PROJECT MAC

Computer Systems Research Division

January 26, 1973

Request for Comments No. 2

PROPOSAL FOR CSR WORK BEGINNING JULY 1, 1973

by J. H. Saltzer

Each year Project MAC submits a proposal to its sponsoring organizations for continuation of its work. Enclosed is the introduction and the Computer Systems Research Division portion of the December, 1972, proposal to ARPA, which covers the period beginning in July, 1973.

If you are interested in reading the other parts of the MAC proposal, I have a copy available for loan.

INTRODUCTION

Project MAC proposes to continue its research program under the support of the Advanced Research Projects Agency. During the present 10 month ARPA contract, Project MAC has undergone a considerable reorientation. This included a major reorganization of the laboratory from 12 independent research groups, to four divisions. The four divisions are:

I. Fundamental Studies

II. Computer Systems Research

III. Programming Technology

IV. Automatic Programming

As was previously proposed, we have directed our ARPA supported work towards Research in Automatic Programming, centered in divisions III and IV. Division II herein proposes to embark upon a new ARPA sponsored program of research in the certification of computer programs. Division I is almost entirely supported by the National Science Foundation.

With this proposal we have decisively committed ourselves toward new programs and to the goals of focusing ARPA supported research at Project MAC in one major direction -- Automatic Programming. In addition, we propose to continue our strong involvement in the Network and to continue that part of our inquiries into fundamental studies that is in support of Automatic Programming. We propose to continue the development of systems and languages that support our major activities such as MACSYMA, MUDDLE, PLANNER, CALICO, and a common base language.

We propose to complete the implementation of LISP on Multics and to make MACSYMA and other systems available on Multics.

We propose to begin the task of pioneering the use of very large random access (500 nanosecond cycle time) memory by an arrangement with the M.I.T. Information Processing Center that will, under partial support from ARPA, vastly increase the size of the main fast memory on the Multics system making this extra

large memory available exclusively to Project MAC when needed.

The goal of the new program of research in the certification of computer programs is to develop techniques that will allow one to be assured that a large and complex program performs as specified. This research effort will maintain a close relationship with other ARPA sponsored activity in this area.

Most important is our proposed research in Automatic Programming. We view our task as one of understanding the problems and possibilities of future uses of computers, so that we may today embark on a program of research that will lead to the development of techniques that will use the capabilities of the systems of tomorrow to solve the problems of tomorrow.

The problem that we see most clearly is that the desires for larger, more complex and more sophisticated systems will find no sensible fulfillment in the products of today's programming technology. At IBM there is already underway a major shift of personnel from hardware to software; but merely increasing the number of programmers will not provide the answer. The problem will be, increasingly, that computer systems will not be doing what is wanted -- because of the difficulty of knowing how to program what is wanted, and much more so because of the immutable fact that what is wanted today is different from what was wanted yesterday and is different from what will be wanted tomorrow. The difficulty of modifying a complex system is strangely independent of the difficulty of specifying the change.

We see ways of attacking the root of the problem. The systems of tomorrow must combine three types of knowledge -- knowledge about the program's domain of operation; knowledge about programming and the particular program itself; and most important, some of the common sense knowledge that we all share. We are engaged in the business of learning how to create systems with the first two types of knowledge, in our divisions IV and III, and we have reason to believe that future basic research will show us how to tackle the third type of knowledge.

The remainder of this proposal is comprised of five segments: four segments outling the specific areas of proposed work of the four divisions, and a fifth, separate segment on the budgetary considerations.

DIVISION II - COMPUTER SYSTEMS RESEARCH

The Computer Systems Research division of Project MAC is engaged in experimental studies on the problem of making the engineering of large-scale computer systems into a more methodical discipline. Its recent activities include the measurement and documentation of the Multics system, and the attachment of that system to the ARPA network. For the coming contract period, the division proposes the following four activities:

1. Research in the certification of computer systems (50%)
2. System measurement and modeling activities (15%)
3. Extensions of the ARPA network attachment to Multics (30%)
4. Exporting the Multics design (5%)

The percentages given suggest the fraction of ARPA supplied division resources to be applied to each activity. The first activity is a new one; the other three are continuations of present activities. More detailed descriptions of these activities are the subject of the following three sections.

Research in the Certification of Computer Systems

The Computer System Research division proposes to undertake a major new research project. The goal of this new project is to make possible the certification that a large-scale computer system has been correctly implemented. As will be explained, the initial vehicle for this research will be the security kernel of a modified version of the Multics system.

