WIKIPEDIA

Multics

Multics ("**Multiplexed Information and Computing Service**") is an influential early time-sharing operating system based on the concept of a single-level memory.[4][5] Multics "has influenced all modern operating systems since, from microcomputers to mainframes."[6]

Initial planning and development for Multics started in 1964, in Cambridge, Massachusetts. Originally it was a cooperative project led by MIT (Project MAC with Fernando Corbató) along with General Electric and Bell Labs. It was developed on the GE 645 computer, which was specially designed for it; the first one was delivered to MIT in January, 1967.

Multics was conceived as a commercial product for General Electric, and became one for Honeywell, albeit not very successfully. Due to its many novel and valuable ideas, Multics has had a significant impact on computer science despite its faults.[7]

Multics has numerous features intended to ensure high availability so that it would support a computing utility similar to the telephone and electricity utilities. Modular hardware structure and software architecture are used to achieve this. The system can grow in size by simply adding more of the appropriate resource, be it computing power, main memory, or disk storage. Separate access control lists on every file provide flexible information sharing, but complete privacy when needed. Multics has a number of standard mechanisms to allow engineers to analyze the performance of the system, as well as a number of adaptive performance optimization mechanisms.

Multics

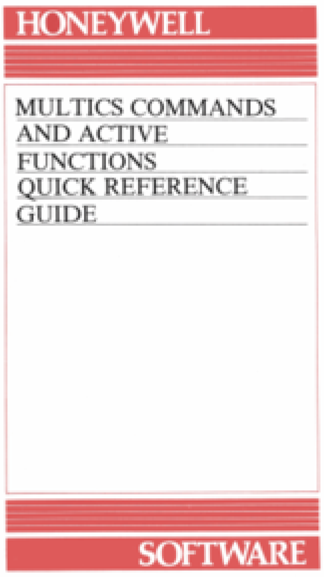
Developer	MIT, GE, Bell Labs
Written in	PL/I, Assembly language[1]
Working state	Mature, historic, emulator available
Source model	Free software
Initial release	1969
Latest release	12.6f / December 2016
Repository	web.mit.edu/multics-history/source/Multics_Internet_Server/Multics_sources.html (http://web.mit.edu/multics-history/source/Multics_Internet_Server/Multics_sources.html)
Available in	English
Platforms	GE-645

Contents

	mainframes, Honeywell 6180 series machines
Kernel type	Monolithic kernel
Default user interface	Command-line interface
License	Originally proprietary, Free software Multics License since 2007[2][3]
Official website	www.multicians.org (http://www.multicians.org)

Novel ideas

Multics implements a single-level store for data access, discarding the clear distinction between files (called *segments* in Multics) and *process memory*. The memory of a process consists solely of segments that were mapped into its address space. To read or write to them, the process simply uses normal central processing unit (CPU) instructions, and the operating system takes care of making sure that all the modifications were saved to disk. In POSIX terminology, it is as if every file were `mmap()`ed; however, in Multics there is no concept of *process memory*, separate from the memory used to hold mapped-in files, as Unix has. *All* memory in the system is part of *some* segment, which appears in the file system; this includes the temporary scratch memory of the process, its kernel stack, etc.



Multics Commands reference manual

One disadvantage of this was that the size of segments was limited to 256 kilowords, just over 1 MB. This was due to the particular hardware architecture of the machines on which Multics ran, having a 36-bit word size and index registers (used to address within segments) of half that size (18 bits). Extra code had to be used to work on files larger than this, called multisegment files. In the days when one megabyte of memory was prohibitively expensive, and before large databases and later huge bitmap graphics, this limit was rarely encountered.

Another major new idea of Multics was dynamic linking, in which a running process could request that other segments be added to its address space, segments which could contain code that it could then execute. This allowed applications to automatically use the latest version of any external routine they called, since those routines were kept in other segments, which were dynamically linked only when a process first tried to begin execution in them. Since different processes could use different search rules, different users could end up using different versions of external routines automatically. Equally importantly, with the appropriate settings on the Multics security facilities, the code in the other segment could then gain access to data structures maintained in a different process.

Thus, to interact with an application running in part as a daemon (in another process), a user's process simply performed a normal procedure-call instruction to a code segment to which it had dynamically linked (a code segment that implemented some operation associated with the daemon). The code in that segment could then modify data maintained and used in the daemon. When the action necessary to commence the request was completed, a simple procedure return instruction returned control of the user's process to the user's code.

