Iterations
An Interdisciplinary Journal of Software History

211 Andersen Library, 222 – 21st Avenue South, Minneapolis, MN 55455 USA

Multicians.org and the History of Operating Systems

Thomas Haigh
Colby College

Date published: 13 September 2002

The Multics (Multiplexed Information and Computing Service) operating system occupies an important place in the history of software, yet has received little attention from historians. In most accounts it is a mere footnote—the inspiration for the far better known UNIX operating system. Nothing has been written on its actual usage. However, Multics veterans are determined to change this. Under the leadership of software engineer Tom Van Vleck, who spent almost sixteen years working on Multics at both MIT and Honeywell, they have created the Multicians.org Web site. According to the site, a Multician is "anybody who loves the system (with all its faults), who contributed to its development and success, who advocated it to others and tried to make it better."

Historical significance

Multics began its life at MIT in 1963, as a key element of Project MAC (standing, among other things, for Machine Aided Cognition). This bold name was characteristic of its initial sponsor, J. C. R. Licklider, a Harvard psychologist then serving as director of DARPA's Information Processing Technology Office (IPTO). Pursuing a personal vision of "man-machine symbiosis," he dispensed funds liberally and with little formal review to those of his acquaintances he believed able to achieve fundamental advances. Everything about Project MAC took place on a grand scale. Its initial DARPA grant of $2 million a year had risen to $4.3 million by 1969. At its peak it employed 400 research staff, and, as its activities broadened, it was eventually christened MIT's Laboratory for Computer Science.

In 1963, the concept of an operating system was quite novel. While the term itself was sometimes used, it did not (especially when applied to commercial products) always mean what it does today: a complex system mediating all interactions between users, application programs, and computer hardware. Instead, the term was applied to a collection of compatible utility programs—perhaps including

assemblers, compilers, debugging tools, standard routines for input and output, buffers to “spool” printer and tape output, and utilities designed to load a sequence (or “batch”) of programs into memory and automate some of the reconfiguration performed by human operators. Utilities of this kind played an increasingly important part in the bundles of hardware, software and services supplied by computer manufacturers.

Multics was something far more ambitious. By the early 1960s, the concept of “timesharing” was receiving considerable attention among computer researchers. Timesharing was intended to support interactive, general-purpose use of a computer—via a teletype unit or video screen. Commercial computers of the early 1960s ran only one program at a time—users submitted programs and received results hours or days later once their place in the queue came up. This ruled out many kinds of application, such as interactive management information systems, and made program-debugging and testing a very laborious process. Yet hardware was far too expensive to provide users with exclusive use of their own computers, and pundits of the era believed that extremely powerful computers would always be much more cost effective than small ones. Although “real-time” systems were sometimes constructed, such as the airline reservation system SABRE, these involved specially produced hardware, a specially built operating system, and exclusive use of some expensive computers. Many believed that huge centrally located computers would soon provide most of America’s computer power, just as large central generating installations produced most of its electrical power. The projected facilities were called “computer utilities.”

Timesharing had already been proven in university and laboratory tests—not least through MIT’s own Compatible Time-Sharing System (CTSS). Fernando J. Corbató, creator of CTSS, led the Multics effort. Multics was designed to provide general purpose, timesharing computing facilities to hundreds or thousands of simultaneous users. Its designers set ambitious targets. It had to be able to support all the programming capabilities and ways of working that its many and varied users demanded. It must be highly expandable and flexible. It needed to protect confidential information, while providing mechanisms for users and programs to exchange data. The system software itself had to be supremely reliable, and coordinate the sharing of memory, disk, and processor resources between hundreds of users while protecting itself from tampering or corruption. These requirements led to some then-novel design features now found: memory was paged and segmented, the operating system itself was written in a high level language, a hierarchical filing system provided a standard storage interface between applications, tasks were distributed dynamically between several processors, customers could be billed precisely according to the amount of memory, processor time and disk storage used, and the operating system was modularized so that component parts could be loaded independently into memory and shared simultaneously between several users.

