G0102

file w/ Multics papers

DATE: JULY 29, 1971

TO: GOO DISTRIBUTION

FROM: D. R. VINOGRAD

SUBJECT: WHAT'S A SYSTEM TO DO? - ASSURING SYSTEM DATA INTEGRITY

The attached paper entitled "What's a System to do? - Assuring System Data Integrity" is to be presented at the IEEE Conference which will be held in Boston on September 22-24, 1971.

/11

WHAT'S A SYSTEM TO DO? - ASSURING SYSTEM DATA INTEGRITY

David R. Vinograd
Cambridge Information Systems Laboratory
Honeywell Information Systems, Incorporated
575 Technology Square
Cambridge, Massachusetts 02139

Summary

Multics, a multiple access computer system, has employed various software approaches to meet its reliability goals. These approaches include a comprehensive system protection scheme and the on-line reconstruction of system data bases when errors are detected. Considerable attention has been paid to system failure recovery methods which preserve system data integrity.

Multics[1,2] is a multi-user remote access system being developed jointly by Project MAC of Massachusetts Institute of Technology and Cambridge Information Systems Laboratory of Honeywell Information Systems, Incorporated. The system is implemented in the PL/I language. Multics provides, for its users, a virtual memory within which exists a tree-structured directory hierarchy. The virtual memory implementation is supported through both paging and segmentation.[3] System protection is provided by an access control scheme based on concentric rings of descending authority complemented by control of individual file access.[4]

Multics is designed to be a computer utility and is presently providing daily round-the-clock service to the MIT community. Its reliability must be such that 24-hour operation is standard. Administrative functions, such as billing and user registration must be operational in an on-line environment as must all system services such as automatic file backup, where, independent of the user, copies of his most recently modified files are periodically written onto tape. Repair of minor system bugs and programming deficiencies must also be possible without an interruption of service.

In general, the Multics system has attempted to prevent, anticipate, and recover from as many errors as possible, without imposing any severe constraints on its users. The design goals were as follows:

(1) The system should recover from the loss of data in a manner which does not disrupt any user initiated task.

(2) If the above is not possible then the system should recover in a manner which is not damaging to continued system operation.

(3) If the above is not possible then the system must be capable of being restarted with a low data loss in a short time period.

Within the bounds imposed by the above rules, it was recognized that system reliability and performance do interact and the trade-offs must be considered. Further, a system implementation strategy has been to deal with those error conditions which most often occur or have the greatest severity, before attempting to solve those errors which seldom happen.

A major Multics design problem was to prevent unauthorized access to both system and user files. A user's identity is verified when he logs in. Once logged in the user cannot be allowed to reference a file to which he has not been given access. Control of user access to a file is specified by user name and can be manipulated by the user if he has the proper permission. The user can specify read, write, or execute permission or a combination of these attributes for such other users as he desires. The system specifies user access to system library data and procedures via the same mechanism. A result of the virtual memory implementation is that file access can be enforced by the hardware and is checked during each memory reference.

To protect the system from the user, the system operates in a ring other than that of the user. When the user makes a request of the system he is switched to a ring of higher authority, the system's ring. The points of entry into this ring of higher authority are completely defined and access to them is controlled. The act of ring switching invokes the mechanism of argument validation. Arguments supplied by user programs to system procedures are checked as to access, type and number. Where information is to be returned, user access to the return area is also checked. The system also uses the access control mechanism to protect itself from itself, (i.e., from buggy code or hardware malfunction). System data and procedures have only as much access as they require for operational purposes, and should an error occur inadvertant modification will be trapped by the hardware before any damage is done.

It is an inescapable fact of large system operation that errors will occur; their occurrence must be anticipated. When an error does occur the data loss can be minimized by good design of both data bases and procedures that manipulate them. Multics has used this design approach as described below.