Multics also supported extremely aggressive on-line reconfiguration: central processing units, memory banks, disk drives, etc. could be added and removed while the system continued operating. At the MIT system, where most early software development was done, it was common practice to split the multiprocessor system into two separate systems during off-hours by incrementally removing enough components to form a second working system, leaving the rest still running the original logged-in users. System software development testing could be done on the second system, then the components of the second system were added back to the main user system, without ever having shut it down. Multics supported multiple CPUs; it was one of the earliest multiprocessor systems.

Multics was the first major operating system to be designed as a secure system from the outset.[8] Despite this, early versions of Multics were broken into repeatedly.[9] This led to further work that made the system much more secure and prefigured modern security engineering techniques. Break-ins became very rare once the second-generation hardware base was adopted; it had hardware support for ring-oriented security, a multilevel refinement of the concept of master mode. A US Air Force tiger team project tested Multics security in 1973 under the codeword ZARF. On 28 May 1997, the American National Security Agency declassified this use of the codeword ZARF.

Multics was the first operating system to provide a hierarchical file system,[10][11] and file names could be of almost arbitrary length and syntax. A given file or directory could have multiple names (typically a long and short form), and symbolic links between directories were also supported. Multics was the first to use the now-standard concept of per-process stacks in the kernel, with a separate stack for each security ring. It was also the first to have a command processor implemented as ordinary user code – an idea later used in the Unix shell. It was also one of the first written in a high-level language (Multics PL/I), after the Burroughs MCP system written in ALGOL.[1]

The deployment of Multics into secure computing environments also spurred the development of innovative supporting applications. In 1975, Morrie Gasser of MITRE Corporation developed a pronounceable random word generator to address password requirements of installations such as the Air Force Data Services Center (AFDSC) processing classified information. To avoid guessable passwords, the AFDSC decided to assign passwords but concluded the manual assignment required too much administrative overhead. Thus, a random word generator was researched and then developed in PL1. Instead of being based on phonemes, the system employed phonemic segments (second order approximations of English) and other rules to enhance pronounceability and randomness, which was statistically modeled against other approaches.[12] A descendant of this generator was added to Multics during Project Guardian. [13]

Project history

In 1964, Multics was developed initially for the GE-645 mainframe, a 36-bit system. GE's computer business, including Multics, was taken over by Honeywell in 1970; around 1973, Multics was supported on the Honeywell 6180 machines, which included security improvements including hardware support for protection rings.

Bell Labs pulled out of the project in 1969; some of the people who had worked on it there went on to create the Unix system. Multics development continued at MIT and General Electric.

Honeywell continued system development until 1985. About 80 multimillion-dollar sites were installed, at universities, industry, and government sites. The French university system had several installations in the early 1980s. After Honeywell stopped supporting Multics, users migrated to other systems like Unix.

In 1985, Multics was issued certification as a B2 level secure operating system using the Trusted Computer System Evaluation Criteria from the National Computer Security Center (NCSC) a division of the NSA, the first operating system evaluated to this level.

Multics was distributed from 1975 to 2000 by Groupe Bull in Europe, and by Bull HN Information Systems Inc. in the United States. In 2006, Bull SAS Free softwared Multics versions MR10.2, MR11.0, MR12.0, MR12.1, MR12.2, MR12.3, MR12.4 & MR12.5.[14]

The last known Multics installation running natively on Honeywell hardware was shut down on October 30, 2000, at the Canadian Department of National Defence in Halifax, Nova Scotia, Canada.[15]

Current status

In 2006 Bull HN released the source code for MR12.5, the final 1992 Multics release, to MIT.[16] Most of the system is now available as Free software with the exception of some optional pieces such as TCP/IP.[17]

In 2014 Multics was successfully run on current hardware using an emulator.[18] The 1.0 release of the emulator is now available.[19] Release 12.6f of Multics accompanies the 1.0 release of the emulator, and adds a few new features, including command line recall and editing using the video system.[20]

Commands

The following is a list of programs and commands[21] for common computing tasks that are supported by the Multics command-line interface.[22][23]

- apl
- ceil
- change_wdir (cwd)
- cobol
- copy (cp)
- echo
- emacs
- floor
- fortran (ft)
- gcos (gc)
- help
- home_dir (hd)
- if
- list (ls)
- login (l)
- logout
- ltrim
- mail (ml)
- pascal
- pl1
- print (pr)
- print_wdir (pwd)
- runoff (rf)
- rtrim
- sort
- teco
- trunc
- where (wh)
- who
- working_dir (wd)