This was to be no mere academic exercise. By 1964, General Electric had been awarded a contract to supply a new computer designed to Multics specifications —the GE 645. GE had targeted timesharing as an area where IBM was weak, hoping to establish itself as both a leading provider of timesharing services and a supplier of suitable hardware. The software development effort was conducted jointly by MIT, GE, and Bell Labs. A great deal was riding on the success of Multics, including the economic viability of the computer utility concept itself. The development exercise was originally expected to take two and a half years.

By 1968, Multics was still confined to the lab and was struggling for survival—a struggle it never definitively won, but took a very long time to lose. Like other big system software projects of the era, it proved bigger, slower, less powerful, more expensive and much later than expected. In early 1969, six months before a still-flaky Multics entered commercial use at MIT itself, Bell Labs pulled out. Following the abrupt termination of their work on Multics, two Bell Labs researchers, Ken Thompson and Dennis Ritchie, used an obsolete and underpowered minicomputer to produce a tiny timesharing operating system. They called it Unics, meaning “emasculated Multics”—a fitting name given the many Multics capabilities left behind. Together with the C programming language, developed as part of the same project, the slightly renamed UNIX became the leading research operating system of 1980s.

In 1970, shortly after Bell Labs dropped out of the project, General Electric abandoned its computer hardware business without ever having used Multics. Honeywell acquired GE’s computer division, and with it the Multics effort. Honeywell boasted several other operating systems, each supported by its own internal groups. Although around 80 Multics sites were installed between its debut as a standard commercial product in 1973 and its cancellation in 1985, the operating system never attained critical mass in the marketplace, or even won clear internal acceptance as Honeywell’s own next-generation product. MIT continued to collaborate in its development until 1977, after which point the burden passed exclusively to Honeywell, which in turn divested its computer operations to the French firm Bull in 1987.

The last Multics site was closed down in 2000. The initial target market of commercial “computer utilities” never fully developed, due in part to unexpected difficulty in producing efficient operating systems on this scale and in part to the equally unanticipated emergence of powerful minicomputers as a cost-effective alternative. Its two largest groups of users were universities, for whom the ability to support diverse types of users simultaneously was an attraction, and governmental research centers in the USA, Canada, and France. Corporate users included General Motors, Ford, and McDonnell Douglas. Multics hosted the world’s first commercially available relational database system—MRDS, released in 1976. While many Multics installations enjoyed long and productive lives, its primary achievement was as a research project rather than a working operating system. No emulator has been produced for any of the machines it ran on, and the

source code is still guarded by Bull. Multics is therefore—as of this writing—lost to the world.

Web site content

The Multicians Web site is a labor of love for its authors, and contains a wealth of material. The site functions primarily as the hub of an otherwise destroyed community of developers, managers, application programmers and users involved with the system over its three and a half decade career. Its creators believe that "as long as we have the Multicians, we have the most important part of Multics." Its directory lists almost 1,500 individuals, including more than five hundred email addresses and almost 120 home pages (many of which, it must be said, no longer appear to exist—resulting in broken links). A few dozen of these people have contributed actively to the development of the Multicians site and its precursors.

Its main contents are lists of (and links to) original technical papers and development documents, a glossary of Multics terms, personal home pages and individual reminiscences. The reminiscences include accounts of Multics operations at particular user sites, and the memories of some of the original system designers. Much of the site's content was originally assembled from the FAQs of the long-established Usenet newsgroup alt.os.multics, and from other documents apparently salvaged from old Multics systems themselves. This makes for something of a patchwork impression. Some five year old articles still include placeholders reminding the author to check certain information or fill in a particular section. Some installations and aspects of the Multics development story receive elaborate treatment, while others are ignored completely.

