Notes on the history of fork-and-join

Linus Nyman and Mikael Laakso

Hanken School of Economics

As part of a PhD on code forking in open source software, Linus Nyman looked into the origins of how the practice came to be called forking.[1] This search led back to the early history of the fork system call. Having not previously seen such a history published, this anecdote looks back at the birth of the fork system call to share what was learned, as remembered by its pioneers.

The fork call allows a process (or running program) to create new processes. The original is deemed the parent process and the newly created one its child. On multiprocessor systems, these processes can run concurrently in parallel.[2] Since its birth 50 years ago, the fork has remained a central element of modern computing, both with regards to software development principles and, by extension, to hardware design, which increasingly accommodates parallelism in process execution.

The fork system call is imagined

The year was 1962. Melvin Conway, later to become known for "Conway's Law,"[3] was troubled by what seemed an unnecessary inefficiency in computing. As Conway recalls:[4]

> *I was in the US Air Force at the time – involved with computer procurement – and I noticed that no vendor was distinguishing between "processor" and "process." That is, when a proposal involved a multiprocessor design, each processor was committed to a single process.*

By late 1962, Conway had begun contemplating the idea of using a record to carry the status of a process. In the spring of 1963, "as an intellectual exercise," Conway wrote a paper suggesting a means of implementing parallel processing in a multiprocessor system design, using what he called "fork and join" system calls. The original illustration of the fork and join system call is shown in Figure 1. He presented the paper, "A Multiprocessor System Design," later that fall at the American Federation of Information Processing Societies (AFIPS) joint computer conference, and it was published in the conference proceedings.[5,6] Conway explains the basic idea in the paper (p. 146):

Fundamental to the concepts presented here is the principle, not yet commonly accepted, that parallel paths in a program need not bear fixed relationships to the processors of a multiprocessor system executing that program.

Though Conway had not come across anyone offering a solution to this inefficiency, he is nonetheless very modest about the novelty of his suggestion:[4]

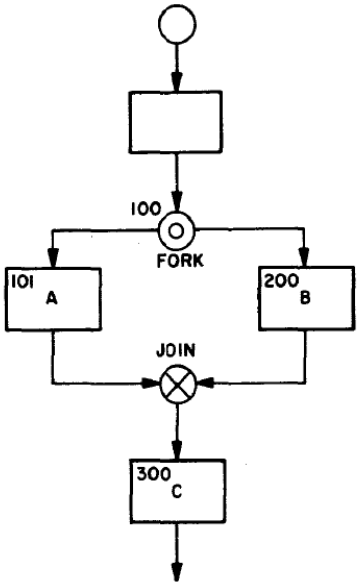

Figure 1. Conventions for drawing fork and join.

Figure 1 The fork system call as originally envisioned by Conway in 1963

Of course I do not claim to have invented that idea, since it must be present in all implementations in some form or other.

Indeed, there were some who had addressed the issue before him. In a footnote of the article, Conway notes that "the fork-join notion has been around for a while," albeit under different names: for example, Branch in the CL-II and Burroughs D825 AOSP computers; SIMU in the GAMMA 60 computer, and Branch Transfer or BRT in the conceptual machine discussed by Richards.[7] Reflecting on that footnote, Conway states:[4]

I only vaguely remember the origin of the footnote. The fact of its placement as a footnote suggests to me that it was added after submission of the paper at the urging of the editors, who may have noted that the technical concept was already in use. I conjecture that I added the footnote to assert the first use of the word in this technical context. If that is indeed the history of the footnote, the acceptance of my wording by the editors suggests their consent.

In fact, there was a second paper at that same 1963 Fall joint computer conference in which the terms "fork" and "join" were also used: "Generalized Multiprocessing and Multiprogramming Systems – Status Report", by A. J. Chritchlow.[6]

While the idea was not Conway's alone, the publication of the paper describing it seems to have codified the idea and become the impetus for its widespread implementation in

later computer systems. The system call's popularity was not due to Conway himself implementing the fork – he did not. Nor was it due to the paper having had a significant impact at the conference – Conway recalls that it did not. Rather, it appears due to the paper finding its way to Project Genie.

