.device.

Eventually, when I have finalised the internal API (just at this moment undergoing a major revision to adapt it to Mac disks), I hope to release the source code, or at least enough to allow further additions to the disks that are recognised.

I should make it clear at this point that internally the handlers don’t know they’re running on an amiga, as they only see a ’virtual’ machine interface. This will allow me to port the code to other environments. It should be possible to get it to run on the PC, maybe even the QL.

1.5 system requirements

xfs is designed to be compatible with the amiga operating system from WB1.3 upwards.

Note that there are two versions of fd.device, the current default is a slightly naughty hardware-hitting one. The other one (which

```
can be found in devs/storage) is more system friendly, but has proved to
be slightly incompatible with some mac/pc disk controllers, so don't
use it unless you have to.
(XL DRIVES: you _must_ use the system friendly fd.device on XL drives,
because the hardware-hitting version bypasses the highdensitypatch.)
```

1.6 installation

```
Note that in the instructions below, FD0 / FD1 refers to the
mountlists for xfsd using fd.device, and FX0 / FX1 if for some
strange reason you insist on using mfm.device.

Quick installation notes:

FIRST

  Open the drawer "L" in the XFS directory and drag the program
  "xfsd" to your system "L:" drawer.

  Open the drawer "DEVS" in the XFS directory and drag the program
  "fd.device" to your system "DEVS:" drawer.

THEN for Workbench 2.1 and later

  Drag the icons FD0:, FD1:, FX0: and FX1: into your system
  "Storage/DOSDrivers" drawer or "Devices/DOSDrivers" drawer.

  Double click the FD0: or FD1: to mount the volume using fd.device
  Double click the FX0: or FX1: to mount the volume using mfm.device

OR for old Workbenches

  Copy the updated mount.xfsd file to DEVS: and add a line to
  your startup for each drive you want to connect to XFS, as follows:

  MOUNT FD0: FROM DEVS:mount.xfsd  ;to use fd.device
  MOUNT FD1: FROM DEVS:mount.xfsd

  or

  MOUNT FX0: FROM DEVS:mount.xfsd  ;to use mfm.device
  MOUNT FX1: FROM DEVS:mount.xfsd
```

1.7 usage

```
  Usage should be (generally) fairly transparent, but here are
a few disk type specific notes.

             readdir
  read
 write
 delete
 rename
 readpart
 format
  amiga         +       +       +      +       +        +
  msdos/w95     +       +       +      +       +        +
  macintosh     +       +                               +
  ql            +       +       +      +       +
  minix         +       +       +      +       +
  archimedes    +       +
  cpm           +       +
  1541/1581     +       +
  spectrum      +       +


  fd.device         Workbench       x-code
  file.device       ZIP / JAZ drives      other...
  devd / DEV:
```

1.8 readdir notes

```
   By 'readdir' I mean that commands like List, Dir etc should
work, and hopefully directory utilities should as well.

   The Examine/ExamineNext calls are supported, but currently
the ExamineAll packet is disabled, which means it should work ok.
Yes, you did read that correctly.  If a handler doesn't support
ExamineAll the Dos emulates it via Examine and ExamineNext.
I'll re-enable it when it's fixed.

   Note that (internally) file names are converted from however
they are stored on disk into utf-8, which is a byte-sequence
implementation of unicode. This should allow any filename to
be read correctly. Of course, the amiga is limited in its own
character set, so the internal utf-8 representation is further
converted to either normal 8 bit ascii (iso) if that is possible,
or to a sort of hybrid www/utf-8 sequence, eg

Unicode  Entity Name   utf-8 sequence   in amiga directory listings

0x0192     &fnof;        0xC6,0x92         %C6%92

  Note that standard AmigaDos commands don't allow '%' characters
in filenames, but any third party programs like NewList, Ls etc,
and directory utilities like DirWork should have no problem
accessing these files through xfsd.

   As an added bonus, I have included my program x-code which
is a utility that converts between the various character
representations shown above.
```

1.9 read notes

By ’read’ I mean that commands/programs that open files to read should work, so you can use Copy, Type etc, and various viewer programs. For this to work, Lock/Open/Read style calls are supported.

In general, xfsd does no translation of contents, as I don’t consider a disk handler the proper place to be converting files from one locale, code page, cr/lf etc to another system. The only (slight) exception to this is for spectrum disks, where the files are converted to ZX82 format, which is in any case just a ’wrapper’ format, and doesn’t change the contents.

1.10 write notes

By ’write’ I mean that commands/programs that open files to write should work, and also that (where applicable) you should be able to create directories, so you can use Copy, MakeDir etc. See also usage >> read.

1.11 delete notes

By ’delete’ I mean that commands/programs that actually remove files / directories should work. This is not the same as just being able to write to a disk, so when a new file system is being implemented you may find it is done in stages, with ’write’ access available but not ’delete’.

1.12 rename notes

’Rename’ is actually two functions, firstly to just rename a file in place, that is in its current directory, secondly to actually move the file to another directory. Because these processes are (usually) quite different, depending on the file system being used, they may not both be implemented, since the ’rename in place’ function is usually sufficient for most purposes.

REMEMBER if you are using a directory utility and you move files from one directory to another directory on the same disk, this will be implemented using the ’rename’ function, so if the ’rename to move’ part of this command is not available, you will get an error. Currently the PC, Amiga and Minix handlers DO work correctly in this respect, but the QL handler does only the ’rename in place’ operation.

1.13 read partition notes

Currently, partitions as used by the PC, Amiga and Macintosh are understood by xfsd, but only if the mountlist points to where the partitioning info is actually stored on the disk (usually on block 0, or somewhere on the first track for the Amiga).
If the mountlist points directly to where a volume starts, this facility is ignored.

At the moment, if xfsd finds partitioning info, it finds all the partitions it can understand, and may even create internal links to them, but it only mounts the FIRST partition, as the internal fork() facility to create more processes for extra dos device entries is not yet complete.