To protect one user from another, data bases in the system have been decentralized on a per-user basis wherever feasible. These data bases exist in disjoint address spaces and the invalidation of one, while affecting its user, does no harm to the other users of the system. Where this decentralization was not feasible, such as system data bases, the data bases have been designed to serve very specific functions, such as I/O terminal buffers. Management of each type of data base is centralized to one procedure as much as possible. This is done both to aid system debugging and to insure system control over the data base contents.

Where a data base is shared and may be modified by more than one user at a time, a locking strategy must be used. In Multics, locks can be either read or write locks. If the system desires to read a data base it must first lock it. If writing is desired a further indicator is set. Thus, as long as the lock is not set in the 'write' mode then the data is assumed to be consistent. To further control system access, some data bases are set to hardware read access until they are locked.

In any large and complex computer operation, hardware failures will occur. In many cases, Multics can recover gracefully. For example, I/O errors on secondary storage are retried while malfunctioning I/O terminals are disconnected. Memory parity errors and other hardware faults are also usually recoverable. In the case of such hardware errors the system operator is informed about the occurrence of the error and the state of the machine at the time the error was detected. User-caused "errors" such as illegal machine operations are reported directly to the user on his console with appropriate supplementary information.

Any errors which occur while operating in the system's ring within the Multics supervisor are of a much more critical nature than those which occur in the user's ring. System data bases must be intact and consistent before control can be returned to the user's ring. When an error is detected a PL/I condition is signalled and the handler for the condition is invoked. This handler can then verify the consistency of any locked system data bases and take appropriate action. If the data base is only read-locked it can be unlocked. If the data base is write-locked then it must be checked for consistency. This 'checkout' is referred to as "salvaging". For example, the on-line directory "salvager" is invoked to rebuild inconsistent directory data bases before control is returned to the user's ring. A similar scheme is used by the procedures that allocate remote access terminal buffers and the technique will be extended to other system data base managers as the need develops.

Certain types of system errors are extremely difficult to recover from and usually cause system failure. In this event, the system must be restarted with minimum information loss. If the information in main core can be updated into secondary storage then data consistency is

preserved and no further work need be done. If this cannot be accomplished then a free-standing "salvager" subsystem is used. This salvager has two main functions. First, it performs consistency checks on Multics directories, rebuilding them when necessary. Second, it validates the secondary storage assignment for each file. The salvager views the entire Multics directory hierarchy as questionable data and is capable of rebuilding from incomplete information a new consistent directory and replacing the older copy.

Should the salvager be unable to completely restore the integrity of the directory hierarchy and its contents, the lost information can be recovered from tapes created by the automatic file backup facility. This background service, which is invisible to the user, periodically scans the directory hierarchy for modified files and writes a copy onto tape. Less frequently, all files in the directory hierarchy are copied onto tape.

Multics employs other techniques to minimize service interruptions. The system is capable of dynamic reconfiguration of processors, memory and I/O channels without service interruptions. Thus, maintenance and repair of various pieces of hardware can be completed off-line without the user being aware of the change. This feature also allows the system operator to selectively remove a hardware module that is failing. The system's hardware clock also provides an alarm clock facility. When the alarm goes off an interrupt is generated. The alarm feature is used by I/O software as a 'time-out' to restart I/O traffic should a peripheral interrupt get lost.

Multics is still under development but is daily proving itself as a service within the MIT user community. Its availability at all hours, day in, day out, is due to a great extent to the techniques outlined here and efforts to improve system reliability are continuing.

References:
1) Corbato, F.J. and Vyssotsky, V.A.
"Introduction and Overview of the Multics System"
AFIPS Conf.Proc.27 (1965 FJCC)
Spartan Books,Washington,D.C.1967, pp 185-196
2) Multics Programmers' Manual, Revision #7
Copyright 1969, 1970, 1971
Massachusetts Institute of Technology
3) Bensoussan, A, Clingen, C.T. and Daley, R.C.
"The Multics Virtual Memory"
ACM Second Symposium on Operating System Principles, October 20-22, 1969
Princeton University pp 30-42
4) Graham, R.M.
"Protection in an Information Processing Utility"
Comm. ACM 11, 5(May 1968) 365-369