Indeed, the site lacks a simple overall description of Multics and its significance. Many documents assume a detailed knowledge of Multics hardware and terminology, or of operating system concepts. For example, the use of Multics as a commercial product and the relationship of Multics to the "computer utility" concept are both referenced repeatedly but neither receives a clear overall statement. Some of the links to external sites covering other operating systems of interest are broken, or point to sites with little relevant content. Many of these links are useful, though neither here nor elsewhere are any published works by historians of computing referenced. This reflects the general focus of the site on technical aspects of the story, and toward Multics participants and younger software engineers.

Its editors have made some efforts to bring structure to the site. It can be browsed via a chronology of events, an alphabetical list of Multics installations (supplemented by a chart showing the years during which each was in operation), a somewhat sketchy attempt at an overall history, and a list of online documents. There are many links between sections, but the overall structure can be confusing. This is exacerbated by the fact that some useful if non-standard JavaScript pull-

down navigation menus exist only on the main page, and are replaced with a different set of simple hyperlinks elsewhere in the site.

Audience and purpose

The Multicians site is much closer to being a primary source than a secondary one. It represents an intriguing mixture of archive, public relations on behalf of a defunct product, and private conversation between long-time colleagues. Its intended relationship to a wider audience is not entirely clear. The site itself lists its purposes as

- preserve the technical ideas and advances of the system so others don't need to reinvent them.
- record the history of the system, its builders, and its users before we all forget.
- give credit where it's due for important innovations.
- remember some good times and good people.

Notable in their absence from this list are needs of professional historians and the general public. The last two goals—credit and remembrance—imply an audience within the Multics community itself. It can be assumed that they have been taken care of. The first, preservation of good ideas, is an important one. Multicians complain that later and more commercially successful systems included implementations of features such as memory segmentation, dynamic linking, and security that were not merely guilty of failing to improve on their own work, but were actually designed in ignorance of their labors. However, few entirely new operating systems are now being written and it is too late to go back and fix the existing ones. This particular horse has already bolted.

Despite this, the technical coverage is likely to prove an eye-opener to experienced UNIX programmers interested in its intellectual origins, and in seeing how different design philosophies can shape an operating system. Thanks to its lavish funding and the involvement of both MIT and Bell Labs, the project included as much innovative computer science as any operating system ever produced. (Having said this, a fuller technical history would include a better comparison of Multics with the earlier work on paging and virtual machines included in the British Atlas system). This audience is likely to dive right in and enjoy the descriptions of virtual memory architecture and programming language support. Spirited discussions on Multics can be found at the Slashdot.org ("News for nerds") discussion forums—see http://slashdot.org/articles/00/11/13/066228.shtml and http://slashdot.org/articles/00/03/07/0948211.shtml.

Indeed, the story of Multics is likely to be of considerable interest to anyone with a stake in today's open source software movement, and its flagship GNU/Linux system. Like UNIX, Linux started out as a system so stripped down as to be devoid of useful functions, and grew threw a process of messy accretions and

iterations, shaped by the interests and skills of hundreds of individual programmers and loosely coordinated teams. The most important feature of the original system was its openness and expandability. Through this process it has acquired many, though by no means all, of the capabilities of Multics—and is used on millions of computers.

Multics, in contrast, was a painstakingly planned and coordinated system of enormous ambition. Its proponents point to the advantages of this approach. It was, they say, far more secure than UNIX, because security was built in to its very memory allocation model. Because the hardware was customized for the operating system, it included elaborate support for secure memory management. Likewise, clever design made it possible to remove or reconfigure processors and memory while it was running. A trivial but revealing contrast is that Multics used a standard set of command switches between different utility programs, whereas the options supplied to UNIX commands depend on the whims of their many authors, a confusing and time-wasting situation. Multics was coherent and rational, where UNIX is chaotic and whimsical. On the other hand, UNIX has thrived and proliferated, whereas Multics is dead.