Note that the current implementation of the Amiga partition reader only looks at the first 9 sectors to find the partition blocks.
This is a compromise based on the fact that exactly the same routines are used for all media, and I didn't want the partition scanning to go further than the first track on a floppy.
Yes, that does mean that partitioned floppies will work.
9 sectors should usually suffice, but may not in unusual cases (for instance, if there is some sort of MSDOS partition manager taking up lots of room in the first cylinder before the Amiga information, or you may simply have told whatever partition utility you were using to leave lots of space before the RDB).
Anyway, this implementation will probably change sometime soon.

1.14 format notes

As noted elsewhere, some of the handlers do actually have the facility to format disks built in to them, but there is currently no way to access this. Remember, when you format a disk, whatever format program you are using will (usually) use information taken from the mountlist to access the low level exec device directly, to format the tracks on the disk. It will then just send the ACTION_FORMAT packet to the handler to tell it to write out the file system dependant information. Now, this is fine when the handler is only capable of writing one kind of filesystem, but on xfsd, which one should it write?. Well, actually, there is a possible point of access available, as the format packet is sent with a parameter, usually used to select between the different types of Amiga disk, OFS/FFS etc, and perhaps this will be how the file system selection process is implemented. Eventually.

1.15 ql notes

Since this was the first handler I wrote, most things are complete. Only thing left to do is that rename only works as an actual rename, that is it doesn't allow you to rename things to move them.

```
   Directory listings of ql disks show the following info:

   Names - can be up to 36 characters on the ql, whereas the
           amiga (officially) allows only 32. I just ignore
           this because most amiga programs can accept up to
           around 40 characters. Still, a possible source of
           random crashes, so I'll be looking at a fix.

   Size -  Doesn't include the (hidden) 64 byte file header,
           so files may appear to take more blocks than you
           would think.

   Flags - r/w/d ignored at present, 'e'xecutable flag is used
           to indicate a type 1 (program) file, with the
           data space held in the comment.

   Dates - translated from base 1961
```

Note that ql directories are implemented in such a stupid way I can scarcely bring myself to explain it. Oh go on then... The whole path, including separators, is stored! So if you have a directory with for example a 10 byte name, the files inside can only have 25 character names (36 - 10 - 1 separator). And the deeper into the directory structure, the worse it gets...

1.16 spectrum notes

Currently read-only, this handler is different from all the rest in that the files are translated into a common format as they are read. The format used is called ZX82. This has the advantage of allowing, for instance, the use of datatypes to automatically recognise particular Spectrum file types, in particular screen shots.

The spectrum has a veritable cornucopia of different disk types, falling into two main categories. Here is a list of some of the types that should be usable.

What	Where	Who
BetaDOS	Spectrum and +D	BetaSoft
GDOS	Sinclair Spectrum	MGT Disciple Interface
G+DOS	Sinclair Spectrum	MGT/Datel +D interface
MasterDOS	SAM Coupe	MGT and BetaSoft
SAMDOS	SAM Coupe	Miles Gordon Technology
Opus	Spectrum/Opus Discovery	Opus Supplies Ltd

In addition, the Spectrum +3 uses its own version of CP/M. This, of course, is handled via the CP/M section of xfsd, and means +3 files (currently) are NOT in ZX82 format.

I have now (from V2.13b) included some support for the original Opus Discovery disk format (180k, 40 track, 1-sided, 18 sectors of 256 bytes), though at present I only have one disk to test it on.

1.17 msdos notes

Since this handler has been getting the most attention, it should be the most reliable. Just a few things you should note.

Accented characters in ’short’ names (ie 8+3) are always translated (when reading a directory) as code page 850. Of course, characters in ’long’ names need no translation, but where they are not supported on the amiga, they will be displayed in a sort of combined utf-8 / ’www’ format, eg %C6%92 for the curvy letter f, (also known as &fnof;).

The process followed to generate ’short’ names (aliases) from ’long’ names is similar to W95, except that in order to get rid of the need for code pages, the ’short’ names will be strictly 7-bit ascii, with accented characters mapped onto their uppercase unaccented equivalents. Again, this doesn’t affect the proper ’long’ names.

Note also that the ’short’ names appear in the comment field, what’s more they can be directly changed there. This can be useful if you want the ’short’ names to appear as they would on other handlers. For instance, if you save a disk icon onto a pc disk using xfsd, the name ’disk.info’ would be saved as the ’long’ name, but since this is not a legal ’short’ (ie 8+3) name, a new one would be generated, in fact DISK~1.INF would be used. Now, since other handlers expect the disk icon to be called DISK.INF they wouldn’t recognise it, so just change it via the comment, eg

```
FileNote FD0:disk.info DISK.INF
```

(if the disk was in FD0: of course).

1.18 archimedes notes

From xfs version 2.13 both ’E’ AND ’D’ type disks are supported (although D type support is rudimentary at present).

Note that since the only info I have on the archimedes consists of a two Basic programs written by Richard Kettlewell in SuperBasic for the ql, as supplied to me by Simon Goodwin, support is limited to whatever those programs could do. In particular, dates were not supported, so I don’t know how they work, so the dates on directory listings will be wrong.

1.19 amiga notes

Why an amiga handler?

Well, this allows you to read/write FastFileSystem disks on WB1.3, and, because fd.device is used to decode the tracks, you can transparently read diskspare formatted floppies.

In addition, having amiga support in xfsd lets you have just one mountlist entry for other types of removable media that may be formatted in a variety of ways, like ZIP drives for instance.

Currently the amiga handler is read / write on all media with the exception of Directory Cacheing (DC) disks, which will remain read-only until I fix that part of the handler (next release ;-).

V2.11b2 contains a completely re-written sector allocation system (to allow writing to large media) and consequently will require substantial testing before I can recommend it for anything other than experimental use. But it should be safe just reading. Hopefully it will be thoroughly tested by the next release.

1.20 minix notes

From xfs version 2.14 this handler should be virtually complete, including renaming files to move them between directories.

