A case study in software restoration: IBM 704/709/7090/7094

Paul McJones

May 5, 2006

Abstract

Software, to the extent that it is pure information, never decays. However the motivation for writing a program, the techniques used in its construction, its method of operation, and other context from the time of its creation and use is easily lost. Recovering that context involves a process not unlike that of restoring historic hardware. By studying some examples of individual and collective projects, we can extrapolate to how an institution such as the Computer History Museum can contribute qualitatively and quantitatively to this aspect of software history. We focus on recent work involving software for the IBM 704/709/7090/7094 series.

Introduction

In November 2003 I decided to try to track down the source code of the original IBM 704 Fortran compiler. Along with other volunteers, staff, and trustees of the Computer History Museum, I had joined the Software Collection Committee[1] in order to explore the software side of the Museum's mission: "To preserve and present for posterity the artifacts and stories of the information age." Fortran was arguably the first higher-level programming language, and its compiler was the first optimizing compiler, so this seemed like a good way to gain experience with the issues of collecting, preserving, and presenting historic software.

In the 30 months since then, I've made progress toward my original goal, but perhaps more importantly I discovered a community of people who are engaged in a variety of projects involving historic software from the same era. In this paper I refer to this extended activity as "software restoration" in analogy to the "hardware restoration" projects that are being carried out at the Computer History Museum and elsewhere. Software restoration can result in runnable software, the ability to rebuild the software from its source code, and a detailed appreciation for the obstacles that were overcome by the original developers of that software.

Although I initially started looking for a specific program, IBM 704 Fortran, this paper attempts to describe the larger "ecosystem" of software restoration for the IBM 704/709/7090/7094 family of computers. This line of computers spanned from the Von Neumann-class IBM 701 to the IBM System/360, and served as the workhorse of the scientific and engineering community during these Cold War years. Because each machine in the line was upward compatible from its predecessor, software could be reused. These factors led to the development of a number of operating systems,

[1] See http://www.computerhistory.org/events/special_projects/scc.shtml .

programming languages, and applications written by IBM, customers, and the SHARE user group[2].

The next sections of this paper describe the work of this community, grouped under these topics:

- Locating physical media: tapes, listings, etc.
- Transcribing to modern media
- Locating supporting materials: manuals, logic diagrams, etc.
- Locating processors, real or simulated
- Reacquiring expertise (reverse engineering)

I close with some observations on the nature of this community, and how institutions such as the Computer History Museum can facilitate this work.

Physical media: tapes, listings, etc.

It is important to acknowledge the many contributions of Paul Pierce in launching the field of preserving historic hardware and software from this era. A private collector, he has assembled an impressive collection of computer hardware, software, and documentation, and has made it available to the public. He has developed techniques for reading old 7-track magnetic tapes, and offers his services to others contributing to the public domain.

Pierce's library includes a treasure trove of material for the IBM 704/709/7090/7094 family, including tapes and/or card decks containing[3]:

- Machine diagnostics
- SHARE library, including SAP assembler
- IBSYS operating system, including many compilers
- IBM System/360 emulator (executable only)
- CTSS timesharing system from Massachusetts Institute of Technology

Another source of historic software is source listings. Paper turns out to be longer-lasting than many magnetic storage media. Some years ago, P.Z. Ingerman donated a listing of the IBM 704 Fortran II compiler to the Smithsonian Institution's National Museum of American History[4]. Ingerman received it from IBM in 1959 in response to his request to study it as part of his research into automatic coding problems at the University of Pennsylvania. In January 2006, a digital scan of the listing was acquired by the Computer History Museum.[5]

---

[2] See Atsushi Akera. Voluntarism and the Fruits of Collaboration : The IBM User Group, Share. *Technology and Culture*, Volume 42, Number 4, October 2001, pages 710-736.
[3] See http://www.piercefuller.com/oldibm-shadow/709x.html .
[4] NMAH catalog number 304,349.
[5] See http://archive.computerhistory.org/resources/software/IBM/X3435.2006/ for digital scans of the listing and a cover letter to Ingerman from A.L Harmon of IBM.

