Appendix

COMPARISON OF THE MULTICS SYSTEM AND CTSS

by F. J. Corbató

November 8, 1968

(Note: The * designates features which are not yet complete in Multics, but which are planned.)

The first topics are present both in CTSS and Multics, but are listed merely for completeness for they are considered to be of critical importance in both systems.

1. Access Control on Files and File Sharing on a Copy Basis

Access to files is controlled on both a per-user and per-class-of-user basis. While files may be shared, different users must each obtain a copy. Thus in any application where there is frequent up-dating of the files, the different users will have inconsistent information unless special precautions are taken.

2. File Backup and File Retrieval

File backup refers to the ability to completely restore the secondary storage contents in the event of major catastrophe such as: fire, disk crash etc. File retrieval refers to the ability to restore individual files when they become inexplicably damaged. File retrieval allows an individual to be given some satisfaction without forcing the work of all users of the system to be backed up in time. These two aspects have been key in the success of the CTSS system.

* 3. Incremental Dumping and Creation of the File Backup Tapes

This phrase refers to a refinement of the previous one; it is the ability to continuously backup the secondary storage (except for a brief time lag.) The issue becomes an important one as the system size grows and it becomes increasingly awkward to use a brute-force complete dump of secondary storage.

4. Password Logic at Login for Each User

This refers to the ability to authenticate the existence of a user at a terminal and to properly control, thereafter, his access to files within the system. This also is the means that the background jobs are prevented from having an unauthorized backdoor to the file system since the dispatching of background jobs is expected to be done through an on-line console.

* 5. User Accounting

There is obvious need to be able to account for the diverse use of resources and to distribute the system costs in an equitable fashion. Since Multics is capable of expanding its capacities in various directions this aspect will be especially important.

* 6. Automatic Logout and Saving of User Status

Whenever, time-sharing system operation must be terminated on either a scheduled or unscheduled basis, it is essential to be able to gracefully shut-down each users process in such a way that he can restore without error the operation of this process when system operation resumes. This obviously desirable property turns out to be non-trivial to implement, because of the potential for a vastly differently configured system on the resumption of operation.

7. A Command System Which is Open-Ended

This phrase refers to the ability of the user to create programs which are commands of his own choosing and which he can use on the same footing as those supplied with the initial system. This issue has especially large import when it is realized that in an effective time-sharing system many of the better commands are developed by the users rather than by a system programming staff. It is considered important that there be no requirement of reprogramming in order to utilize user contributions.

The remaining features listed are important improvements in Multics which are not present in CTSS.

8. An Expandable Number of Processors Treated Homogeneously

This is a property which allows a system gracefully to expand in capacity and be maintained during 24-hour a day operation. In addition, the property assists in the isolation of major unit malfunctions.

9. An Expandable Amount of Core Memory Treated Homogeneously

The reasons for this feature are similar for those given for processors.

10. The Ability to Quickly Reconfigure the System Without Recompilation

The problem here is that as soon as one begins to develop pools of processors and memory etc., with specific port assignments and cables, the potential number of system configurations rapidly reaches an incredibly large number. For this reason it is essential that the proces of initializing a system from a system tape be a generative one since reconfiguration is expected frequently. (Multics initialization currently requires about three minutes to establish the hard core supervisor and about two more minutes to establish the basic system commands.)

11. Isolation of the User from the Configuration

This feature is the notion of the users programming for a virtual machine. It is an essential feature with any machine which changes configuration frequently. CTSS avoided this problem by not being reconfigurable.

12. Sharing of Segments

Access control for files (segments) with the ability to share segments dynamically in core memory is probably the single most important aspect of the Multics system. It is through this seemingly innocent feature that one has the ability to develop systems wherein it is possible for users to work in concert in real-time. Sharing of segments allows cooperating users to selectively merge their computations without losing control of the privacy of their work. Many other properties which are listed below are intimately connected with this idea.