This handler is mostly based on info taken from linux source.

I also hope at some future date to support more popular linux filesystem types like ext2, and later still BerkeleyFastFileSystem.

Note that minix support has only been tested on 720k floppies.

1.21 macintosh notes

Characters in file names are considered to be MacOS_Roman.

Understands mac-style partitions, so ZIP media partitioned and written on a mac should be readable.

Currently reads only the data fork of files.

Hopefully writing support should appear quite soon. Although this section of the handler is based on the hfsutils package by Robert Leslie, it didn’t prove possible to just drop it into xfsd, only the btree routines being directly usable, others needing major changes to fit in with the way xfsd works. Nevertheless, having an example of working code shortened the development time considerably (as did actually having a macintosh).

1.22 cpm notes

Bit of a mistitle this, as the number of CP/M disks supported is only two out of the many hundreds of types, specifically the PCW and the +3. At least it works on the PCW disks I’ve got, and on the one +3 disk I have as well.

Hopefully it won’t take too long to implement the ZX82 file format for +3 disks, now that I have an example disk to experiment with.

1.23 1541 / 1581 notes

Currently, this handler is able to read 1581 disks directly using fd.device (ie FD0:), and 1541 disk images (.D64) via file.device (FILE0:).

Access to 1581 disks required some modifications to fd.device to take account of the side skewing that is used on these disks. (Basically, the top is on the bottom). Because the 1541 disks use speed zones (different speeds at various points across the disk) it is unlikely that fd.device will ever be able to read them. So if you want direct access to 1541 disks you will have to use a hardware solution, eg Catweasel.

I should also mention that in an earlier version of fd.device I included a routine to pulse-width-modulate the disk motor, (basically that means turning the motor on and off very quickly), to see if it was possible to read disks that used speed zones. This was intended for reading Mac 800k disks. Unfortunately, the built-in disk hardware in the Amiga didn’t like this and cancelled the data stream every time the disk motor was turned back on. HOWEVER, I never tried this technique with old 5.25 drives, so there is still a chance that it could be possible to directly read 1541 disks from a 5.25 disk drive connected directly to the Amiga. Otherwise, the only option is to slow the drive down mechanically so the bit rate is within the parameters of the Amiga hardware - see Aminet misc/emu 1541.lha.

1.24 fd.device notes

fd.device

This is a standard exec device designed to read multiple track formats. At present it can read and write Amiga style tracks and (from V42.84) can read/write diskspare.device disks, and MFM tracks (any size sector from 128 to 16k, any number of tracks and sectors). It may also at some future date be able to access GCR tracks.

From version 42.84, the initial format of fd.device may be set by passing the size in the open flags. Note that this is only the format fd.device uses initially, and is discarded when the disk is changed, so it really can’t be used with xfsd, and is intended for use with DEV: to allow you to copy disks.

example usage with DEV:

```
copy diskimage.adf DEV:fd.device,0,880
```

...will write out an Amiga disk image, the flags value of 880 telling fd.device the geometry to use. Here are some more examples.

```
gzip -c - <DEV:fd.device,0 >ram:temp.adz
```

...compresses an amiga disk to an .adz, (the dash on its own is to force gzip to use stdin, ie indirection, the -c does the same for stdout). To write it out again...

```
gzip -d -c ram:temp.adz >DEV:fd.device,0,880
```

Note the ,880 to force the geometry.

Actually, the gzip compress example above will compress ANY disk that fd.device can read, not just Amiga disks. So for any disk...

```
gzip -c - <DEV:fd.device,0 >ram:temp.gz
```

...would read it in and compress it, then to decompress it...

```
gzip -d -c ram:temp.adz >DEV:fd.device,0,720
```

...for a PC 720k disk, (just change the ,720 for other formats).

See further examples / explanation in devd / DEV:

Some background info - Format programs:-

Note that format programs don't ask the low level device what format it COULD write to a disk, they expect just ONE style of track layout. Catweasel / multidisk.device gets round this by setting the layout when the device is opened, using the flags field. This of course means that you must open multidisk.device multiple times, once for each type of disk, and have a mountlist for each.

ALSO fd.device inserts an input stream handler to allow you to force it to re-read the disk (sometimes necessary if it gets confused), at the same time sending a disk change interrupt to whatever dos handler is using it.

The disk change key combination(s) for fd.device are

```
  left_alt + numeric_pad(unit#)
```

ie left alt with numeric pad 0 would force a disk change on unit 0.

1.25 file.device notes

file.device is low level exec device to enable you to access files as if they were actually disks, allowing you to keep many floppy disks as disk image files on you hard disk, allowing quicker access.

The current version of file.device allows not only flat file disk images to be used (eg .ADF), but also PCTask / janus (ABOOT) hard disk files, d64 files (read-only), and DMS crunched files (also read only).

I hope at some point to implement .ADZ file access.

There are already quite a few disk/file devices around, the most popular being fmsdisk.device, and you can try them all with xfsd if you like.

The reason file.device was written was convenience, as when it is opened, it allows you to select the disk/file via a requester, currently using reqtools.library (if available) to enable it to work on WB1.3.

Whenever you want to change disk/file, there is a special key combination you can use. The key combination(s) for file.device are

```
  left_alt + left_shift + numeric_pad(unit#)
```

ie left alt plus left shift plus numeric pad 1 would force a disk change on unit 1.

Disk / file images are by default first looked for in a directory called devs:file, but you can use the requester to access them anywhere, even CDROM.

ALSO to enable you to move over from fmsdisk.device, if there is a file called unit0 in devs:file, and you open file.device unit 0, this will be automatically selected, and NO requester will appear.

BUT FIRST if there is an environment variable ENV:file/unit0, it will be read first and its contents used as the file name for unit 0.

ALSO file.device will go through the above process of file selection all over again EITHER when you use the key combination to tell it to do so OR when it receives the CMD_RESET message. Since you can use the 'devio' command (supplied) to send this, you should be able to change disk/file via a script that sets the ENV: variable then sends the CMD_RESET message. Or even use ARexx/DOS script to do it.

