Notes on the History of Fork and Join

Linus Nyman and Mikael Laakso
Hanken School of Economics

Editor: David Walden

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 to the early history of the fork system call. Having not previously seen such a history published, here we look 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 regard to software development principles and, by extension, to hardware design, which increasingly accommodates parallelism in process execution.

Fork System Call 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 recalled,

> 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.[4]

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. Figure 1 shows the original illustration of the fork and join system call.

Conway presented a 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 explained the basic idea in the paper:

> 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.[7]

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

> Of course I do not claim to have invented that idea, since it must be present in all implementations in some form or other.[4]

Indeed, there were some who had addressed the issue before him. In a footnote in the article, Conway noted that "the fork-join notion has been around for a while," albeit under different names, such as 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 P. Richards.[8] Reflecting on that footnote, Conway stated:

> 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.[4]

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.[9]

Although the idea was not Conway's alone, the publication of his 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 recalled that it did not. Rather, it appears due to the paper finding its way to Project Genie.

Project Genie

Project Genie began in 1963 at the University of California, Berkeley. It was funded by the Department of Defense's ARPA and involved modifying a Scientific Data Systems (SDS) 930 minicomputer.[10] A primary goal of Project Genie was implementing time-sharing, and the resulting computer system was commercialized as the SDS 940.[11]

The initial principal investigators were David Evans and Harry Huskey, but direction of the project was soon passed to Wayne Lichtenberger.[12] In a 1966 paper, Butler Lampson, Wayne Lichtenberger, and Melvin Pirtle credited Conway's 1963 paper when discussing the multiprocessing capabilities of the SDS 940.[13]

Although the details of how Conway's ideas found their way into Project Genie have been lost in the passage

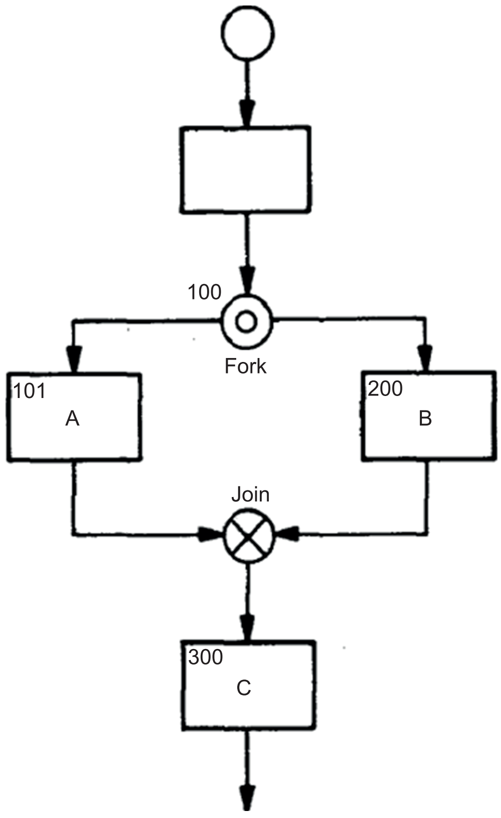


Figure 1. Fork system call. With this original illustration, Melvin Conway envisioned "fork and join" system calls in 1963.

of time, two possibilities seem the most likely. Wayne Lichtenberger recalled:

> 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.[14]

Lichtenberger further noted:

> 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 brought up a second possibility,[15] noting that he may have come across Conway's idea through a preprint of a 1966 paper by Jack Dennis and Earl Van Horne,[16] 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 that Project Genie's implementation was not identical to what Conway proposed.[15]

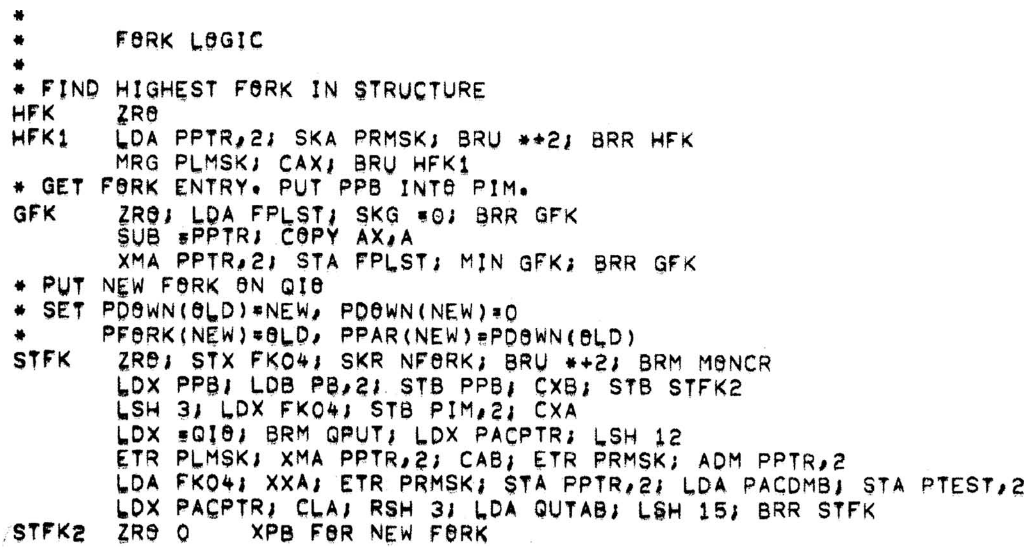
Our attempts to track down the original Project Genie source code were unsuccessful. However, Dag Spicer of the Computer History Museum brought our attention to a project derived from Project Genie, the Tymshare Monitor. (At the time we were conducting our research, the source code for the Tymshare Monitor was available online.[17]) Butler Lampson was kind enough to have a look at the relevant section of the Tymshare code, noting that it "looks to me as though most of it is untouched from the Berkeley system code."[18] 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[19] and Tymshare.[20]