The fork system call is implemented at Project Genie

Project Genie began in 1963 at the University of California, Berkeley. It was funded by the Advanced Research Projects Agency of the Department of Defense and involved modifying a Scientific Data Systems (SDS) 930 minicomputer.[8] A primary goal of Project Genie was implementing timesharing, and the resulting computer system was commercialized as the SDS 940.[9,10]

The initial principal investigators were David Evans and Harry Huskey, but direction of the project was soon passed to Wayne Lichtenberger.[11] In their 1966 paper, "A User Machine in a Time-Sharing System," Butler Lampson, Wayne Lichtenberger, and Melvin Pirtle credit Conway's 1963 paper in discussing the multi-processing capabilities of the SDS 940.

Although the details of how Conway's ideas found their way into Project Genie have been lost in the passage of time, two possibilities seem the most likely. Wayne Lichtenberger recalls:[12]

> *In those days I was co-director of Project Genie, along with Melvin W. Pirtle. I was an Assistant Professor while Mel was a graduate student, working on his Ph.D. … As I had teaching duties and he did not I tended to act as liaison between the project and the Electrical Engineering department. Mel, on the other hand, was more free to interact with the students working for or with the project. These ranged from Butler Lampson, also a grad student, to Peter Deutsch, a sophomore when he came to us.*

Lichtenberger further notes that

> *Mel made many mentions of the Conway paper … in ordinary discussions over the next several years. That prompts me to suggest that he might have been the source of the idea within the project. Of course I don't know that for a fact.*

Peter Deutsch[13] brings up a second possibility, noting that he may well have come across Conway's idea through a pre-print of Dennis and Van Horne's 1966 paper "Programming Semantics for Multiprogrammed Computations,"[14] which both describes the idea and cites Conway's paper. Whether it was Pirtle or Deutsch who first

discovered Conway's paper may well remain unknown. However, based on recollections from the time, it was Deutsch who implemented the idea of the fork system call in the SDS 940. Even so, Deutsch recalled[13] that Project Genie's implementation was not identical to what Conway proposed.

Our attempts to track down the original Project Genie source code were unsuccessful. However, Dag Spicer of the Computer History Museum brought our attention to a project derived from Project Genie, the Tymshare Monitor, whose source code has been preserved online.[15] Butler Lampson was kind enough to have a look at the relevant section of the Tymshare code, noting[16] that *"[I]t looks to me as though most of it is untouched from the Berkeley system code."* Thus, the source code shown in Figure 2, as copied from Tymshare documentation dated July 1967, is likely to be a close approximation of, if not identical to, its implementation in Project Genie. Bitsavers has preserved manuals online for both the SDS 940[17] and Tymshare.[18]

```
*
*       FORK LOGIC
*
* FIND HIGHEST FORK IN STRUCTURE
HFK     ZRO
HFK1    LDA PPTR,2; SKA PRMSK; BRU *+2; BRR HFK
        MRG PLMSK; CAX; BRU HFK1
* GET FORK ENTRY. PUT PPB INTO PIM.
GFK     ZRO; LDA FPLST; SKG =0; BRR GFK
        SUB =PPTR; COPY AX,A
        XMA PPTR,2; STA FPLST; MIN GFK; BRR GFK
* PUT NEW FORK ON QIO
* SET PDOWN(OLD)=NEW, PDOWN(NEW)=0
*     PFORK(NEW)=OLD, PPAR(NEW)=PDOWN(OLD)
STFK    ZRO; STX FKO4; SKR NFORK; BRU *+2; BRM MONCR
        LDX PPB; LDB PB,2; STB PPB; CXB; STB STFK2
        LSH 3; LDX FKO4; STB PIM,2; CXA
        LDX =QIO; BRM QPUT; LDX PACPTR; LSH 12
        ETR PLMSK; XMA PPTR,2; CAB; ETR PRMSK; ADM PPTR,2
        LDA FKO4; XXA; ETR PRMSK; STA PPTR,2; LDA PACDMB; STA PTEST,2
        LDX PACPTR; CLA; RSH 3; LDA QUTAB; LSH 15; BRR STFK
STFK2   ZRO 0    XPB FOR NEW FORK
```