1.26 dev: handler notes

devd is a device handler, pretty much the same as many others, designed to give a unix style access to low level exec devices as though they were files on a pseudo DOS device (DEV:).

Once mounted, you can access disks, etc, as though they were files. For instance..

```
copy DEV:fd.device,0 ram:temp

..would copy the whole of whatever disk was in drive 0 into a file
in ram: called temp.

  From devd version 1.02b a FLAGS parameter is allowed after
the unit number, so, for instance,

copy DEV:multidisk.device,0,16 ram:temp.d64

would read a 1541 disk into a file called temp.d64 in ram:

Of course, fd.device automatically determines the geometry, so it
doesn't require a flag value to set the disk size when reading.
However, to write the disk out again, possibly to a blank disk
(which therefore has no real geometry), fd.device must be told
somehow what geometry to use. From fd.device V42.84 this is
accomplished again by use of the flags field, but using the size
of disk as the value. So, if you copy a PC 720k disk using...

copy DEV:fd.device,0 ram:pcdiskimage

...you can write it out again using...

copy ram:pcdiskimage DEV:fd.device,0,720

In fact, the flags value is 5 decimal digits, fnnnn, where

  if f is 0, nnnn is size in k, using 512 byte blocks

  if f is 1-3, nnnn is size in blocks, and blocksize is 2**(f+7)

  if f is 4, side skewing is used (for 1581), NOTE: nnnn now size in k

  if f is 5, sector numbers start at 193 (for CPC), NOTE: nnnn size in k

so values of 720, 800, 880 should work as you would expect (the leading
digits taken as zeroes), other values may look a little strange, eg

10720 is a 180k disk (720 blocks of 256 bytes) (eg spectrum opus)

40800 for a CBM 1581 side skewed 800k disk

30800 for Archimedes disks (800 blocks of 1k)

50360 for CPC 360k disks (sector numbers starting at 193!)

Note there is some redundancy in this scheme, 21440 == 720

Please don't expect weird combinations to work; 10683 will NOT write
out a .D64 disk image as fd.device can't access them. Use the above
example using multidisk.device.



Some background info - how the Copy command works:-
```

The Copy command (at least the normal DOS one) is fairly simple. It
first asks the source device (DEV:) if it is a filesystem. Of course,
it isn't, so the copy command then just opens the source, and keeps
trying to read 64k chunks, writing them to the destination, until
it hits the end-of-file. NOTE: other copy commands don't ask the
handler if it's a filesystem, and insist on trying to do things like
obtaining FileLocks, Examine'ing etc. To use these you must tell them
NOT to do anything clever, but to access the handler DIRECTLY. Or
you could tell the programmer to use the IsFileSystem() call...
Anyway, when the copy command tries to open the source, the DEV:
handler then opens the exec .device, in this case fd.device, unit 0,
(as indicated by the ,0 after the device name), and asks for the
geometry, using this to determine the size of the 'pseudo' disk/file.
In the case of fd.device, the geometry is automatically determined,
thus using the DEV: / fd.device combination, you can use the command
line shown above to copy ANY size disk into an image file.

1.27 Workbench notes

You should note that disk icons only appear on the Workbench
when VOLUME entries appear for them in the internal dos list.
Unfortunately, you can't have two volumes with the same name,
so if you have another handler mounted that has e.g. recognised
a pc disk AND xfsd also recognises it and they BOTH put volume
entries in the dos list, you WILL see two icons appear. USUALLY
this doesn't immediately crash the machine, but opening the icon(s)
or removing the disk may well do so.

So? Well, in order to let you try xfsd without having to take
all your other handlers out, you can disable the mounting of
the volume entries via the Flags entry in the mountlist.

Here is how to control it (excerpt from mount.xfsd)

```
8<-----------------------------------------------------------------

   Now Flags = 0 stops the volume being mounted by name.
   This prevents the WorkBench recognising the disk but still
   allows you to access it using the device name (ie FD0:).
   There is no longer a separate handler called xfsd-ami.
   To get the volume to be mounted (and the Workbench icon
   to appear) the bits in the top byte represent each different
   section of the handler.

                          8 4 2 1 - 8 4 2 1
                          | | | |   | | | |
             ql ----------+ | | |   | | | |
             spectrum ------+ | |   | | | |
             msdos -----------+ |   | | | |
             archimedes --------+   | | | |
             amiga -----------------+ | | |
             minix -------------------+ | |
             mac -----------------------+ |
             cpm -------------------------+
   So if you wanted the handler to only mount MSDOS or MAC
   volumes you would use Flags = 0x22000000

   REMEMBER, even if the volume isn't mounted, the disk is
   still accessible via the device (eg FD0:).

   ALSO to disable a filesystem completely, the same bits in
   the next byte can be set, so to disable the amiga file
   handler section of xfsd use Flags = 0x00080000

   ALSO only the top word of the flags is used by the handler,
   this being masked off before the flags are sent to the low
   level device, so you can put any values you like in the lower
   word, whatever is required for the .device you are using.
   This means you can use the xfsd handler with mfm.device, or
   with Catweasel if you like.

8<----------------------------------------------------------------
```

I know that those of you who use the Workbench for everything would really like a more sophisticated support for disk icons, for example giving different disk formats a different default icon, but this isn't possible at the moment. However, for the disk types that are writeable using xfsd, you can of course copy any disk icon of your choice onto the disk, and the Workbench will read it when you insert the disk.

1.28 ZIP / JAZ drive notes

There is an example mountlist called ZIPX in devs/DosDrivers, but you MUST remember to set the device/unit (in the tooltypes).

Also, if you are using another ZIP mount entry at the same time you MUST also read the explanation (in Workbench) about setting the flags field correctly (ie to not let xfsd put up a disk icon).

There is also now an example mountlist for JAZ drives called JAZX in devs/DosDrivers, again you MUST remember to set the device/unit (in the tooltypes).