While most of the Multicians themselves appear to blame this on half-hearted support by Honeywell and the slow development of suitable hardware, the more interesting question may be whether such an approach could ever have succeeded—even with the backing of IBM itself. Was Multics a system that drowned in the ambition and talent of its own creators? Indeed, Multics stands as an interesting contrast to the better-known travails of Frederick P. Brooks and the IBM OS/360 team, whose experiences were chronicled in the classic *Mythical Man-Month* (Addison-Wesley, 1995) and thereby emerged as a ubiquitous fable in the new field of software engineering. Work on Multics took place over a much longer period, and its architectural goals were far more ambitious—though it was at least spared the OS/360 requirement of running on an enormous range of different machines. If one is interested in what Eric Raymond has dubbed the "cathedral" approach to operating systems construction, one could not hope to find a better illustration of its strengths and weaknesses.

Contribution to historical research

The second goal, to "record the history of the system, its builders, and its users," is the hardest to achieve—because of its breadth, and because it implies the eventual transmission of this information beyond the Multicians themselves. Anyone coming to the site with no more than a general interest in the history of software is unlikely to get a clear sense of the historical place of Multics—there are too many trees on view and not enough wood. The necessary background is hard to acquire—in part because there is no adequate published history of Multics. The MIT, Project MAC side of the time-sharing story has received some attention as a failed but instructive experiment. It is given a clear and elegant exploration in the first chapter of Simson Garfinkel's short book *Architects of the*

Information Society (MIT Press, 1999), a history of Project MAC and the origins of the Laboratory for Computer Science. While this lacks detail, a non-specialist interested in Multics would be well advised to start here. DARPA's contribution to computer science, of which Project MAC was a key element, receives its most thorough overall examination in Norberg and O'Neill's *Transforming Computer Technology* (Johns Hopkins Press, 1996). The importance of Multics has also been recognized in Martin Campbell-Kelly and William Aspray's *Computer: A History of the Information Machine* (Basic Books, 1996) again in the context of the computer utility and timesharing. Paul Cerruzi's *A History of Modern Computing* (MIT Press, 1998), Kenneth Flamm's *Creating the Computer* (Brookings Institution, 1998), Peter Salus' *A Quarter Century of UNIX* (Addison-Wesley, 1994), and Steve Lohr's journalistic *Go To* (Basic Books, 2001) all include Multics primarily as a foil to the later UNIX.

The site is clearly the richest source of information in existence on any aspect of Multics. How might the site help a professional historian or dedicated amateur go beyond existing accounts? The technical documents preserved here allow for quasi-archival research, and might also prove useful as primary documents for instructional uses in classes on research methodology—how to read social detail from technical documents. However, the site's most useful contribution is probably as a repository of information on the use of software. Very little has been written by historians about the history of operating systems, about failed projects, or about the way in which any kind of software was used—yet all these topics must be explored if the history of computing is to establish its relevance to broader historical fields, or resonate with the experiences of veteran computer staff themselves.

As the above summary of existing references made clear, anyone with an interest in the history of software is likely to have come across a mention of Multics. Few of these accounts, however, go much beyond the judgment made by Per Brinch Hansen in *Classic Operating Systems* (Springer Verlag, 2001) of a system "never widely used outside MIT… an overambitious dead end in the history of operating systems" (p. 14). John Couleur, in "The Core of the Black Canyon Computer Corporation," *IEEE Annals of the History of Computing*, vol. 17, no. 4 (1995), even implies that Multics was never ported beyond its initial implementation on the GE 645 designed for Project MAC because "the cost of upgrading Multics to a more modern architecture was prohibitive"—a claim apparently contradicted by the many reports given by the Multicians of installations based on the later Honeywell 6180 (announced in 1973).

The fact that Multics actually surfaced in the 1970s as a credible commercial product is seldom acknowledged, and it is here that the information gathered by the Multicians is most stimulating. There are many reasons for general historical neglect of Multics. It ran on large, expensive, and rare computers. It was not produced by IBM, and so escaped the attention of most potential customers at the time and thus far, given their not unreasonable bias toward the firm which

controlled more than 80 percent of the market, most historians. It was not the direct ancestor of any current operating system, and no currently operational hardware—real or simulated—can run it. [Having said this, the commercial story of Multics is given a brief examination—from the viewpoint of its producer—in JAN Lee's "The Rise and Fall of the General Electric Corporation Computer Department," *IEEE Annals of the History of Computing*, vol. 17 no. 4 (1995)].