Figure 2 "Fork logic" as implemented in the Tymshare monitor, a program derived from Project Genie

The fork system call was only one of several innovative features of Project Genie; other notable aspects of the SDS 940 include timesharing, the line-oriented text editor QED, command-line completion, and state-restoring crash recovery.[11] Soon after Project

Genie other computer systems also implemented the fork or some fork-like functionality, among them the TENEX (later TOPS-20) operating system for the PDP-10,[19] the Berkeley Computer Corporation 500,[20] and the RC4000.[21] Furthermore, Project Genie's fork system call was the inspiration behind its implementation in Unix.[22] Since its creation, Unix has seen many significant derivatives, further increasing the amount of operating systems that make use of the fork.[23] In addition to these, Apple's iOS and OSX operating systems are based on the Berkley Software Distribution (BSD), which is itself a Unix derivative.[24] Furthermore, Linux also implemented the fork system call. Since its creation, Linux has seen hundreds of different distributions,[25] including the popular Android operating system.

The fork system call has become an integral part of the software driving everything from desktop machines and servers to mobile phones, tablets, and the ever-increasing array of computerized devices all around us. The significance of the fork has even extended beyond software: the basic principle of supporting parallelism in software has led first to multithreaded processors,[26] and then to multi-core processors[27] becoming the norm.

In hindsight, the envisioning and implementation of the fork system call seems an inevitability—a matter more of *who* would first accomplish it, than of *whether or not* it would ever come to be. However self-evident advances seem in retrospect, we tip our hats to the pioneers who labored without the luxury of retrospection. More than half a century after Melvin Conway's paper pondering an inefficiency in computing, the fork concept remains today an integral part of both computer software and hardware.[28]

Acknowledgements

The authors offer their heartfelt thanks to the pioneers and experts who took the time to answer our questions: Melvin Conway, Wayne Lichtenberger, L Peter Deutsch, Butler Lampson, Ken Thompson, Jack Dennis, Fernando Corbató, Dag Spicer, Brian Randell, Jerry Saltzer, and Tom Van Vleck. The Anecdotes editor directed us to some additional documents and people.

References and notes

---

[1] Nyman's dissertation "Understanding Code Forking in Open Source Software – an examination of code forking, its effect on open source software, and how it is viewed

and practiced by developers" can be accessed here: https://helda.helsinki.fi/handle/10138/153135

[2] For a somewhat deeper explanation, with sample code, see: https://en.wikipedia.org/wiki/Fork_(system_call)

[3] "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure." The thesis was originally part of a 1968 article by Melvin Conway published in Datamation, but it was popularized and given its name by Fred Brooks in The Mythical Man Month (1975).

[4] Melvin Conway, personal correspondence with Nyman, 2012.

[5] M. Conway, "A Multiprocessor System Design," AFIPS Proceedings of the Fall Joint Computer Conference, vol. 24, pp. 139–146, 1963.

[6] The front matter including the table of contents for the 1963 FJCC Proceedings is available at: **http://tinyurl.com/1963FJCC**

[7] P. Richards, "Parallel Programming," Technical Operations Incorporated Report No. TO-B 60-27, August 1960, p. 4.

[8] W. Lichtenberger and M. Pirtle, (1965) "A Facility for Experimentation in Man-Machine Interaction," Proceedings of the Fall Joint Computer Conference, part I, pp. 589–598, Nov. 30–Dec. 1, 1965. DOI: 10.1145/1463891.1463955

[9] See also: https://en.wikipedia.org/wiki/SDS_940

[10] The Annals has previously published an anecdote by Bo Lewendal covering, among other things, his time at Project Genie. See: B. Lewendal, "My Corner of the Time-Sharing Innovation World," IEEE Annals of the History of Computing, vol. 36, no. 4, pp. 97–101, 2014. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6982118 DOI: 10.1109/MAHC.2014.57

[11] See P. Spinrad and P. Maegher "Project Genie: Berkeley's piece of the computer revolution," Forefront, Fall 2007, pp. 22-25. Available at: http://engineering.berkeley.edu/sites/default/files/docs/2007Fall.pdf

[12] Wayne Lichtenberger, personal correspondence with Nyman, 2012.