Retrospective observations

Peter H. Salus, author of a book covering Unix's early years,[24] stated one position: "With Multics they tried to have a much more versatile and flexible operating system, and it failed miserably".[25] This position, however, has been widely discredited in the computing community because many of Multics' technical innovations are used in modern commercial computing systems.[7]

The permanently resident kernel of Multics, a system derided in its day as being too large and complex, was only 135 KB of code. In comparison, a Linux system in 2007 might have occupied 18 MB.[26] The first MIT GE-645 had 512 kilowords of memory (2 MiB), a truly enormous amount at the time, and the kernel used only a moderate portion of Multics main memory.

The entire system, including the operating system and the complex PL/1 compiler, user commands, and subroutine libraries, consisted of about 1500 source modules. These averaged roughly 200 lines of source code each, and compiled to produce a total of roughly 4.5 MiB of procedure code, which was fairly large by the standards of the day.

Multics compilers generally optimised more for code density than CPU performance, for example using small sub-routines called *operators* for short standard code sequences, which makes comparison of object code size with modern systems less useful. High code density was a good optimisation choice for Multics as a multi-user system with expensive main memory.

During its commercial product history, it was often commented internally that the Honeywell Information Systems (HIS) (later Honeywell-Bull) sales and marketing staff were more familiar with and comfortable making the business case for Honeywell's other computer line, the DPS 6 running GCOS. The DPS-6 and GCOS was a well-regarded and reliable platform for inventory, accounting, word processing, and vertical market applications, such as banking, where it had a sizeable customer base. In contrast, the full potential of Multics' flexibility for even mundane tasks was not easy to comprehend in that era and its features were generally outside the skill set of contemporary business analysts. The scope of this disconnect was concretized by an anecdote conveyed by Paul Stachour, CNO/CSC:

> When American Telephone and Telegraph was changing its name to just AT&T in 1983, a staffer from Honeywell's legal department showed up and asked a Multician if he could arrange to have the name changed in all of their computerized documents. When asked when the process could be completed, the Multician replied, "It's done." The staffer repeated that he needed *hundreds perhaps thousands* of documents updated. The Multician explained that he had executed a global search and replace as the staffer was speaking, and the task was in fact completed.

Influence on other projects

Unix

The design and features of Multics greatly influenced the Unix operating system, which was originally written by two Multics programmers, Ken Thompson and Dennis Ritchie. Superficial influence of Multics on Unix is evident in many areas, including the naming of some commands. But the internal design philosophy was quite different, focusing on keeping the system small and simple, and so correcting some perceived deficiencies of Multics because of its high resource demands on the limited computer hardware of the time.

The name *Unix* (originally *Unics*) is itself a pun on *Multics*. The *U* in Unix is rumored to stand for *uniplexed* as opposed to the *multiplexed* of Multics, further underscoring the designers' rejections of Multics' complexity in favor of a more straightforward and workable approach for smaller computers. (Garfinkel and Abelson[27] cite an alternative origin: Peter Neumann at Bell Labs, watching a demonstration of the prototype, suggested the pun name UNICS – pronounced "eunuchs" – as a "castrated Multics", although Dennis Ritchie is said to have denied this.[28])

Ken Thompson, in a transcribed 2007 interview with Peter Seibel[29] refers to Multics as "overdesigned and overbuilt and over everything. It was close to unusable. They [Massachusetts Institute of Technology] still claim it's a monstrous success, but it just clearly wasn't". On the influence of Multics on Unix, Thompson stated that "the things that I liked enough (about Multics) to actually take were the hierarchical file system and the shell — a separate process that you can replace with some other process".

Other operating systems

The Prime Computer operating system, PRIMOS, was referred to as "Multics in a shoebox" by William Poduska, a founder of the company. Poduska later moved on to found Apollo Computer, whose AEGIS and later Domain/OS operating systems, sometimes called "Multics in a matchbox", extended the Multics design to a heavily networked graphics workstation environment.

The Stratus VOS operating system of Stratus Computer (now Stratus Technologies) was very strongly influenced by Multics, and both its external user interface and internal structure bear many close resemblances to the older project. The high-reliability, availability, and security features of Multics were extended in Stratus VOS to support a new line of fault tolerant computer systems supporting secure, reliable transaction processing. Stratus VOS is the most directly-related descendant of Multics still in active development and production usage today.