1.29 x-code usage

When using xfsd to list file names, any characters unavailable on the amiga are printed in a sort of hybrid www percent / utf-8 style, so for instance, if a file name includes the 'L with a slash' character in it (otherwise known as &Lstrok;), it is displayed as %C5%81.

To assist you with these strange codes x-code can convert and display the various code formats. So...

```
x-code &Lstrok;

prints out...

&Lstrok;   0x0141  %C5%81

where 0x0141 is the unicode representation, and...

x-code %C3%B6

prints out...

&ouml;   0x00F6   %C3%B6

V1.01
-----
x-code now lists ALL entities that even partially match, so

x-code &Z

prints out...

&ZHcy;          0x0416          %D0%96
&Zacute;        0x0179          %C5%B9
&Zcaron;        0x017D          %C5%BD
&Zcy;           0x0417          %D0%97
&Zdot;          0x017B          %C5%BB
&Zeta;          0x0396          %CE%96
&Zgr;           0x0396          %CE%96

Note that it's still case-sensitive, so

x-code &z

would be required to print out the lower case entities
```

1.30 other bits and pieces

```
As you can see, this guide has been thrown together rather quickly,
and as a result certain details may have been missed out.  I suggest
you examine the history for more details.

-------------------------------------------------------------------------

POWER XL DRIVE
--------------
You MUST use the system-friendly version of fd.device with the XL drive.
This may be found in devs/storage.

RAWPATCH / MFM
--------------
Two utilities to experiment with raw disk data. So obscure as to be
almost unusable. Use rawpatch to read in raw track data, then mfm to
decode it.
NEW: I've changed mfm so that it can decode tracks even if you didn't
```

```
use the -W (word sync) flag. This was necessary for me to better
analyse the contents of tracks (since without word sync the track
starts reading as soon as possible after the index sync).
Hint: ALWAYS use -I with rawpatch (index sync). You can only get an
idea of the sector skewing if you read starting from the index pulse.
Hint: use -v on mfm to get a verbose output (or even -vv ).

DEVIO
-----
This program sends a various low level commands to whatever exec
device you specify. For instance, if you want to determine the
geometry of a disk in say, drive 0 (the internal drive), you can use
devio to send the TD_GETGEOMETRY command to fd.device, just type

devio -geometry fd.device 0

This command should work with most low level disk devices, but with
fd.device you get extra information about the interleave, skew factors etc.

CRC
---
Just a simple program to generate CRC's for files, but I've added an
option to generate a CRC for each block (by using eg -b512), which by
default is the one used on disk sectors (pc MFM that is, not amiga style).
```

1.31 todo

```
   First thing to do is get the Mac handler finished.

   Then maybe I can write a Format utility.

   Then perhaps upgrade the Spectrum handler to allow writing.
```

1.32 acknowledgements

```
   Simon Goodwin <simon@studio.woden.com>
       - keeps me supplied with spectrum/ql/cpm disks and info.
       - also for writing the original documentation.

   Mark Weller
       - for supplying Archimedes 'D' and 'E' format disks.

   Richard Kettlewell
       - for his two Archimedes (QL Superbasic) reader programs.

   William James
       - (author of Speculator) for his co-operation over ZX82.

   Robert Leslie
       - for his excellent hfsutils package.

   Chris Thornley <osu036@sos.bangor.ac.uk>
```

```
    - for e-mailing me example disk images.

Brian Gaff <briang@bgserv.demon.co.uk>
    - for sending me the +3 disk / info

Joerg Deutschle <agima@studbox.uni-stuttgart.de>
    - for sending me some 1581 disks to try out.

Plus a few others with useful bug reports/suggestions:

Dirk Neubauer      <neubauer@rz.uni-greifswald.de>
Guido Mersmann     <geit@studst.fh-muenster.de>
Andreas Paul       www.student.uni-kl.de/~apaul
Lars Nelson        <lars@infohwy.com>
Adrian Siemieniak  <sauron@pwr.wroc.pl>
Marcel Timmermans  <mtimmerm@worldaccess.nl>
Christian Kersting <kerstic@uni-muenster.de>
Marko Seppanen     <marko.seppanen@wwnet.fi>
Ken J Powell       <ken@dra.ex.ac.uk>
Michel De Meerleer <michel.demeerleer@iname.com>
Tim Jackson        <timjackson@radiolink.net>
Ralph Debusmann    <rade@coli.uni-sb.de>
Jernej Pecjak      <jernej.pecjak@kiss.uni-lj.si>
Udo Zallmann       <huz@informatik.rwth-aachen.de>
Jess Sosnoski      <starblaz@ptdprolog.net>
Roman Fierfas      <xorceror@friko.onet.pl>
Mustafa Kayikci    <mustafa@dame.de>
Anthony Doggett    <doggett@zetnet.co.uk>
Karsten Kiehn      <icke@dame.de>
Stephan Rupprecht  <stephan.rupprecht@metronet.de>
Henning Thielemann <henning.thielemann@student.uni-halle.de>
Lars Bischoff      <lars_bischoff@hb2.maus.de>
Matthew Garrett    <cavan@enterprise.net>
Johan Otterstrom   <videoking@mbox200.swipnet.se>
Andreas Rega       <arega@berlin.snafu.de>
Clifford Ramsden   <gcramsden@aol.com>
Alastair Booker    <ali@squark.demon.co.uk>
Fritz Hopfner      <a9107686@unet.univie.ac.at>
Cantaro            <cantaro@gmx.net>
Dirk Stoecker      <stoecker@amigaworld.com>
Mario Cattaneo     <cattaneo@coli.uni-sb.de>
Sitsofe Wheeler    <swheel11@bgs.bristol.sch.uk>
Rob Graham         <rgraham@vcn.bc.ca>
Jerome Chesnot     <jchesnot@noname.fr>
Flavio Stanchina   <flavio@ies.it>
Stefan Haase       <stefan_haase@linux.fl.fhtw-berlin.de>
Martin Pritchard   <mspritch@powys.gov.uk>
Ian Armstrong      <ian@iarmst.demon.co.uk>

Plus I got postcard from Heidelberg, which, after careful
examination with a magnifying glass, I managed to discover
as coming from (I think) Joerg <jc7@ix.urz.uni-heidelberg.de>.

I also got another letter from Germany, from Tobias Herold.

Perhaps I should make xfs 'snailmailware';-)
```

Your name here!