Multics—a system few historians acknowledge was ever used commercially—is now the only vintage operating system for which data on commercial usage and first-hand accounts are readily accessible. It is hoped that users of other, less clannish, operating systems may follow the lead of the Multicians. The anecdotes and site histories given here tantalize us with a glimpse into the local communities that sprang up around each Multics installation. Many discuss the applications produced at each site, and give mythic moments from their local folklore—epic disk crashes, hair-trigger sprinkler systems, heroic programming feats. Their most striking feature is the extent to which the Multicians bonded with each other across the many boundaries separating them. They were employed by MIT, Bell, GE/Honeywell/Bull and a number of corporations and university research centers. They worked as researchers, systems programmers, applications programmers, marketers and managers. Many retained a commitment to Multics as they moved between different firms and roles—though the relationship between Honeywell and its Multics customers appears to have been strained at times. The most dedicated devoted a major chunk of their working lives to this system, and continue to defend its honor today. How could such a troubled, indeed doomed, project earn so much loyalty? You will not find an explicit answer here, but you do get a sense of the personal commitments behind the project—akin to the flavor of Kidder's classic *Soul of a New Machine* (Little Brown, 1981).

Multics deserves, and in time may well receive, a more prominent place in popular histories of computer technology. Most people read books on the history of computing because they want to understand where the machine on their desktop came from. As the complexity and sophistication of personal computers has increased, the relevant history has expanded. During the 1980s, they did not need to look far beyond the history of the microprocessor and the early PC hobbyists such as Steve Jobs—a story best captured in by Paul Freiberger and Michael Swaine in *Fire in the Valley* (Osborne/McGraw-Hill, 1984). In the early 1990s the graphical user interface became a staple of personal computing, and in the late 1990s the Internet entered everyday life. As a result, the public developed a mild interest in histories of the Mac, Xerox PARC, the ARPANET, and the Web browser.

Less visible to most people has been the transformation of the underlying portions of desktop operating systems. The three main desktop systems now shipping are Windows XP, Mac OS X and Linux. All three were shaped by Multics—OS X and Linux are both derived from UNIX, and therefore from the processes of emasculation and implantation through which features were stripped from Multics

and then reintroduced piecemeal. Windows XP, today's mainstream desktop operating system, has a separate line of descent from Multics. It is really Windows NT with a consumer-friendly facade, and NT had its origins in an aborted project to build an operating system for a new generation of Digital Equipment Corporation (DEC) minicomputers (a fact attested to by a sizable cash and services settlement on the part of Microsoft). The origins of NT are explored in Mark Russinovich, "Windows NT and VMS: The Rest of the Story," *Windows NT Magazine* 4, no. 12 (1998): 114-19. DEC, like the other minicomputer vendors of the 1980s, learned a lot from Multics about timesharing, memory protection and multiprocessing. The architecture of a modern PC owes far more to Multics, and other timesharing mainframe (and later minicomputer) systems than to the self-taught hackers of the 1970s. This is well known to specialists. Indeed it is hard to teach operating system principles without taking an historical approach to their evolutionary development—witness the dinosaurs on the covers of all six editions of the standard Abraham Silberschatz and Peter Baer Galvin textbook *Operating System Concepts* (latest edition is John Wiley, 2001). The challenge, which may well be insurmountable, is to convey enough of the principles of operating system architecture to make the story comprehensible to an interested lay reader while giving enough of the sense of the broader issues shaping the usage and development of operating systems to place the story in its human and social context. It is the sense of immediacy given by the Multicians site that historians are liable to find most valuable when they attempt this task.

Thomas Haigh, "Multicians.org and the History of Operating Systems," *Iterations: An Interdisciplinary Journal of Software History* 1 (September 13, 2002): 1-9.