The protection architecture of Multics, restricting the ability of code at one level of the system to access resources at another, was adopted as the basis for the security features of ICL's VME operating system.

See also

- Time-sharing system evolution
- Peter J. Denning
- Jack B. Dennis
- Roberto Mario Fano – director of Project MAC at MIT (1963–1968)
- Robert M. Graham
- J. C. R. Licklider – director of Project MAC at MIT (1968–1971)
- Peter G. Neumann
- Elliott Organick
- Louis Pouzin – introduced the term *shell* for the command language used in Multics
- Jerome H. Saltzer
- Roger R. Schell
- Glenda Schroeder – implemented the first command line user interface shell and proposed the first email system with Pouzin and Crisman
- Victor A. Vyssotsky

References

1. R. A. Freiburghouse, "The Multics PL/1 Compiler" (http://www.multicians.org/pl1-raf.html), General Electric Company, Cambridge, Massachusetts, 1969.
2. "Multics License (Multics) - Open Source Initiative" (http://opensource.org/licenses/Multics). *opensource.org*. Retrieved April 11, 2018.
3. "Myths about Multics" (http://www.multicians.org/myths.html#source). *www.multicians.org*. Retrieved April 11, 2018.
4. Dennis M. Ritchie, "The Evolution of the Unix Time-sharing System", Communications of the ACM, Vol. 17, 1984, pp. 365-375.
5. Dan Murphy (1996) [1989]. "Origins and Development of TOPS-20" (http://tenex.opost.com/hbook.html).
6. Gregory, Nathan (May 2018). *The Tym Before* (https://www.google.com/books/edition/The_Tym_Before/kn9eDwAAQBAJ?hl=en&gbpv=1&bsq=page%2066). Lulu.com. p. 66. ISBN 9781387824755. Retrieved March 29, 2020.
7. "Myths about Multics" (http://www.multicians.org/myths.html). *www.multicians.org*. Retrieved April 11, 2018.
8. Jerome H. Saltzer, "Protection and the Control of Information Sharing in Multics", in "Introduction to Multics", MAC TR-123, Project MAC, Cambridge, February 1974; pg. 2-41.

9. Tom Van Vleck (2002). "How the Air Force cracked Multics Security" (http://www.multicians.org/security.html).
10. "Multics Glossary -F-" (http://www.multicians.org/mgf.html#filesystem). *www.multicians.org*. Retrieved April 11, 2018.
11. R. C. Daley and P. G. Neumann, "A general-purpose file system for secondary storage" (http://dl.acm.org/citation.cfm?id=1463915), AFIPS '65 (Fall, part I) Proceedings of the November 30 – December 1, 1965
12. "A Random Word Generator for Pronounceable Passwords" (https://apps.dtic.mil/sti/pdfs/ADA017676.pdf¬¬). Bedford, MA: Electronic Systems Division, Air Force Systems Command, USAF. November 1975. ESD-TR-75-97. Retrieved March 8, 2021.
13. Van Vleck, Tom. "Password Generator" (https://multicians.org/thvv/gpw-js.html). Retrieved March 8, 2021.
14. Multics history (http://stuff.mit.edu/afs/athena/reference/multics-history/) MIT
15. "Multics History Dates" (http://www.multicians.org/chrono.html). Retrieved September 13, 2015. "Shutdown of DND-H (17:08Z 10/30/00)"
16. Van Vleck, Tom. "Open Source for Multics" (http://web.mit.edu/multics-history/#community). *Multicians.org*. Retrieved April 11, 2016.
17. Anthony, Charles. "(email) Re: [dps8m-developers] Multiprocessor and/or networked Multics" (https://sourceforge.net/p/dps8m/mailman/message/34310157/). *Sourceforge.net*. Retrieved April 11, 2016.
18. "RingZero - Multics reborn" (http://ringzero.wikidot.com/start). *WikidotCom*. Retrieved April 11, 2015.
19. "Multics Simulator" (http://multicians.org/simulator.html). Retrieved July 9, 2017.
20. "Installing Multics" (http://multics-wiki.swenson.org/index.php/Main_Page#Installing_Multics). Retrieved May 19, 2020.
21. Honeywell Bull, Inc. (February 1985). *Multics Commands and Active Functions (AG92-06)* (http://www.bitsavers.org/pdf/honeywell/multics/AG92-06B_multicsCmds_Nov87.pdf) (PDF). Retrieved January 10, 2021.
22. Unix and Multics (https://www.multicians.org/unix.html)
23. Multics Commands (https://www.multicians.org/multics-commands.html)
24. Salus, Peter H. (1994). *A quarter century of UNIX* (Reprinted with corrections Jan. 1995. ed.). Reading, Mass.: Addison-Wesley Pub. Co. ISBN 978-0-201-54777-1.
25. Ward, Mark (August 20, 2009). "40 years of Unix" (http://news.bbc.co.uk/2/hi/technology/8205976.stm). *BBC News*. Retrieved April 27, 2010. Quoting Peter Salus.
26. Collings, Terry; Wall, Kurt (April 10, 2007). *Red Hat Linux Networking and System Administration* (https://books.google.com/books?id=PrepsktXgScC&pg=PA668) (3rd ed.). John Wiley & Sons. p. 668. ISBN 978-0-7645-9949-1. Retrieved February 4, 2017.
27. Garfinkel, Simson and Abelson, Harold. Architects of the Information Society: Thirty-Five Years of the Laboratory for Computer Science at MIT. MIT Press, 1999. ISBN 978-0262071963
28. Karn, Phil (October 28, 1981). "Origins of unix" (http://article.olduse.net/4743@Aucbvax.UUCP). Newsgroup: fa.unix-wizards (news:fa.unix-wizards). Usenet: 4743@Aucbvax.UUCP (news:4743@Aucbvax.UUCP). Retrieved April 11, 2014.
29. Peter Seibel. Coders at Work: Reflections on the Craft of Programming. APress Publications, 2007. ISBN 978-1-4302-1948-4