It is well-known that large-scale computer operating systems have a tendency to be extraordinarily complex, large in size, difficult to maintain, and awkwardly organized. There seem to be several reasons for these tendencies, such as:

- Attempts to stretch the functional capabilities of the system as far as possible.
- Working in a hardware environment which was determined before software requirements were fully understood.
- Attempts to squeeze the hardware and software system to its absolute limit of performance.
- Attempts, because of the high cost of system development, to get the system "on the air" in the absolutely shortest time possible.

Of these four, probably the last two are the strongest contributors to unmaintainable and incomprehensible design, since they both encourage shortcuts to be taken and modularity to be violated against the better judgement of the system designer.

The net effect of these pressures is that most modern large-scale operating systems contain implementation errors which cannot be found except by running the system, and waiting for the errors to be exposed accidently. It is bad enough that timing-dependent errors which cause a system crash may occur just often enough to be disruptive, but not frequently enough to be easily reproduced by a benchmark test (e.g., every two hours.) In addition, there are at least two classes of implementation errors which may not routinely be noticed by running the system against a normal load:

1. Errors which result only in performance degradation of the operating system.
2. Errors in the implementation of the system's information protection mechanisms.

For both of these classes of errors, the system usually appears to correctly process its normal workload; only specialized tests (e.g., reproducible loads with known performance consequences, or test programs which systematically attempt to violate the security mechanisms), which are not part of the usual workload, have any chance of revealing the difficulty.

There appear to be three distinct strategies available for attacking the problem of reducing the probability of implementation errors in a computer operating system:

1. Use a top-down constructive programming approach to the initial design of the operating system, following the principles of Dijkstra and Mills.
2. Subject all of the modules of the operating system, and its inter-module reference pattern, to the emerging techniques of program correctness proofs.
3. Modify an existing operating system so as to simplify its organization sufficiently that it may be subjected to line-by-line auditing by human examiners with high probability that they will detect any inconsistencies, errors, or subversions of the system.

The first of these three methods, while probably most appealing, has two interacting defects. First, there is no way to release the top-down designer from the pressures toward complexity which were mentioned earlier, especially the pressure to get a system operational as soon as possible because of the development expense. Second, unless one can permit the top-down designer the freedom to discard functional requirements which are hard to fit into the design (that is, to simplify the problem being solved by the system under design) then for a large system, a very long and expensive design period seems inevitable, during which time much thought and experimentation goes into devising schemes which methodically do everything called for in the system specifications.

The second method, correctness proofs, is in its infancy -- at the level of theoretical development -- and seems to be far from practical application to a set of programs as large as a typical computer operating system. Work is proceeding at many laboratories on expanding the capabilities of correctness proofs, but it will probably be some time before this technique is applicable.

For these reasons, the third technique, evolution of an existing operating system to simplify it to the point of auditability, is proposed as the subject of this research. This proposal is based on the existence of a fully-equipped, operating, and easily evolvable system: the Multics system, previously developed by the CSR division of Project MAC. Because of the special difficulty in discovering errors in the security and protection areas, and because of the recent growing interest in that area both in private industry and in government and defense circles, the vehicle proposed for initial study is the "protection kernel" of the system.

There are several reasons for choosing this particular system and this particular area. First, Multics has been developed from the ground up to be a protectable system, and it already contains protection mechanisms as advanced as any available, including special hardware features such as "protection rings". Second, the Multics system is better organized than most for evolution and modification, being relatively modular, being largely written in the PL/I language, and allowing initial checkout of most supervisor programs in a user environment. Third, a pool of expertise in the system organization is available at Project MAC, and at Honeywell, whose continued cooperation is assured. Thus, productive work can begin immediately. Also, documentation of the Multics system is now sufficiently well-organized that new graduate students and staff members can rapidly learn about the environment. Finally, the result, if successful, will be exportable. The present Multics system is a commercially available product, and new ideas developed in the course of this research should be relatively easy to retrofit to the standard Multics system. Since it is intended that a working prototype be constructed, it would also be straightforward for, say, a government agency to request that the manufacturer turn the prototype directly into a production model. The relatively clean organization of the basic system will make the result suitable for study

and imitation by designers of other operating systems in the same way that the present Multics system is already a subject for study and imitation in many locations in the U.S. and abroad. The specific area chosen, namely the security kernel, is one for which there is much interest in understanding and exportation.

