A SIMPL Distributed Operating System and Its Formal Definition

R. Noonan, V. Basili, R. Hamlet, M. Lay, D. Mills, J. Turner, M. Zelkowitz
Computer Science Center, University of Maryland

In recent years there has been a phenomenal growth of interest in networks of computers sharing both data and programs. For example, Cmdr. Grace Hopper has advocated a hierarchical network of minicomputers to perform the data processing function in which the tasks to be done would be distributed over the network, with only summary or extract information being passed from one level to another. While the hardware exists to build such a network, the software mechanisms to effectively (from both a time and cost viewpoint) control such a network have not yet been developed.

At the Computer Science Center of the University of Maryland a distributed operating system, which is independent of the specific number and hardware of the particular computers it is running on, is being developed and implemented. Supporting this project is the development of two linguistic tools: the design and implementation of a systems programming language called SIMPL and the use of formal semantics in the specification of both SIMPL and the distributed operating system. So far, each of these three elements has had a considerable effect on the others.

The operating system will be coded in SIMPL. In fact, this language is intended to be used for systems work on both the Center's UNIVAC 1108 and DEC PDP-11/45. Unlike ALGOL 68, SIMPL itself is not an attempt to push forward the boundaries of language design. Rather it is an attempt to incorporate what has been learned about programming and algebraic languages since the introduction of FORTRAN in the late 1950's. As a language, SIMPL resembles both BLISS and NUCLEUS.

In the design of the language, a number of sometimes conflicting goals had to be resolved. These goals included the following:

1. SIMPL was based on the principles of structured programming. Certain features, notably the go to statement, were abolished from the language.
2. The language was to serve as a student programming language. It must be simple and easy-to-learn and must encourage good programming habits.
3. The language must be free-format and yet have a simple syntax. Because of student difficulties in learning ALGOL, the obnoxious semicolon was abolished.
4. The language was to be used as a systems programming language. It thus includes both bit and shift operators. However, in order to preserve machine-independence, it did not include the capability of addressing registers or of inserting machine language instructions.
5. As a systems language, only features which could be efficiently implemented were allowed. Thus, for example, block structure was not included in SIMPL.
6. The language will be used for the verification of programs. This influenced the design considerably. For example, functions may not have side-effects and may not be recursive.
7. Later versions of the language will include an assert statement. Assertions which cannot be verified at compile time will cause run time tests to be inserted when compiled in debugging mode.
8. Automatic traceback and subscript checking are provided in the compiler. Various trace statements can be compiled or omitted depending on the trace mode selected.

Features of the language include both internal and external procedures, and arithmetic assignment, if, case, call, while, return, and exit statements. Simple parameters are passed either by value or by reference and arrays are passed by reference. Simple variables and arrays may be declared external. The usual arithmetic operators have been included but only integer arithmetic and one dimensional arrays have been implemented.

At the current time two versions of SIMPL are running on the UNIVAC 1108; one produces code for the 1108, while the other for the PDP 11. To minimize the differences between the two implementations, both a reference language and a hardware language have been defined.

The second major element of this effort is research in the area of semantic models. A language called HGL (Hierarchical Graph Language) has been developed based on the earlier work on H-graphs due to Pratt (1969) and Basili (1970). HGL is essentially a structured programming model; despite this, it is capable of modeling any programming language feature, including go to statements.

In HGL the basic structure is a graph. The contents of a particular node can be either a data item or a graph (possibly the one containing this node). In addition, a graph may have attributes associated with it; an important attribute of a graph is its entry node. The basic operators of HGL are the basic graph operators, such as positive adjacency, positive incidence, etc., extended to hierarchical graphs. As a language HGL is functional and LISP-like.

The principal advantages of HGL over VDL is that it preserves the natural program topology as a graph and that graph structures are a more natural representation of data structures than the tree structures of VDL. In order to compare the two, formal definitions of SIMPL have been produced in both HGL and VDL. Several definitions were produced in HGL and evaluated as implementation strategies. In fact, this should be a major use of a model and of a formal definition. This was one of the major defects of VDL, however; the definition developed depended on a dump mechanism (as in the formal definition of PL/I) even though this was totally inappropriate for a non-block structured language. The best HGL model, on the other hand, relied on a more natural stack definition.

The use of HGL in modeling various language constructs has had a major impact on the evolution of SIMPL. For example, the study of various models of block structure in HGL led to the conclusion that block structure is an unnecessarily complicated mechanism to achieve independence of names. Hence, this feature was not included in SIMPL, which relies instead on named procedures. It was felt that the latter was more in keeping with the precepts of structured programming, especially with regard to limiting the length of a given routine (as advocated by E. Dijkstra and H. Mills).

Another use of HGL will be a proof of correctness of the implementation of SIMPL using the twin model approach as developed by the IBM Vienna group.

Both SIMPL and HGL are being used in the design and implementation of a distributed operating system for a network of minicomputers (Lay, 1973). In this system network resources are provided and shared through segment creation and transmission, where segments can be either processes or messages. Processes are given capabilities to influence their own environment as well as that of other processes through message communication. These facilities are constructed so that processes need not be aware of the hardware configuration (possibly distributed over several minicomputers) upon which they are run.

This project will be implemented on the Center's PDP 11/45 and will be principally coded in SIMPL. The system at the current time has 2 teletypes and has a low speed link with the Center's 1108, which is used for compilation. A Canberra 2020 (3 unit cassette) tape unit, which is being programmed to be used like DECtape, is the primary mass storage device. Thus, the system will have a low speed, random access capability.

The use of SIMPL as the prime systems language is expected to have a major impact on the system. For example, the compiler will produce location-independent, absolute code. Linking will be handled by the monitor at run-time; this will obviate the need for a linkage editor, at least for systems modules. It will also eliminate the need for overlaying, even though the system is already paged (a PDP 11/45 has 8 I bank registers and 8 D bank registers). Only a simple loader and monitor will need to be core resident, with other modules being loaded and linked as necessary.

It is planned that a formal description of this operating system will be produced in HGL. This definition will be used in order to prove the correctness of the operating system. It is expected that this effort will be beneficial both for the distributed operating system and HGL itself.

Summary

In this paper we have briefly described a distributed operating system which is currently being designed and implemented. To support this project a systems implementation language called SIMPL has been developed in which to code this operating system. A language called HGL in which formal models can be defined has been developed. HGL has been used to formally define SIMPL and will be used to formally define the distributed operating system. It is expected that all three efforts will continue to have a considerable influence on each other.

References

1. Basili, V., "A Semantic Model for Programs," Ph.D. dissertation, University of Texas at Austin, TSN-9, (Jan., 1970).

2. Basili, V., "SIMPL-X: A Language for Writing Structured Programs," U. of Maryland, TR-223, (1973).

3. Lay, M., Mills, D., and Zelkowitz, M., "A Distributed Operating System for a Virtual Segment Network," AIAA Conf. on Comp. Network Systems, (April, 1973).

4. Pratt, T., "A Hierarchical Graph Model of Programming Language Semantics," Proc. SJCC, (1969).