Further reading

The literature contains a large number of papers about Multics, and various components of it; a fairly complete list is available at the Multics Bibliography (http://www.multicians.org/biblio.html) page and on a second, briefer 1994 Multics bibliography (http://web.mit.edu/srz/www/multics-bibliography.html) (text format). The most important and/or informative ones are listed below.

- F. J. Corbató, V. A. Vyssotsky, *Introduction and Overview of the Multics System* (http://www.multicians.org/fjcc1.html) (AFIPS 1965) is a good introduction to the system.
- F. J. Corbató, C. T. Clingen, J. H. Saltzer, *Multics – The First Seven Years* (http://www.multicians.org/f7y.html) (AFIPS, 1972) is an excellent review, written after a considerable period of use and improvement over the initial efforts.
- J. J. Donovan, S. Madnick, Operating Systems (http://catalog.loc.gov/cgi-bin/Pwebrecon.cgi?v1=6&ti=1,6&Search_Arg=Donovan%2C%20John%20J.&Search_Code=NAME%40&CNT=100&PID=fahdDtlh7Y1ODWe5X_fO2_UUafeDT&SEQ=20090914123346&SID=1), is a fundamental read on operating systems.
- J. J. Donovan, Systems Programming (http://catalog.loc.gov/cgi-bin/Pwebrecon.cgi?v1=5&ti=1,5&Search_Arg=Donovan%2C%20John%20J.&Search_Code=NAME%40&CNT=100&PID=fahdDtlh7Y1ODWe5X_fO2_UUafeDT&SEQ=20090914123346&SID=1), is a good introduction into systems programming and operating systems.