So far, listings represent the only known versions of the source code for the original Lisp 1.5 system. The oldest was donated to the MIT Library by Tim Hart, a member of the original Lisp 1.5 project[6]. The Computer History Museum now has a digital scan of this listing.[7] A later listing of the Lisp 1.5 system, as adapted to the CTSS operating system by Joel Moses and Robert R. Fenichel, has been located in Fenichel's basement, who entrusted it to me.

Transcription to modern media

Paul Pierce has developed techniques for reading 7-track tapes involving a 1970s Pertec 7-track tape drive, an analog-to-digital interface board, and custom software.[8] Using these techniques, and an older all-digital approach, he has read many tapes and put images of them online.

A more labor intensive but still workable approach is manual transcription. For example, Rich Cornwell has typed in machine diagnostics and parts of 704 Fortran II. Pascal Bourguignon has typed in Tim Hart's Lisp 1.5 listing[9], and Cornwell and others are close to having it error-free enough to run on Cornwell's simulator.

Another approach that shows promise is an OCR system tuned for old line printer typefaces by James Markevitch.

Yet another possibility is to take advantage of the human resources of Distributed Proofreaders[10], an organization set up to support digitization of public domain books in association with Project Gutenberg[11].

Manuals

Although it is a truism today that users never read the manual, documentation can play an important role in recreating the context necessary to understand and use the software. Al Kossow has assembled a magnificent collection of digitized manuals spanning from the early 1950s into the 1980s.[12]

Paul Pierce also has a significant online collection of manuals, logic diagrams, etc., typically related to physical machines in his collection.[13]

---

[6] MIT Library catalog number 1993.053.
[7] Until it has been given a formal lot number, it is available at http://community.computerhistory.org/scc/projects/LISP/LISP1.5-Bonnie-sBirthdayAssembly.pdf .
[8] See "7-track Magnetic Tape" on http://www.piercefuller.com/collect/proj.html .
[9] See http://groups.google.com/group/comp.lang.lisp/browse_frm/thread/67b1cabdf271870c .
[10] See http://www.pgdp.net/
[11] See http://www.pgdp.net/phpBB2/viewtopic.php?t=19898 (free registration required) for an initial discussion of the possibility of proofreading the Smithsonian 704 Fortran II compiler listing.
[12] See http://www.bitsavers.org/ . In particular, the subdirectories of pdf/ibm/ named 704/, 709/, 7090/, 7094/, and fortran/ are quite relevant to the subject of this paper.
[13] See http://www.piercefuller.com/library/index.html .

Based on my experience, there are still significant caches of historic manuals, technical reports, etc. in garages and basements around the world. The trick is to identify them and collect or copy them before they are thrown out or damaged by insects or moisture.

Processors, real or simulated

Historic source code would be worth collecting and studying even if it couldn't be executed. However, executing it adds greatly to understanding and appreciation. In some cases it's possible to run the software on the original hardware, but as the years go by this becomes increasingly more difficult. For example, Paul Pierce makes an effort to collect complete computer systems that in principle could be put into running condition, but in fact he has not attempted to operate his two IBM 709s or his IBM 7094 since he acquired them.

Luckily, it's quite feasible to simulate older, slower computers on newer, faster ones. Bob Supnik, through his Computer History Simulation Project, has been instrumental in writing simulators, obtaining freely available software, publishing papers, and helping others get involved.[14]

Consistent with their historic importance, the IBM 704/709/709x machines have inspired a number of simulators (mostly written in C):

- Paul Pierce wrote an early 709 simulator and related utilities for working with tape images that served as a starting point for several of the other efforts.[15]
- Rob Storey wrote an emulator (in Borland Delphi) with a detailed graphical simulation of the front panel and peripherals.[16]
- Dave Pitts extended Pierce's 709 simulator to better support the 709x, and also wrote a cross assembler and linker.[17]
- Bob Supnik added a 7094 simulator to SIMH version 3.6; his initial focus is supporting the CTSS operating system.
- Rich Cornwell is independently developing a SIMH-based simulator. He used the original diagnostics to debug his simulator, and is focusing on IBSYS and the older 704/709 software.
- Fritz Schneider is writing an emulator (in Java) with a detailed graphical simulation of the front panel and with a reference library linking to documentation at bitsavers.org.