The method which will be followed will be to build, by evolution from the standard Multics system, a new version. This new version is to have a "most protected area" which is simpler in organization and much smaller in size than the standard system. The present "most protected area" consists of about 400 modules, or about 80,000 lines of mostly PL/I code. An initial target is to reduce both of these numbers by approximately one order of magnitude. Although a change of this magnitude may seem at first to be unrealistic, there are several reasons why we expect that it is possible:

- Experience in working with Multics over the last several years has proven that its PL/I code and general structure make the system malleable. Recent drastic revisions such as new formats for directories, replacement of software by hardware rings, and even introduction of a new subroutine calling sequence have gone smoothly and quickly.
- Even before a comprehensive review has been undertaken, several specific possible simplifications have already been recognized which can strongly reduce the size of the most protected area. Again, past experience in reviewing and revising such areas as system initialization, interprocess communication, and drum and core management strategy, have proven that the opportunity for a more leisurely review of a design, after the initial rush to get a first draft design completely operational, can yield real insight into the necessary structure, and a resultant simplification and shrinkage of the code required for implementation. In the present system, user terminal management, dynamic linking, and library search

are examples of mechanisms which do not need to be protected from the user to the same degree as, say, the interrupt handlers. Yet, they are so protected, and for the wrong reason: to maintain high performance, since they must communicate intimately with the modules which must be protected, such as shared I/O buffer managers and directory format managers. These are examples in which the full effect of the newly-available ring protection hardware can be easily exploited to segregate into the more protected areas only the functions honestly requiring maximum protection. This ring hardware permits calls to procedures in a protected area to be performed with the same mechanism and at the same cost as a call to a procedure which is not protected.

- In the rush to make Multics fully operational, several well-understood shortcuts were taken. These shortcuts can now be reviewed, and redone, generally producing a more methodical structure. The most significant of this class of change is to permit multiple, parallel, processes in a single address space. Such a change would eliminate a vast number of special tricks used to provide rapid interrupt-time response, to provide a "quit" button on the user terminal, and to coordinate activities which are more simply viewed as parallel ones, such as management of all inactive telephone lines. The primary effect of this change, to reduce all hardware interrupt handlers to a half-dozen lines of code plus one subroutine call, is especially far-reaching in terms of simplicity and auditability of the resulting structure.

In light of these three reasons, it is anticipated that substantial progress toward a full order of magnitude reduction in the size of the most protected area can actually be accomplished. From past experience with such activities, it would appear that a target date of about three years from the time work begins, a prototype system may be available.

An important difference between this kind of activity and the traditional activity of building or modifying an operating system is that when an unexpected problem is encountered, it must be mastered by completely consistent redesign rather than by shortcuts and patches, if simplicity of organization is to be a result. For this reason, we propose to work in an environment free from deadlines and expectations by users of products to be delivered on a firm schedule. To accomplish this change of environment, a method of operation distinctly different from that used in the past to develop Multics itself is proposed. In the past, as rapidly as innovations were checked out, they were delivered and installed in the standard system, to be used immediately at M.I.T., and after some delay at the other sites operating the Multics system. This pattern will probably continue to be the method of Honeywell with respect to the Multics system. However, for purposes of this project, we propose to split off and develop a distinct version of the operating system, unhampered by compatibility requirements and day-to-day operational needs of the various Multics sites. (Of course, our Honeywell counterparts may well wish to pick up ideas, techniques, or even changed modules, with the intent of adapting them to operate in the standard Multics systems.)

As suggested above, it is planned that this project be carried out jointly with the Cambridge Information Systems Laboratory of the Honeywell Company. The manager of that laboratory has expressed interest, and a separate proposal to that effect will be made to Honeywell. The level of effort proposed would include six to eight professional programmers (half of which would come from Honeywell), about eight graduate research assistants, half a dozen undergraduate students, and parts of the time of three faculty members.

In addition to cooperation with Honeywell, close coordination with several other activities will be maintained. Most important of these are the "software assurance project" at

University of Southern California Information Sciences Institute, the computer security activities in the Air Force Electronics Systems Division, and the RISIS project at the Lawrence Livermore Laboratory. Discussions with the people in charge of these three projects have produced a general agreement to remain in close contact. At the present time, it would appear that the four organizations (including MAC/CSR) can supply distinct, unique and interlocking capabilities:

1. ISI can suggest the constraints on an operating system design which will make auditing possible, and can help develop standards for program organization and structure which will make auditing easier.
2. AF/ESD can provide the constraints on an operating system design needed by a user who has both operational and security requirements.
3. LRL can provide a tiger team which can look for weakness in the system itself.
4. MAC/CSR can provide the experience in the real world of operating system construction to make sure that the prototype which results has a suitable functional capability, reasonable performance, and is maintainable.

As a target, the following objective is proposed: the actual prototype operating system should consist of a small set of structured programs with annotation and commentary to make it easy for a reader to understand, and suitable for publication in a technical report or possibly even in the "Algorithms" section of the Communications of the ACM. On the other hand, this publishable system kernel should not restrict the present functional capabilities of the full Multics system in any important way.