Technical details

- Jerome H. Saltzer, *Introduction to Multics (https://web.archive.org/web/20060918161127/http://www.lcs.mit.edu/publications/specpub.php?id=691)* (MIT Project MAC, 1974) is a considerably longer introduction to the system, geared towards actual users.
- Elliott I. Organick, *The Multics System: An Examination of Its Structure* (MIT Press, 1972) is the standard work on the system, although it documents an early version, and some features described therein never appeared in the actual system.
- V. A. Vyssotsky, F. J. Corbató, R. M. Graham, *Structure of the Multics Supervisor (http://www.multicians.org/fjcc3.html)* (AFIPS 1965) describes the basic internal structure of the Multics kernel.
- Jerome H. Saltzer, *Traffic Control in a Multiplexed Computer System (https://web.archive.org/web/20011127053649/http://ncstrl.mit.edu/Dienst/UI/2.0/Describe/ncstrl.mit_lcs/LCS-TR-30?abstract=saltzer)* (MIT Project MAC, June 1966) is the original description of the idea of switching kernel stacks; one of the classic papers of computer science.
- R. C. Daley, P. G. Neumann, *A General Purpose File System for Secondary Storage (http://www.multicians.org/fjcc4.html)* (AFIPS, 1965) describes the file system, including the access control and backup mechanisms.
- R. J. Feiertag, E. I. Organick, *The Multics Input/Output System (http://www.multicians.org/rjf.html)*. Describes the lower levels of the I/O implementation.
- A. Bensoussan, C. T. Clingen, R. C. Daley, *The Multics Virtual Memory: Concepts and Design (http://www.multicians.org/multics-vm.html)*, (ACM SOSP, 1969) describes the Multics memory system in some detail.
- Paul Green, *Multics Virtual Memory – Tutorial and Reflections (ftp://ftp.stratus.com/pub/vos/multics/pg/mvm.html)* is a good in-depth look at the Multics storage system.
- Roger R. Schell, *Dynamic Reconfiguration in a Modular Computer System* (MIT Project MAC, 1971) describes the reconfiguration mechanisms.

Security

- Paul A. Karger, Roger R. Schell, *Multics Security Evaluation: Vulnerability Analysis (http://csrc.nist.gov/publications/history/karg74.pdf)* (Air Force Electronic Systems Division, 1974) describes the classic attacks on Multics security by a "tiger team".

- Jerome H. Saltzer, Michael D. Schroeder, *The Protection of Information in Computer Systems (http://cap-lore.com/CapTheory/ProtInf/)* (Proceedings of the IEEE, September 1975) describes the fundamentals behind the first round of security upgrades; another classic paper.
- M. D. Schroeder, D. D. Clark, J. H. Saltzer, D. H. Wells. *Final Report of the Multics Kernel Design Project (https://web.archive.org/web/20011124185621/http://ncstrl.mit.edu/Dienst/UI/2.0/Describe/ncstrl.mit_lcs/LCS-TR-196?abstract=saltzer)* (MIT LCS, 1978) describes the security upgrades added to produce an even more improved version.
- Paul A. Karger, Roger R. Schell, *Thirty Years Later: Lessons from the Multics Security Evaluation (http://www.acsac.org/2002/papers/classic-multics.pdf)* (IBM, 2002) is an interesting retrospective which compares actual deployed security in today's hostile environment with what was demonstrated to be possible decades ago. It concludes that Multics offered considerably stronger security than most systems commercially available in 2002.

External links

- multicians.org (https://www.multicians.org/) is a comprehensive site with a lot of material
  - Multics papers online (http://www.multicians.org/papers.html)
  - Multics glossary (http://www.multicians.org/mga.html)
  - Myths (http://www.multicians.org/myths.html) discusses numerous myths about Multics in some detail, including the myths that it failed, that it was big and slow, as well as a few understandable misapprehensions
  - Multics security (http://www.multicians.org/security.html)
  - Unix and Multics (http://www.multicians.org/unix.html)
  - Multics general info and FAQ (http://www.multicians.org/general.html) Includes extensive overview of other software systems influenced by Multics
- Honeywell, Inc., MULTICS records, 1965–1982 (http://purl.umn.edu/41295). Charles Babbage Institute, University of Minnesota. Multics development records include the second MULTICS System Programmers Manual; MULTICS Technical Bulletins that describe procedures, applications, and problems, especially concerning security; and returned "Request for Comments Forms" that include technical papers and thesis proposals.
- Official source code archive at MIT (http://web.mit.edu/multics-history/)
- Link page to various Multics information (http://webarchive.loc.gov/all/20011130013712/http://www.mit.edu:8001/afs/net/user/srz/www/multics.html) at the Library of Congress Web Archives (archived 2001-11-30)
- Multics repository at Stratus Computer (ftp://ftp.stratus.com/pub/vos/multics/multics.html)
- Multics at Universitaet Mainz (http://www.vaxman.de/historic_computers/multics/multics.html)
- Active project to emulate the Honeywell dps-8/m Multics CPU (http://sourceforge.net/projects/dps8m/)
- Various scanned Multics manuals (http://bitsavers.org/pdf/honeywell/multics/)
- Multicians.org and the History of Operating Systems (http://www.cbi.umn.edu/iterations/haigh.html), a critical review of Multicians.org, plus a capsule history of Multics.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Multics&oldid=1025954404"