Reacquiring expertise

Let's say you have software, a real or simulated computer to run it on, and documentation – what do you do next? Unfortunately it's not like popping a CD-ROM into a modern personal computer and following the prompts. Early computers such as the IBM 704 had

---

[14] See http://simh.trailing-edge.com/ .
[15] See http://www.piercefuller.com/oldibm-shadow/709x.html and http://www.piercefuller.com/oldibm-shadow/tool.html .
[16] See http://members.optushome.com.au/intaemul/Emul7094.htm .
[17] See http://www.cozx.com/~dpitts/ibm7090.html .

no operating system – each program, such as the Fortran compiler, ran "stand alone", and the computer operator needed to be trained how to operate it. But at least the original programmers were available to train the operators! Forty or fifty years later it is much more challenging to run this software. Here are a few examples of recent progress with respect to the IBM 704/709/7090 series:

- Rob Storey and James Fehlinger may have been first to boot IBSYS (from tape)
- Dave Pitts rebuilt IBSYS from source using his cross-assembler
- Rich Cornwell
    - may have been first to boot IBSYS from disk
    - rebuilt IBSYS from source, running on IBSYS
    - got Fortran II running
    - deciphered the SAP library tape format
    - is currently getting Lisp 1.5 to run
- Bob Supnik is reverse engineering CTSS hardware[18]

I asked Rich Cornwell for some "lessons" he's learned in this work. He gave me these examples of the importance of hardware diagnostics in addition to the manuals:

- A 704 with less than 32K words did not have the full 15 bits in its index registers
- 7094 double precision is not fully specified in the Principles of Operation
- Software depended on undocumented features of "dangling tape writes" (no tape mark)

Rich, who finds his intensive efforts very rewarding, notes: "I have found so many neat tricks and learned so much about programming computers from examining old code, where 32K of memory was considered a huge amount of memory (more than could be used :-)."

How are these projects organized?

My quest for the original Fortran compiler source code began with several months of following paths that seemed to lead to blind alleys. Gradually I became aware of people like Paul Pierce, Al Kossow, and Bob Supnik, newsgroups like alt.folklore.computers, and fascinating web sites of pioneers, historians, and hobbyists. Eventually word of my interest spread to people who contacted me with relevant information[19]. Now I feel like a member of a loosely-connected community of people interested in learning about, collecting, preserving, and operating historic hardware and software. The skills and interests of this community are wide-ranging and complementary. Working often in isolation, but happy to share their hard-won knowledge, this community is slowly reestablishing the technical context that existed almost half a century ago.

---

[18] See http://simh.trailing-edge.com/docs/ctss_hardware.pdf .

[19] The Dusty Decks blog I started to chronicle my progress has led to several important contacts. See http://www.mcjones.org/dustydecks/ .

How can institutions help?

Clearly the "classic" activities of museums – archival collections and public exhibits – are of great value for software history. In addition, institutions such as the Computer History Museum can contribute to the work being done by the informal community described in this paper by provide support for "software restoration" projects. This support could include:

- Holding workshops like the one at which this paper was presented
- Providing services such as media conversion
- Hosting online collaboration services (blogs, Wikis, etc.)
- Serving as a liaison to intellectual property owners
- Archiving the software and other artifacts
- Creating public exhibits

Coupling the initiative, enthusiasm, and domain expertise of the informal community with the long-term stability and professional expertise of institutions is a promising framework for progress in the field of software history.

About the author

Paul McJones learned to program the IBM 7094 in MAP assembler and Fortran IV as a high school senior in 1966, and landed his first programming job a few months later. He's been continuously employed since then in software research and development. Projects he's worked on include the CAL Time-Sharing System for the CDC 6400, the IBM System R relational database management system, the Pilot operating system for the Xerox Star office automation System, the operating system for the DEC Firefly multiprocessor workstation, implementations of the Snobol4, APL, and Modula-3 programming languages, and a variety of applications. He was named an ACM Fellow in 1994 and is currently a Principal Scientist at Adobe Systems Incorporated. He caught the software history bug documenting reunions of CAL-TSS and System R alumni and as an associate editor of the ACM SIGMOD Anthology. Since 2002 he's volunteered at the Computer History Museum on collections of documents and software for the IBM 7030 ("Stretch"), IBM 704 Fortran, and Lisp.