If you would, could you specify the version of xfs you are using in any messages.

1.33 the author

francis swift AKA frank AKA (in computerspeak) frans

- e-mail: (via my brother Mark)

  msw@blackpool.ac.uk

- snail mail:

  325 Charlestown Road, Blackley, Manchester M9 7BS, UK.

bug reports, suggestions, ideas, banker's drafts etc are welcome.

1.34 bugs

Most of the current 'bugs' are caused by the program being incomplete. For instance, the Spectrum handler doesn't write, but may or may not be disallowed from doing it, so weird things can happen if you try to save something.

Still, if you do have a disk which shows up a particular bug, make a disk image file of it and e-mail it to my brother Mark.
You can make a disk image file with the DEV: (dos) device and fd.device.

```
Mount DEV: from devs:mount.dev    (or wherever you keep it)

Copy DEV:fd.device,0 Ram:dodgydisk
```

Note: this is my DEV: device - others may operate differently.
(e.g. the ,0 means unit 0)

1.35 history

---------------------------- xfs215 -------------------------------

Added support for 1581 disks (read only at present). These disks sometimes have what are loosely referred to as 'partitions', really fixed length directories. These are accessed just like directories.

You can now use fd.device to 'format' disks. That is, you can use the DEV:fd.device,<unit>,<flags> pseudo file to write out disk images of a particular format. So if you have a blank disk, read it in using

```
   copy DEV:fd.device,0 ram:temp

This will copy whatever disk is in drive 0 into file 'ram:temp'.
You can then write it out again using, for instance

   copy ram:temp DEV:fd.device,0,880  ; if it is an amiga disk
   copy ram:temp DEV:fd.device,0,720  ; for a pc / ql / minix disk

etc. See further explanations in 'usage > fd.device' in the guide.

To match up with the above, devd (the DEV: handler) has been
slightly amended, as has the devio utility. You will also find
a later version of the XL friendly fd.device in devs/storage.

While rewriting fd.device I took some time to examine the way
diskspare.device generated checksums (which was the reason for
fd.device not being able to write to these disks properly). I
also implemented an additional track scan to determine whether
a disk has more than 80 tracks. So now there should be full
support for all the different diskspare permutations. There is
also a new version of the mfm utility (for use with rawpatch)
to allow track checksum verification.

--------------------------- xfs214 ------------------------------

Completed MINIX handler. Of course there may be some hidden bugs
so it should be considered just a beta version for now.

Patched disk type into dl_DiskType. Only reason it hadn't been
done till now was I thought I'd already done it. So now you will
get sensible results from Stephan Rupprecht's Info program.
Anyway, it gives me an excuse to explain the disk types that
are returned.

Amiga        DOS\0
             DOS\1    FFS
             DOS\2    Int
             DOS\3    Int FFS
             DOS\4    DC
             DOS\5    DC FFS

MSDOS/W95    MSD\0

MAC          MAC\0

Minix        MNX\1    (and later maybe MNX\2)

QL           QL5A      720k (and ED disks)
             QL5B     1440k

Spectrum     ZXS\0    Disciple
             ZXS\1    UniDos
             ZXS\2    SamDos
             ZXS\4    Opus (180k)

Archimedes   ARMD
```

```
            ARME

CP/M        CPM\2    (and later ZXS\3 for Plus3Dos)

C64         1541     (and later 1581)
```

These will (eventually, NOT YET) be the dos types passed to the
Format() command to select the file system.

Also, the file.device has been updated to allow it to work better
with devd (DEV:) so that you get the chance to select a file
before the DEV: device asks if there's a disk (file) in the 'drive'.

Now that I have some 1581 disks to experiment with I should be
able to add handling of these to xfs. Since these disks use
side skewing (quick explanation - the top is on the bottom), I
will need to modify fd.device to accept this format. Also, although
the physical sectors are 512 bytes, the logical sectors are only
256 bytes, which will make sector allocation complicated, so
for the moment support will be read only.

Finally I tidied up some clean-up routines so some random crashes
on removal of disks should disappear.

--------------------------- xfs213 ------------------------------

Added a couple of extra disk types to existing handler modules,
specifically Archimedes D type disks, and Spectrum 180k (Opus
Discovery) type disks. Note that both use unusual sector sizes,
the Archimedes D using 1024 byte sectors, and the Spectrum 180k
disks using 256 byte sectors (it's also 40 track single sided!).
Luckily fd.device sorts all this out transparently.

One more technical point. Whenever a disk is recognised by xfsd
the disk dos type is now written into de_DosType in the environment
table of the device entry in the dos list. This is to allow
anyone using the Info() call (which xfsd implements like most
handlers by returning ID_DOS_DISK in id_DiskType for disks it
recognises) to access the true disk type.

Updated crc utility to fix a slight bug.

There's a slight change to the flags setting in the mountlists,
which should be backwards compatible, see mount.xfsd.

