lisafile.py

Read and write files inside Apple Lisa disk images from a modern host.

Put a Pascal source file into a floppy image on your PC, copy the image to a Floppy Emu SD card, and compile it on the Lisa a minute later. No serial cable, no terminal program, no typing source at the Lisa keyboard.

Proven end to end on real hardware: a program written in Windows Notepad, injected by this tool, compiled and linked in the Lisa Workshop with zero errors and no editor step, and ran.

Works on:

Image	Tags	Interleave	Typical source
400K / 800K floppy, DC42	12 bytes	none	Floppy Emu, LisaEm
ProFile / Widget, DC42	20 bytes	none (already logical)	LisaEm
ProFile / Widget, raw	20 bytes	5:1	ESProFile, Cameo/Aphid, ArduinoFile

Format is detected automatically. Pure Python 3, no dependencies.

What this is for — and what it is not for

This tool transfers TEXT: Pascal source, EXEC scripts, data you can read. `inject` writes the Workshop's `.TEXT` file layout (see Format notes below), which is the only reason the compiler accepts the result. That same layout would destroy anything binary.

It will **not** transfer a compiled program. You cannot inject a `.OBJ` file, or an executable built on an emulator, and expect it to run. That is not a limitation to engineer around — it is how the workflow is meant to go: the source travels from the PC to the Lisa, and **the Lisa itself compiles and links it**. The Workshop's compiler is fast (well under a second for a small program); the build step costs you almost nothing.

`extract` has the same scope in reverse: it is for reading text files off an image, not for pulling binaries out.

The one rule

Let the Lisa allocate. Let the host fill.

This tool never creates or renames a catalog entry, and you should never want it to. The Lisa catalog is a B-tree keyed on filename whose entries must stay in sorted order, with a parallel structure that also holds the names. Editing it from outside appears to work and then fails later, in two ways I have seen:

```
Error 972, Calling Lookup on -LOWER-SHELL.TEXT
 Entry not found in specified catalog
```

File-mgr List still prints the name, but nothing can open it. The alphabetically first and last entries keep working, which makes it look intermittent.

```
Error number 1288, Setting Fs_info in destination
Directory tree is inconsistent
```

Turns up later, on an unrelated Copy, once the damage has spread.

So instead, create the file on the Lisa, with the name you want and a size at least as large as your payload. The Lisa allocates the blocks and builds a correct catalog entry. This tool then overwrites only that file's data blocks. Nothing structural changes, so nothing can be left inconsistent.

Step-by-step walkthrough (no experience assumed)

This is the complete round trip, every keystroke, for getting a program called `MYPROG` from your PC onto the Lisa and running it. It assumes a Lisa booting the Workshop from a ProFile (volume `-#12-`), with a Floppy Emu presenting a 400K floppy image (volume `-LOWER-`), and a Windows PC with Python 3 installed.

One-time setup on the PC

1. Make a working folder, for example `C:\Users\you\Desktop\lisatest`, and put two things in it: `lisafile.py` and your floppy image (say `XFER.image`). Always work from this one folder — stray copies of the image on the desktop, in Dropbox, or left on the SD card are the single most common cause of "it still doesn't work."
2. Open a Command Prompt in that folder (in File Explorer, click the address bar, type `cmd`, press Enter).
3. Check the tool answers:

```
python lisafile.py --version
```

It should print the version. If Windows says Python is not recognized, install Python 3 from python.org and tick "Add to PATH."

One-time setup on the Lisa: create the donor file

The Lisa must create the file that will receive your source. You do this once per program name; after that you can re-inject into the same file as many times as you like.

4. At the Workshop menu, press `F` for File-mgr, then `C` for Copy.
5. Answer the prompts exactly like this (the donor just needs to be bigger than your source; `QD/sample.text` at 24 blocks ≈ 11 KB is a good default):

```
Do you really want source file(s) removed ... ? N
Copy from what existing file(s)? -#12-QD/sample.text
Copy to what new file? -LOWER-MYPROG.TEXT
```

6. Quit File-mgr. Turn the Lisa off (or you will reboot it in step 10 anyway) and put the floppy image's SD card in the PC.
7. Copy the image **from the SD card into your working folder**, replacing the copy there, so you are always injecting into the live image.

Every time you want to test a program

8. Write or edit your Pascal in any editor — Notepad is fine. Save it as, say, `myprog.txt` in the working folder. Normal Windows line endings are fine; the tool converts everything.
9. Inject and copy back:

```
python lisafile.py inject XFER.image MYPROG.TEXT myprog.txt
```

   then copy `XFER.image` back onto the SD card, replacing the old one. To check the card really has the new version, you can point the tool straight at it: `python lisafile.py list E:\XFER.image` (use your card's drive letter).