A substantial amount of computation resources will be required. Time and storage space on the M.I.T. Information Processing Center Multics site are the primary requirement, to

allow editing, compiling, generating and storage of test systems. In addition, time on some distinct, small configuration of hardware will be required for checkout and performance benchmarking of test systems.

System Measurement and Modeling Activities

An exceptionally productive area for the last year or so has been in taking measurements of the M.I.T. Multics site, and developing models which track these measurements. There are several reasons for this productivity:

- The M.I.T. Multics site is now being used in production by some 700 individuals, for a wide variety of purposes. Thus we are able to observe a realistic working computer facility.
- Almost all programs run at M.I.T. have been written new for Multics, and are designed for its advanced programming environment. Most other computer systems in production use have loads consisting at least in part of programs designed for other systems, which makes interpretation of measurements difficult.
- The relatively methodical organization of the Multics supervisor, and the fact that it is written largely in PL/I, make addition of meters and interpretation of measurements a reasonable project at the graduate student level. There is therefore a pool of available and willing talent to apply to measurement and modeling problems.
- The move to a new generation of hardware, underway now, will provide an opportunity for systematic comparison measurement and algorithm evaluation in a new operating region.

Thus we expect that the measurement and modeling area is one in which a continued modest investment will yield substantial results.

Extensions of the ARPA Network attachment to Multics

The basic objective of this activity is to make the ARPA network the preferred method of information flow in and out of a Multics-class system. A secondary objective is to feed back into the network designs the special concerns of a general-purpose computer utility which is being used as a service system. Some users have already found that the inherent flexibility of the ARPA network terminal interface is superior to the direct terminal attachment interface of Multics. For example, display terminals operating above 2000 bits per second and typewriter terminals operating at 300 bits per second can be used only via the ARPA net at present. The logical extension of this goal is to develop the capabilities of the network attachment to the point that:

- Line printers, card readers, and card punches are readily attachable via the network.
- Terminal I/O buffering for the network is as effective as for directly attached terminals.
- Display systems of high bandwidth can be used over the network.
- User magnetic tapes are attachable, via the network.
- Ultimately, the need for operators in the computer machine room is eliminated, as all I/O operations have been decentralized, and placed in the user's hands. (The hardest function to automate is that related to initiating the correct degree of backup/restart following a system hardware or software failure.).

To this end, several specific activities are proposed:

1. Input and output buffering strategies for the ARPA network are primitive, and not suitable for high traffic, in contrast to direct attachment teletype software, which by now uses a highly optimized buffering strategy. The ARPA net buffering strategy is complicated by the need, on the one hand, to

drive terminals attached to TIP's with negligible internal buffering, and on the other hand to pass large files to other systems with arbitrarily large buffer capabilities. Quite a bit of additional work is needed to develop, implement, and install a strategy which will handle both ends of this spectrum effectively. The strategy must also be capable of supporting character-at-a-time and full duplex use of Multics.

2. Attachment of the Project MAC display system to the ARPA network will introduce, for the first time, large numbers of high bandwidth devices, all attempting to access one or a few hosts. In addition, evolution of the ARPA net itself will cause more attention to turn to development of graphics protocols. Participation in the development and also implementation of the graphic protocols for Multics is planned as well as the deeper integration of graphical mechanisms into the system itself.
3. Initial planning will begin on the problem of allowing files which appear in the directories (catalogues) of one network host to be physically stored at a different network host. This idea is a step beyond currently proposed "file transfer protocols" in that the user of such a file would see nothing exceptional compared with access to a file which happened to be stored locally. The tricky problem here is to integrate this view into the segmented virtual memory view that all on-line storage is directly addressed and shareable. Planning and development of suitable protocols is proposed.

In addition to those specific activities, the CSR division of Project MAC will continue to be an active participant in Network Working Group activities, lending its experience in operating system design and its viewpoint as a general-purpose server site to the various activities of the NWG. This activity will primarily be reflected in travel and borrowing of time of people nominally assigned to other network activities.

Finally, CSR will continue to administer a small fund for experimental use of the Multics system by ARPA-sponsored network users outside M.I.T. The purpose of this fund is two-fold:

1) To permit other network sites to feel that they may experiment with or demonstrate their own capabilities by remote use of the Multics system.
2) To permit other network users to get onto and evaluate Multics as a possible host for their own work. (Any productive work which follows such an initial introduction would normally be funded by direct arrangements between the other network site and the M.I.T. Information Processing Center.)

Exporting the Multics Design

Although the development and implementation of the initial Multics system is now complete, the transfer of the know how and ideas embodied in the system to other system designers is not. To this end, a small activity, included in this proposal to establish its legitimacy, is planned to continue support of writing of technical papers and books about the Multics design, and offer space and computer time to permit visitors to examine and use the Multics system to learn how it is constructed.