13. Individual Segment Attributes

The ability to have data bases with independently varying lengths, as well as, with independent attributes such as read_only, write, etc., gives great flexibility and convenience to a programmer. In particular, he is spared ugly difficulties such as overlays.

14. The Ability to have Rings of Protection Between Sections of a Program

This property, which is invokable by the user, allows him to declare that certain segments are in a separate ring of protection. In this way, he is able to develop a superior/inferior relationship between one group of subroutines and another. This is particularly valuable in applications such as teachers and students interacting by means of a grading program.

15. The Ability to Simultaneously Operate Separate Time-Sharing Systems

This property allows one to have different classes of users who believe they are working with different systems. It also allows a certain amount of system program checkout of future systems without having to take over the machine for another purpose. (Of course, the hardcore functions of the system must be invariant for all users, but these are normally masked by the particular versions of the command language interpreter and login command.)

16. Dynamic Linking or Pre-Binding

Here the user is allowed a choice in Multics of whether to allow subroutines to be discovered on an as-needed basis or to decide to pre-relate subroutines together in a collection so as to eliminate unnecessary linking overhead. The property of dynamic linking is felt to be especially important as one develops larger and larger complexes of programs where there are increasingly uncertain paths of the flow of control.

17. More Flexible Library Search Mechanisms

Whether one is loading, binding, or dynamic linking, a directed search is required to discover the location in the file system of a subroutine with a given name. In Multics this search can be through a sequence of directories according to search rules which may vary from user to user.

* 18. Interchangeability of Absentee and On-Line Execution of Jobs

What is meant here is the ability to run a job interactively or non-interactively, depending on one's wishes, with a minimum of fuss required to switch the job from one mode to the other. In particular, in Multics there are no restrictions on: the number of such jobs, when one switches, or how often the switching process is reversed. The latter property, like that of automatic logout, is extremely difficult to achieve unless considerable care has been spent in the system design.

* 19. I/O Streams are Switchable by User Commands

What is meant by this phrase is that it is possible for a user to issue a command such that a program may receive its input (or send its output) from a different device then previously established, without any modification to the program. It is in this way that much of the simplicity of interchangeability of absentee users and on-line users is achieved. In particular, when an interactive job is to be run non-interactively, the interactive input must be presented to the program as a file of input in place of the typewriter.

20. Hierarchical File System

There is a hierarchical file system with the ability to make links. The hierarchical file system allows the user to refer, with economy of expression, to a large number of different collections of files. Because a hierarchical file system is not always appropriate for the organization of information (for example, all the persons with telephones are not necessarily subordinate to those who have T. V. sets), the extra flexibility of having arbitrary links between directory branches and files in other directories is highly desirable. (The links in the Multics system avoid the design flaw in CTSS which makes removing links difficult to handle.)

* 21. Multilevel Storage Management

The drum, disc, and tape memories of Multics are treated by the system as the beginning of a sequence of devices which have successively longer access times. Neither the bulk of the system nor the user must concern itself with the device that information is stored on; instead, an automatic algorithm is employed such that the more recently used files are kept on the fast access drum, the intermittently used files on disc, and the rarely used files on tape. Eventually, it is expected that a user should be able, if he wishes, to override the default action of the algorithm, provided he is prepared to pay the information storage rental costs associated with a fast access device.

22. Multiple Processes on One Job

It is possible for any user to develop a program with explicit parallelism by means of separate processes. In this way, either by the automatic multi-programming or by multi-processing he can gain the execution speed benefit of expressing his problem in this way. This kind of approach should become important in areas such as weather prediction, etc.

23. Flexibility for New I/O Devices

The design of the I/O system is such that there has been an attempt to make it possible to add conveniently and easily devices as they become of interest to Multics users. The major reason that this flexibility is present is because of the attention which has been given to functional modularity. The importance of this feature can be understood by observing that CTSS has been successively adapted to 14 different typewriter terminal devices during its brief history.