The fork system call was only one of several innovative features of Project Genie; other notable aspects of the SDS 940 include time-sharing, the line-oriented text editor QED, command-line completion, and state-restoring crash recovery.[12] 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,[21] the Berkeley Computer Corporation 500,[22] and the RC4000.[23] Furthermore, Project Genie's fork system call was the inspiration behind its implementation in Unix.[24]

Since its creation, Unix has seen many significant derivatives, further increasing the number of operating systems that make use of the fork.[25] 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.[26] Furthermore, Linux also implemented the fork system call. Since its creation, Linux has seen hundreds of different distributions,[27] including the popular Android operating system.

Implementation Today

The fork system call has become an integral part of the software driving everything from desktop machines and servers to mobile



```
*
*       FØRK LØGIC
*
* FIND HIGHEST FØRK IN STRUCTURE
HFK     ZRØ
HFK1    LDA PPTR,2; SKA PRMSK; BRU *+2; BRR HFK
        MRG PLMSK; CAX; BRU HFK1
* GET FØRK ENTRY. PUT PPB INTØ PIM.
GFK     ZRØ; LDA FPLST; SKG =0; BRR GFK
        SUB =PPTR; CØPY AX,A
        XMA PPTR,2; STA FPLST; MIN GFK; BRR GFK
* PUT NEW FØRK ØN QIØ
* SET PDØWN(ØLD)=NEW, PDØWN(NEW)=0
*     PFØRK(NEW)=ØLD, PPAR(NEW)=PDØWN(ØLD)
STFK    ZRØ; STX FK04; SKR NFØRK; BRU *+2; BRM MØNCR
        LDX PPB; LDB PB,2; STB PPB; CXB; STB STFK2
        LSH 3; LDX FK04; STB PIM,2; CXA
        LDX =QIØ; BRM QPUT; LDX PACPTR; LSH 12
        ETR PLMSK; XMA PPTR,2; CAB; ETR PRMSK; ADM PPTR,2
        LDA FK04; XXA; ETR PRMSK; STA PPTR,2; LDA PACDMB; STA PTEST,2
        LDX PACPTR; CLA; RSH 3; LDA QUTAB; LSH 15; BRR STFK
STFK2   ZRØ 0    XPB FØR NEW FØRK
```

Figure 2. "Fork logic" as implemented in the Tymshare Monitor, a program derived from Project Genie. The Tymshare Monitor source code is likely a close approximation of the Project Genie implementation.

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 led first to multithreaded processors[28] and then to multicore processors[29] 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.[30]

Acknowledgments

We offer our 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. We also thank Anecdotes editor David Walden for directing us to additional documents and people.

References and Notes

1. L. Nyman, "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," PhD dissertation, Dept. of Management and Organization, Information Systems Science, Hanken School of Economics, Feb. 2015; 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 F. Brooks, *The Mythical Man Month*, Addison-Wesley, 1975.
4. M. Conway to L. Nyman, personal comm., 2012.
5. M. Conway, "A Multiprocessor System Design," *Proc. AFIPS Fall Joint Computer Conf.*, vol. 24, 1963, pp. 139–146.
6. The front matter, including the table of contents, for the 1963 FJCC Proceedings is available at http://tinyurl.com/1963FJCC.
7. Conway, "A Multiprocessor System Design," p. 146
8. P. Richards, "Parallel Programming," tech. report TO-B 60-27, Technical Operations Incorporated, Aug. 1960, p. 4.
9. A.J. Chritchlow, "Generalized Multiprocessing and Multiprogramming Systems," *Proc. AFIPS Fall Joint Computer Conf.*, vol. 24, 1963, pp. 127–136.
10. W. Lichtenberger and M. Pirtle, "A Facility for Experimentation in Man-Machine Interaction," *Proc. AFIPS Fall Joint Computer Conf.*, part I, 1965, pp. 589–598; doi:10.1145/1463891.1463955.