[13] L Peter Deutsch, personal correspondence with Nyman, 2012.

[14] J. Dennis and E. Van Horn, "Programming Semantics for Multiprogrammed Computations," Communications of the ACM 9 (3), Mar. 1966, pp. 143–155. DOI: 10.1145/365230.365252

[15] The Tymshare Monitor source code survives in Bitsavers' archives, available at: http://bitsavers.informatik.uni-stuttgart.de/pdf/sds/tymshare/listing/Tymshare_Monitor_Jul67.pdf

[16] Butler Lampson, personal correspondence with Nyman, 2012

[17] The SDS 940 manual can be found at: http://bitsavers.informatik.uni-stuttgart.de/pdf/sds/9xx/940/901116A_940_TimesharingTechMan_Nov67.pdf

[18]Tymshare's manual notes that "The original Berkeley manual was written by Butler Lampson and this manual is a modification of that manual." The manual can be found at: https://ia801609.us.archive.org/7/items/bitsavers_tymshareSDemReferenceManualJul67_3525977/Time-Sharing_System_Reference_Manual_Jul67.pdf

[19] The developers of TENEX were users of a 940 system before and as they developed TENEX. For an Annals anecdote on TENEX and TOP-20, see: D. Murphy, "TENEX and TOPS-20," IEEE Annals of the History of Computing 15; 37 (1): pp. 75–82, Jan.-Mar. 2015. DOI: 10.1109/MAHC.2015.15

[20] For a 1969 BCC document describing the implemented MCALLs on the Model 1 Sub-Process System, see: https://ia601602.us.archive.org/16/items/bitsavers_bccorigina_2269885/BCC_M-07.pdf

[21] A 1969 RC 4000 Computer Reference Manual (2. Edition) can be found at: http://bitsavers.informatik.uni-stuttgart.de/pdf/regnecentralen/RC_4000_Reference_Manual_Jun69.pdf

[22] K. Ritchie and D. M. Thompson "The UNIX Time-Sharing System," Bell System Tech. J. (AT&T), 57 (6): pp. 1905–1929, 1978. Available at: http://www.cs.berkeley.edu/~brewer/cs262/unix.pdf See also K. Ritchie, "The Evolution of the Unix time-sharing System," Proceedings of the symposium on language design and programming methodology, Sydney, 10-11 Sept. 1979. DOI 10.1007/3-540-09745-7_2 Available at: http://link.springer.com/chapter/10.1007%2F3-540-09745-7_2

[23] For more on the history of Unix, including its many derivatives, see, e.g.: http://www.unix.org/what_is_unix/history_timeline.html

[24]Unix (as well as Linux) derivatives, or distributions, can be based on systems that are derivatives of derivatives. In the case of OSX and iOS, they are based on Darwin, which is a Unix derivative developed by Apple, that is itself based on NextStep, which in turn is a derivative of BSD, which is itself a derivative of Unix. Thus, while the proprietary nature of Apple's code doesn't allow us to look under the hood, it is reasonable to assume that the fork system call is a part of those operating systems as well. For more, see, e.g., https://opensource.apple.com/ and https://en.wikipedia.org/wiki/Darwin_(operating_system)

[25] For a list and timeline of Linux distribution, see: https://en.wikipedia.org/wiki/List_of_Linux_distributions

[26]See, e.g., T. Ungerer, B. Robic, and J. Silc, "Multithreaded processors," The Computer Journal, 45 (3): 320–348, 2002. DOI: 10.1093/comjnl/45.3.320

[27] See, e.g., A. Roy, J. Xu, and M. Chowdhury, "Multi-core processors: A new way forward and challenges," in International Conference on Microelectronics 2008, pp. 454– 457, Dec. 2008. DOI: 10.1109/ICM.2008.5393510

[28] We will welcome being told about other aspects of the history of fork-and-join that we could perhaps post on the web as an adjunct to this anecdote.