24. Economic Benefits

a) pure procedure programs
b) multi-programming
c) paging

The above items are considered to be techniques to achieve an end of more effective utilization of the equipment. As such, they should not immediately affect the quality of the services offered any user of the system, but rather the cost. In Multics, at the scale it is presently operating, only minor economies are being realized from these features. But as the scale of a Multics system increases, or as the system becomes used in a specialized application, then these economies will increase. For example, when all users of a system employ a specific sub-system, the pure procedure feature allows an immense reduction in the storage space required per user.

25. ASCII Character Set

The full ascii 128 character set is uniformly used throughout the system. It is difficult to explain the incredible confusion which arises when there is more than one character set within a given machine. In CTSS, although upper and lower case letters were allowable, it was always necessary to do specialized programming and be extremely conscious of interface problems. A great many specific codes were necessary to constantly convert back and forth between the various conventions. Multics has a clean start and did not inherit the problem of being compatible with batch systems.

In a similar way, the system software supports files and programs with names up to 32 characters in length (CTSS only allows 6 characters). Again this is a feature which is difficult to add later to a system.

26. <u>Recursion and Stacks</u>

Recursion and a stack mechanism for data storage are available as a system convention and are freely useable if the user wishes. Again, like the character set, this feature is difficult to build-in after the fact except on an ad hoc basis. (It should be noted that stack discipline is of great assistance in debugging.)

27. <u>The System is Flexible Enough for Widely Different Applications</u>

For example, one could utilize Multics as a basis for developing an airline reservation system, with a minimum change to the system. Great attention has been paid to such issues as the ability to have as many as 4,000 relatively idle users without either clogging core memory with system data bases or having poor response time for users.

28. <u>There is Full Instruction Decoding in the Hardware</u>

This is a minor point, but it is an improvement over CTSS which had to rely on the sloppy logic of the IBM 7094. The 7094 did something unpredictable whenever it encountered an illegal op-code. The general effect of full decoding is to increase system reliability and minimize inexplicable malfunctions.

29. <u>The Ability to have Execute_Only Procedures</u>

As time-sharing utilities begin to catch hold, it will become increasingly important for entrepreneurs to have available a variety of means to protect their proprietary investments. One technique is to let users employ programs on a black box basis by allowing the program to be executed, but not read. Multics has this property because of special features in the GE-645 hardware.

30. <u>The System is Designed for Evolution</u>

a) A subset of the PL/I language has been used for implementation.
b) The system is strongly organized along functional lines.
c) System programs are written using the same rules and conventions as those used in programs written by ordinary users.

The above properties have already allowed the Multics system to evolve heavily from its original implementation. While writing a small module, the average system programmer does design iteration in his head, exploring alternatives and rejecting them. In a large system such design iteration is frequently impossible to do before actually exercising the code. Thus, the initial version of a large system resembles a first draft of a small module. Restructuring and streamlining for simplicity becomes a mandatory requirement for an involved system. It is expected that within a year or two, an overall system design will be determined which meets all of the requirements which have been asked of the Multics system. However, change will still be required for two additional reasons. First, users themselves will begin to stress the system in new directions, directions which are based upon the abilities which they already have. This is a phenomenon which was well observed in the CTSS environment. Secondly, as Multics matures and evolves it will undoubtedly be necessary to anticipate its application to future hardware. New hardware will have different economic tradeoffs for various kinds of components and devices. It is expected that some system restructuring will then become desirable. In any case, it should not be necessary to reflect these changes on to the user, except in the improved cost-performance which he should receive.

This concludes the list of the features which Multics is felt to have in contrast to CTSS. The segment sharing feature is considered crucial; a few points like that of op-code decoding are almost trivial. In general, no attempt has been made to estimate here the absolute values of the different features. At present the attitude has been taken that they all are important, although there is a set of priorities as to the order which they will appear. So far, we think we see how to implement a reasonable system without having to sacrifice any of the features.