Up till this release it was possible to open directories as if
they were files. This was a side effect of the internal
implementation which took advantage of the file-like structure
of directories in certain filesystems (QDOS,MSDOS,RISCOS etc).
I left this in so it would be possible to write external
utilities to access filesystem-specific information that would
otherwise be inaccessible via xfsd. Unfortunately, there are
too many badly written programs out there that, when given no
file name on the command line, open the empty string by mistake.
Of course, this means the current directory, so they get a
'wrong type' error. With xfsd, they would get no error, and
depending on the way the filesystem sets the file size for the

```
directory, may try to read the whole disk into memory, or 4GB,
or any random number. Anyway I've disallowed this again.

-------------------------- xfs212b3 ------------------------------

Just a slight update to file.device. I re-wrote the xDMS source
to be re-entrant / thread safe ('pure' in amiga-speak), so now
file.device, which uses the decompression routines from xDMS,
can safely be opened multiple times for multiple DMS files.
I'll probably upload the modified xDMS source with the next
release of xfs.

-------------------------- xfs212b2 ------------------------------

There was a slight bug in the previous file.device when using
PCTask hard disk files. Basically, writing could trash them.
This was a bug introduced during re-organisation to allow proper
integration of DMS file access. Anyway, it should all work now,
reading/writing PCTask hard disk files (apparently the same format
as janus hard disk files), and reading DMS files. To implement
the DMS facility I used the source from xDMS (by Andre de la Rocha)
to replace my original routines (which called DMS externally).
The current xDMS decompression source is still very similar to
the original LhA decompression routines (written by Masaru Oki
and Haruyasu Yoshizaki), which are not re-entrant, so at present
you can only have one DMS file in use by file.device (ie only one
unit of file.device can be accessing a DMS file - other units can
still access other types of disk file). I have recently re-written
these LhA decompression routines (for another project) to make them
re-entrant, so the next release of file.device should allow any
number of units to access DMS files. Hopefully, it should soon be
possible to transparently access .ADZ files as well.

There is a mis-match between the current version of the devio
utility and the current release version of fd.device. This only
affects the -geometry option, and is caused by a mis-match in the
private structure used to pass back the low-level info (about
sector numbering and skew factors). This will be fixed next time.

Ther was a slight bug introduced by internal rationalisation of
the utf-8 / unicode routines which caused QL comments (which hold
task space info) to be messed up. Should be ok now.

-------------------------- xfs212b ------------------------------

Changed file.device to allow it to skip the first sector of PCTask
hard disk files, so you can use FILE0: (xfsd / file.device) on them.
Also new DOCS for file.device in the guide (->usage->file.device).
Oh yes it also sends back a sensible geometry for .d64 files. This
was necessary to implement a C64 disk handler.

1541 disk images may now be accessed via FILE0: (xfsd), but only to
display directories at present. You will of course require Catweasel
to access the disks directly. This is intended to eventually become
a combined 1541/1581 handler (when I get some 1581 info).

devd (the DEV: handler) can now accept a third parameter for FLAGS
```

```
after the unit number (ie DEV:multidisk.device,0,7) to enable
you to use it with Catweasel. Oh yes, devd is now mentioned in the
guide (->usage->devd).

The guide has been improved, mainly the usage section.

-------------------------- xfs211b2  ----------------------------

I’ve completely re-written the sector access / allocation on the
Amiga handler, so as to allow you to write to media larger than
floppy disks. Unfortunately I haven’t had time to test it, so I
don’t recommend you use it for anything crucial yet, just media
you can afford to re-format. Oh, and I found a few things were
missing in the directory cacheing (DC) part of the handler, so
I’ve made DC disks read-only for the moment till I sort it out.
Also, for some reason the utf-8 internal names were being sent
to the Amiga name hashing routines, causing all files whose names
contained accented characters to be put on the wrong hash chain.
If you have any disks written using this I suggest you use the
old version of the handler to copy the files off the disks then
use 211b2 or later to write them back (if it works!).

No further additions to any other handlers, ’cause I’ve been
adapting various compression routines, to see how easy it would
be to add archive-handling to xfsd.

I might as well mention that I had hoped to replace the rather
awkward DMS uncrunching in file.device, but I couldn’t get access
to the source on the Aminet. Oh, didn’t you know file.device could
read DMS files? Well I can’t recommend it. Or DMS for that matter.
But you will need both, as file.device calls DMS to do the work.

-------------- xfs211b (only released on cover cd) --------------

MAC handler does something at last. You can now do directory
listings and read the data forks of files.

Temporarily disabled ExamineAll support so that ExamineAll should
now work properly. Yes you did read that right. If a handler
doesn’t support ExamineAll the Dos emulates it via Examine and
ExamineNext. I’ll re-enable it when it’s fixed.

On the QL, accented characters should now be completely usable
in filenames, previously they would only read correctly.

Oh, and I’ve upgraded the x-code utility (to V1.01), and I’ve
replaced the geometry program with a more general device command
utility called devio (no docs for it but it’s simple to use).

Oh, there seems to be some confusion as to the names of the
mountlists. I’ve made sure they’re all named in the same way
(devs/mount.#?).

Finally, I’ve changed all the flags fields in the mountlists
to make all volumes appear on the Workbench.  So if you use
xfsd at the same time as another handler for the same disk,
something strange may happen (unless you edit the flags field
```

to stop xfsd mounting volume entries - see mount.xfsd).

---------------------------- xfs210b2 -----------------------------

Amiga PARTITIONS now really looked at properly (a last minute
change to the checksum code stopped them working). Anyway, it
gave me a chance to look at MAC PARTITIONS. Unfortunately,
although the partition info is understood by xfsd, the actual
mac volumes aren't (yet).

Enhanced FINDER.DAT support to deal with names that are the same
on both mac and pc but with the case changed.

Finally squashed the bug that only cropped up on the A4000.
It only appeared in Zorro3 fast memory, so I never saw it.

NEXT RELEASE the archive name will be just xfs.lha with the
version number in the short description. There will also be
an xfs.guide (just working on this at the moment). Also
(hopefully) a working(ish) mac handler. When that's done I
can finally get round to writing a format facility.

---------------------------- xfs210b ------------------------------

Finally got round to fixing WRITING of Amiga disks (OFS & FFS).
This used to work (ages ago) but got messed up when I added a few
things. Anyway, while I was looking at the code I added something
else. Amiga PARTITIONS are now examined. Unfortunately the actual
handler code was only designed for floppies, and doesn't look at
the BitMapExtend entry, so there is a limit of around 50MB on
the volume size. Should be fixed by the next release.