11. The *Annals* previously published an anecdote by Bo Lewendal covering, among other things, his time with Project Genie. See B. Lewendal, "My Corner of the Time-Sharing Innovation World," *IEEE Annals of the History of Computing*, vol. 36, no. 4, 2014, pp. 97–101; http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6982118.
12. See P. Spinrad and P. Maegher "Project Genie: Berkeley's Piece of the Computer Revolution," *Forefront*, Fall 2007, pp. 22–25; http://engineering.berkeley.edu/sites/default/files/docs/2007Fall.pdf.
13. B. Lampson, W. Lichtenberger, and M. Pirtle, "A User Machine in a Time-Sharing System," *Proc. IEEE*, vol. 54, no. 12, 1966, pp. 1766–1774.
14. W. Lichtenberger to L. Nyman, personal comm., 2012.
15. L.P. Deutsch to L. Nyman, personal comm., 2012.
16. J. Dennis and E. Van Horn, "Programming Semantics for Multiprogrammed Computations," *Comm. ACM*, vol. 9, no. 3, 1966, pp. 143–155; doi:10.1145/365230.365252.
17. During our research, the source code was available in the Bitsavers archives. Unfortunately, by the time this article was being prepared, the file in question had been removed. A selection of other materials related to the Tymshare project is still available at http://bitsavers.informatik.uni-stuttgart.de/pdf/tymshare/SDS_940/. We hope the source code file (originally archived as "sds/9xx/940/tymshare/Tymshare_Monitor_Jul67.pdf") will once again be made available in the future.
18. B. Lampson to L. Nyman, personal comm., 2012.
19. "SDS 940 Time-Sharing System Technical Manual," SDS, Nov. 1967; http://bitsavers.informatik.uni-stuttgart.de/pdf/sds/9xx/940/901116A_940_TimesharingTechMan_Nov67.pdf.
20. Tymshare's manual notes that "The original Berkeley manual was written by Butler Lampson and this manual is a modification of that manual." A. Hardy, D. Gardner, and V. Van Vlear, "Time-Sharing System Reference Manual," Tymshare, 21 July 1967; https://ia801609.us.archive.org/7/items/bitsavers_tymshareSDemReferenceManualJul67_3525977/Time-Sharing_System_Reference_Manual_Jul67.pdf.
21. TENEX developers 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*, vol. 37, no. 1, 2015, pp. 75–82; doi:10.1109/MAHC.2015.15.
22. For a 1969 BCC document describing the implementation, see J. Freeman, "MCALLs on the Model I Sub-process System," BCC, 1969; https://ia601602.us.archive.org/16/items/bitsavers_bccorigina_2269885/BCC_M-07.pdf.
23. P.B. Hansen, ed., "RC 4000 Computer Reference Manual," 2nd ed., A/S Regnecentralen, June 1969; http://bitsavers.informatik.uni-stuttgart.de/pdf/regnecentralen/RC_4000_Reference_Manual_Jun69.pdf.
24. D.M. Ritchie and K. Thompson, "The Unix Time-Sharing System," *Comm. ACM*, vol. 17, no. 7, 1974, pp. 365–375; www.cs.berkeley.edu/~brewer/cs262/unix.pdf. See also D.M. Ritchie, "The Evolution of the Unix Time-Sharing System," *Language Design and Programming Methodology*, Springer, LNCS 79, 1980; http://link.springer.com/chapter/10.1007%2F3-540-09745-7_2.
25. For more on the history of Unix, including its many derivatives, see www.unix.org/what_is_unix/history_timeline.html.
26. 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 https://opensource.apple.com and https://en.wikipedia.org/wiki/Darwin_(operating_system).
27. For a list and timeline of Linux distribution, see https://en.wikipedia.org/wiki/List_of_Linux_distributions.
28. For example, see T. Ungerer, B. Robic, and J. Silc, "Multithreaded processors," *The Computer J.*, vol. 45, no. 3, 2002, pp. 320–348; doi:10.1093/comjnl/45.3.320.
29. For example, see A. Roy, J. Xu, and M. Chowdhury, "Multi-core Processors: A New Way Forward and Challenges," *Proc. Int'l Conf. Microelectronics*, 2008, pp. 454–457; doi:10.1109/ICM.2008.5393510.
30. We welcome information about other aspects of the history of fork and join that we could perhaps post on the Web as an adjunct to this article.

Linus Nyman *is a researcher at the Hanken School of Economics. Contact him at linus.nyman@hanken.fi.*

Mikael Laakso *is an assistant professor at the Hanken School of Economics. Contact him at mikael.laakso@hanken.fi.*