Digital archaeology of the microcomputer, 1974-1994

(Or, how to prevent the Dark Ages of computing through free software)

Short URL: http://fsmsh.com/1986
permission for 8bit-wiki.de.vu friendly granted by Steven Goodwin

In a few years time, it will be impossible to study the history of home computers since everything at the time was proprietary; both in terms of the physical hardware, and all the software that ran upon it since most of it is encumbered by software “protection” to prevent copying.

To compound the problem, the hardware is dying (literally) and (being proprietary) can’t be rebuilt in any equivalent manner. In some cases the software is physically disintegrating too since, in the case of many 8-bit micros from the 1980’s, the storage medium was cassette tape; a temperamental mechanism at the time, let alone now. It’s not that no computer innovation took place in the 1980’s, just that none of it will be recorded.

What follows is a ten-point plan outlining the primary issues of digital archaeology, the methods necessary to preserve the legacy, and how free software can lead this endeavour.

1. Keep the ROM image

Amstrad were very good in making the (proprietary) Spectrum ROM available for all emulator users, and other companies should be encouraged to follow suit. Most developers of that era would write code that directly called machine code routines at specific places in ROM (or even jump halfway into an instruction) that, as a consequence, makes the development of a binary-compatible equivalent ROM very difficult. This is contrary to software developed under DOS and later operating systems that have indirect access through interrupts, or dynamically linked API calls, which allows an interface-compatible version to be developed, such as Free-DOS or Wine.

The ROM images must be made publicly available, along with the documents describing the file format in which they’re stored, and into which memory location(s) it should be loaded. An MD5 checksum is also useful to differentiate between the different versions of seemingly identical `machine.rom` files that exist.

2. Document the audio structure

This means both the baud rate of the cassette recording, and the method used of encoding each byte. Knowing the frequency of the intended recording, and how it sounds will help a lot. This is especially true if the only audio recording available is a copy made on a broken tape recorder that is going slightly too fast/slow and suffering from wow and flutter. Software can only fix this if the operator knows what to fix.

Complete sampled audio files must exist for each known platform. If the platform has a different save format for BASIC and machine code programs, for example, then two audio files should be available. Each sampled file should have annotations describing how the header and data block markers are stored, and what the resultant data should look like.

3. Use the highest resolutions

Obvious to most, but always store data in the highest resolution that exists. Don't try and save disc space by compressing the 400 KB WAV file of a program down to an MP3 file, since that introduces loss. And information loss costs future generations more than the few pence saved in disk space. Every ZX Spectrum game ever written will fit onto a single DVD, so no one has an excuse.

Additionally, back-up the data to a remote site should the worst happen. So if you must hoard, give others access to your hoard.

Oh, and if you use compression to store files, make sure the compressor/decompressor has a free software implementation of it, and that it works on each file.

4. Document each piece of software

A cassette tape bearing the name "Jet Set Willy" will mean little to future generations. Furthermore, any historian will not know of which myriad versions of Jet Set Willy this recording is: ZX Spectrum, Commodore 64, or even Dragon 32.

At a bare minimum, all software should describe the platform, memory requirements, language, and method by which it should be loaded, as some machines load programs differently if they were BASIC or Machine Code.

5. Use network-aware content management systems

Information is only of use if it can be read. Store all such information in accessible systems, connected with an open standard such as TCP/IP. In that way, even an old, proprietary system, can hold the information if necessary. In this way, it is easy to find out if a PlayStation 2-Linux kit, or ZX81, needs a special sync-on-green display in order to work.

Free software has provided many good solutions to this problem already, and I shall start no flame wars by recommended any specific one.

6. Circumvent copy protection

Imagine the faces of generation now+n when they've finally got their emulator working, loaded a snapshot of software X, and discovered a screen saying "Now enter the fourth word on page 27 of the manual". Most people don't have the manual to their VCR, let alone a computer game from 20 years ago.

Also, think of all the clever techniques used in copy "protection" through the 1980's; missing file headers, double speed loading, and so on. Perhaps there are 5 people currently in the world that know how any specific variation of copy protection worked: the one who wrote it, and the four who cracked it for pirate disks or to make it emulator-friendly. In a few years, all this knowledge will be lost.

Save all snapshots in a position after any copy protection methods, and/or indicate how the system works so it can be bypassed.

7. Free software emulators

In addition to helping port the emulator to the latest and greatest platforms, this stops bit rot from setting in when the only emulator available for platform X is for MS-DOS 5.0, and has not been updated since 1998. The adage of "the only complete software is obsolete software" does not hold

true when it comes to software archiving.

Having the source code also helps users when it comes to the methods of loading software into the emulator, as documentation can be scant.

If you have emulator source code, release it under a free license now before it's too late! (It's also easier than writing documentation!)

8. Free software tools

No emulator stands alone. There's always a selection of conversion and snapshot management tools to help get the data into the right format. If no one knows what this format is intended to be, the problem of deciphering it is no different to understanding the original format.

Information should also be provided on suggested settings for the conversion processes. To include "how" as well as "why", describing both the host and target machines. This becomes more time-critical for the present as MS-DOS can still be coaxed into working. Because the timings on DOS machines were often implemented with hacks, based on processor speed, problems can and will occur in this area. No matter how last century Free-DOS might appear, it serves a useful purpose in digital archaeology.

9. Latest is not greatest

As most historians will tell you, go back to the original documents. So, at any time you find a new version of some old software, remember to keep the original. Bugs may have been introduced, or features might have been removed, or any number of other issues.

Keep two credos in mind: "keep everything" and "if it ain't broke—don't fix it"

10. The source chain

When considering how to archive material for the future consider "the source chain". That is, you must have the source for every step of the archival process for it to be available to future generations. If the entire chain is not available in source form, store the material up to the most direct tool that includes source. The typical chain is:

- Cassette tape (physical)
- Sampled audio (WAV file)
- Virtual cassette file (VCF), for a specific emulator
- Emulator-specific format, as a program
- Emulator-specific snapshot, as a memory dump

So, if you have the source for the conversion tool that translates WAV into Virtual Cassette File, but not the source for the emulator that uses it, the Virtual Cassette File is the last future-safe format. If you do not have that, and the emulator (or its snapshot format) is closed, then the last safe format is that of sampled audio. Preferably in WAV, because there's lots of source available to describe it, and some modern editors do not support RAW, AU or SND format.

So there you have it. Simple, obvious, and common sense ideas to ensure there is a history of computing to look back upon. After all, if the machines of our past are truly held in such high esteem, shouldn't we expend the effort to preserve it?

Steve Goodwin

Public information

Biography

When builders go down to the pub they talk about football. Presumably therefore, when footballers go down to the pub they talk about builders! When Steven Goodwin goes down the pub he doesn't talk about football. Or builders. He talks about computers. Constantly...

He is also known as the angry man of open source.

Steven Goodwin a blog that no one reads that, and a beer podcast that no one listens to :)