---------------------------- xfs209b4 (unreleased) ---------------

Last minute bug (re-)stomped - on writing out the fat on pc disks
sometimes one extra byte of memory got trashed causing random
crashing. Second time I've fixed this!

Oh perhaps I should remind you to change the ZIPX mount entry to
match your own device / unit, otherwise it obviously won't work.
Now that's what I call documentation.

---------------------------- xfs209b3 (unreleased) ---------------

The fd.device write bug should be cured (it only affected high
density disks anyway - made the gaps too small).

There is now also a new (slightly experimental) XL version of
fd.device (fd.device.V42.77.XL). This tries to be more system
friendly, so it may also work with the raw disk images used by UAE.
If you need this or just want to see if it works, rename it to
fd.device and put it in devs:.

The reason for the delays in releasing the write version of the
PC filesystem was it kept crashing on the A4000, it worked ok
on the A500, and just suffered harmless anomalies on the A1200
(it left the disk spinning after a write). Since this A4000 bug

```
hasn't really been fixed, I can't guarantee anything yet.

Oh, the handler (from xfs209b2 onwards) DOES understand at least
the first partition on PC media - try the ZIPX mount entry.

---------------------------- xfs209b2 -----------------------------

For users who wish to use mfm.device in conjunction with xfsd
instead of fd.device, the appropriate mountlist entries and workbench
icons are now provided.

However, using mfm.device has the disadvantage of only being able
to access disks with a fixed dos-like format of 9 or 18 sectors per
track (as pre-defined in the mountlist). This will prevent you from
accessing some foreign disk formats.


Updated quick installation notes:

  Open the drawer "L" in the XFS directory and drag the program
  "xfsd" to your system "L:" drawer.

  Open the drawer "DEVS" in the XFS directory and drag the program
  "fd.device" to your system "DEVS:" drawer.

Procedure for Workbench 2.1 and later

  Drag the icons FD0:, FD1:, FX0: and FX1: into your system
  "Storage/DOSDrivers" drawer or "Devices/DOSDrivers" drawer.

  Double click the FD0: or FD1: to mount the volume using fd.device
  Double click the FX0: or FX1: to mount the volume using mfm.device

Procedure for old Workbenches

  Copy the updated mount.xfsd file to DEVS: and add a line to
  your startup for each drive you want to connect to XFS, as follows:

  MOUNT FD0: FROM DEVS:mount.xfsd  ;to use fd.device
  MOUNT FD1: FROM DEVS:mount.xfsd

  MOUNT FX0: FROM DEVS:mount.xfsd  ;to use mfm.device
  MOUNT FX1: FROM DEVS:mount.xfsd


* Note - There is a known bug in fd.device which shows up on some
   Amiga models when WRITING to disks. If you experience
   problems, you should use mfm.device until the bug is fixed
   in a later release.

---------------------------- xfs209b ----------------------------

(Never actually released)

The PC/95/NT filesystem should now WRITE as well. For files with
both long and short names the short name appears in the comment
and can be changed via SetComment / FileNote.
```

```
Short names on disk are assumed to be code page 850.

Long names (in unicode) are converted to / from amiga codes.

Any characters unavailable on the amiga are printed in www style
percent format using their UTF-8 coding, so &Lstrok; is displayed
as %C5%81.

To assist you with these strange codes a utility (x-code) is
included to convert / display the various code formats. So...

x-code &Lstrok;

prints out...

&Lstrok;   0x0141  %C5%81

and...

x-code %C3%B6

prints out...

&ouml;   0x00F6   %C3%B6

You can enter FILENAMES including the percent format characters
and the handler converts them to the correct character on the
filename on disk.

Where SHORT NAMES have to be automatically generated from the
long names, the accented characters are mapped to their uppercase
un-accented equivalents, &lstrok; (entered as %C5%82) becoming
the letter L.

Some support for MAC users is included in the PC handler, with
mac style long file names stored on PC disks in FINDER.DAT
being used if possible.

Since at present the handler doesn’t understand PARTITIONS, to
use with any media that has been partitioned the mountlist must
point directly to the volume, not the start of the partition.
If you can figure out a hacked mountlist like this you may be
able to get the handler to access ZIP drives. In any case, this
would be a temporary solution until partition support is
implemented (probably ten minutes after this release).

--------------------------- xfs208b ----------------------------

MINIX FileSystem now working (read-only).

New system for enabling/disabling the mounting of volumes, and
for disabling recognition of certain disks (no more xfsd-ami)

...see mount.xfsd for details

CPM3 Filesystem (well PCW 3.5 inch 720K disks) (read-only)
```

```
I have precisely one PCW disk and it works perfectly, so there.

Sorry, the mac handler just recognises that the disk is indeed a
Mac disk, reads the volume name, then just sits there and refuses
to do anything else. Maybe next time.

--------------------------- xfs207b2 ----------------------------
ALL NEW INTRO!!!

Well, just a few extra words.

Ther was a hash clash (try saying that quickly) in the internal
name cache (or should that be hash cache?) between the short names
and the long names. Basically, the dummy short name entries were
displacing the true long name entries, thus losing all the internal
pointers etc, so the directory listings (which don't use the name
cache) showed all the files, but the handler couldn't access them.

I'm still working on the MAC handler, based on code from the
hfsutils package on linux, written by Robert Leslie.
Hopefully, because I'm basing it on working code, it should work
as a complete read/write handler as soon as it's finished :-)

EXTRA BONUS  *** x-hfsdir ***  EXTRA BONUS

Mount DEV: (you've already got it - see the mount list for details)
then you can get a directory of a MAC disk with

x-hfsdir dev:mfm.device,1

or, if there's a subdirectory called hyper

x-hfsdir dev:mfm.device,1 hyper

and a subdirectory in there called new

x-hfsdir dev:mfm.device,1 hyper/new

you get the picture, just remember the ,1 means drive unit 1 (df1:)

--------------------------- xfs207b -----------------------------

first release!
```