10. Put the card in the Floppy Emu and **reboot the Lisa**. This matters: the Lisa caches the floppy catalog, and without a reboot it will show you the old file.
11. Copy the file to the ProFile and compile from there (compiling directly off the floppy can run it out of room for the output):

```
F  C
Do you really want source file(s) removed ... ? N
Copy from what existing file(s)? -LOWER-MYPROG.TEXT
Copy to what new file? -#12-MYPROG.TEXT
```

(If `-#12-MYPROG.TEXT` already exists from last time, File-mgr will ask whether to delete the old one — answer Y.)

12. Press `P` for Pascal. Input file: `MYPROG`. Press Return through the List and Output prompts. A clean compile prints the code size and no errors.
13. Press `?` to reveal the extra menu, then `L` for Link. Enter `MYPROG`, then `IOSPASLIB`, then a blank line to end the list. Return through the Listing prompt; Output file: `MYPROGX`.
14. Press `R` for Run. File: `MYPROGX`. Your program runs.

If the compiler reports an error, fix the line in Notepad on the PC and repeat from step 8 — that is the whole loop.

Usage

```
lisafile.py info    disk.image
lisafile.py list    disk.image
lisafile.py extract disk.image NAME out.txt
lisafile.py inject  disk.image NAME in.txt [-o new.image]
lisafile.py raw2dc42 profile.raw  out.dc42
lisafile.py dc422raw profile.dc42 out.raw
```

`NAME` is the filename as it appears in the catalog, or `#N` for the numeric extent shown in File-mgr's List output. The numeric form always works, so use it if a name fails to resolve.

`inject` writes back to the same image unless you give `-o`.

Text file conventions

`inject` handles all of these for you; they are documented because they cost real debugging time to find, and because getting any of them wrong produces compiler errors that point somewhere else entirely.

A Workshop `.TEXT` file is paged. The first two blocks are the file's own 1024-byte header. After that, text lives in 1024-byte pages: a line never crosses a page boundary,

each page's content ends with a CR, and the rest of the page is NUL-filled. The Lisa's editor also compresses leading spaces into DLE codes (`0x10` then count+32); `extract` expands these so what you get on the PC is plain text.

Writing the text as one continuous stream — the obvious approach — produces files the editor displays perfectly and the compiler rejects. The two characteristic failures, both of which we hit before understanding the page layout:

```
*** Error 15 *** Illegal character in input.
```

on the line after your END (a long NUL run read mid-stream), and

```
*** Error 18 *** End of file encountered in a comment.
```

when a comment happens to straddle a page boundary — even though every brace is verifiably on disk.

Unused space is filled with comments. A donor file is usually much larger than your source. The remaining pages are filled with short, self-contained Pascal comment lines, each opening and closing inside its own page, so the compiler reads legal text all the way to the end of the allocation.

Line endings are bare CR (`0x0D`). The tool converts CRLF and LF.

One line may not exceed 1024 bytes. The tool refuses the inject and tells you which line, rather than writing a file that cannot be paged.

Format notes

Findings from reverse-engineering against known-good images. Recorded because they are not, as far as I can tell, written down anywhere else.

Tag layout differs between floppy and ProFile. Both put the file ID at bytes 4–5, matching the "Extent" column in File-mgr's List. The sequence number moves:

Tag size	Media	Sequence at
12 bytes	floppy	bytes 6–7
20 bytes	ProFile / Widget	bytes 12–13

The 20-byte tag carries six extra bytes before the sequence. Assuming 6–7 everywhere makes every ProFile file look one block long, because all sequences read as zero and

collapse into one dictionary entry.

Interleave. ProFile raw images are 5:1 interleaved. DC42 images — floppy or ProFile — are in logical order. This tool deinterleaves on load and re-interleaves on save, so all internal work is logical.

DC42 checksums. Two 32-bit checksums at offset 72: data first, then tags. The algorithm sums big-endian 16-bit words, rotating the accumulator right one bit after each addition. The tag checksum skips the first block's tag entirely — 12 bytes on a floppy, 20 on a ProFile. Getting this wrong yields an image that emulators and Floppy Emu accept happily and Disk Copy 4.2 rejects.

Catalog entries. Entries are 64 bytes apart, name NUL-terminated, file ID as a big-endian u16 at name+35. That shape occurs by coincidence throughout a 46 MB image, so the parser requires two independent confirmations: the ID must actually own blocks in the tag area, and its block must hold at least two candidates sharing the same offset-mod-64 alignment. With both checks it reads all 354 files off a Workshop 3.0 volume correctly.

Credits

Written with Claude (Anthropic). The format work was done by comparing against images written by the Lisa itself — the safest available oracle.

Thanks to AlexTheCat123, whose ESProFile, LisaFPGA, LOS Compilation Base image and Minesweeper source made all of this reachable, and to the LisaList2 community where the need for a tool like this was first raised.